

# **Spatial Verification and Validation of Datasets in Fluid Dynamics**

IAN WATSON

A thesis submitted in fulfilment of  
the requirements for the degree of  
Doctor of Philosophy,  
July 2010.



School of Mechanical and Manufacturing Engineering  
The University of New South Wales

THE UNIVERSITY OF NEW SOUTH WALES  
Thesis/Dissertation Sheet

Surname or Family name: **WATSON**

First name: **Ian**

Other name/s: **Thomas**

Abbreviation for degree as given in the University calendar: **Ph.D.**

School: **Mechanical and Manufacturing Engineering**

Faculty: **Engineering**

Title: **Spatial Verification and Validation of Datasets in Fluid Dynamics**

**Abstract 350 words maximum: (PLEASE TYPE)**

There is a present need for post-processing tools capable of synthesising and interpreting the numerous spatial data that are typically generated in modern investigations of fluid mechanics. Recent advances have provided both the analyst and the experimentalist with powerful tools for resolving complete flow-field information, using Computational Fluid Dynamics to simulate the flow, or non-invasive flow metrology such as Particle Image Velocimetry and Laser Doppler Anemometry. A great deal of nodal data is generated by these techniques, which quantity may be expected to increase into the future. This data comprises uncertainties in both numerical modelling and experimental measurement, which traditionally have been quantified using classical approaches in Verification and Validation. However, these techniques were designed with summary scalar values in mind and generally overlook or underestimate the importance of suitable spatial and topological description of the flow-field. The author uses established techniques in geostatistics to address the fluids data assimilation problem, and cross-correlate spatial field variables collected over an experimental domain with field variables calculated by a numerical model that simulates this domain. Spatial statistics are generated on the inter-related nodal data, and are used to inform a stationary covariance model describing the datasets as a particular realisation of a random process. This model is used to provide statistics quantifying the correlation of complete experimental and numerical flow-fields, and make better estimations of local field values taking into account the sum data that is available to the practitioner. Special consideration is given to the application of the random function error model to a calculated flow-field, in which errors are not aleatoric but epistemic, and comprise unknown chaotic processes and higher-order error terms. The kriging estimator was useful for the characterisation of the spatial datasets considered, and may be expected to extend quite generally to other fluids problems. In particular, meaningful blending of experimental and numerical data was achieved by cokriging, and is demonstrated in situations where experimental data is missing or sparse but may be inferred by the secondary numerical data with which it is well correlated. A statistic describing whole-field correlation on the basis of functional covariance was also proposed for fluids problems, with reference to which it is demonstrated that traditional pointwise measures of disparity are inadequate for spatial problems.

**Declaration relating to disposition of project thesis/dissertation**

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).



Signature



Witness

25/8/2010

Date

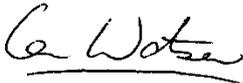
The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

**FOR OFFICE USE ONLY**

Date of completion of requirements for Award:

### **ORIGINALITY STATEMENT**

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed:  Date: 25/08/2010

Ian Watson.



# Abstract

There is a present need for post-processing tools capable of synthesising and interpreting the numerous spatial data that are typically generated in modern investigations of fluid mechanics. Recent advances have provided both the analyst and the experimentalist with powerful tools for resolving complete flow-field information, using Computational Fluid Dynamics to simulate the flow, or non-invasive flow metrology such as Particle Image Velocimetry and Laser Doppler Anemometry. A great deal of nodal data is generated by these techniques, which quantity may be expected to increase into the future. This data comprises uncertainties in both numerical modelling and experimental measurement, which traditionally have been quantified using classical approaches in Verification and Validation. However, these techniques were designed with summary scalar values in mind and generally overlook or underestimate the importance of suitable spatial and topological description of the flow-field. The author uses established techniques in geostatistics to address the fluids data assimilation problem, and cross-correlate spatial field variables collected over an experimental domain with field variables calculated by a numerical model that simulates this domain. Spatial statistics are generated on the inter-related nodal data, and are used to inform a stationary covariance model describing the datasets as a particular realisation of a random process. This model is used to provide statistics quantifying the correlation of complete experimental and numerical flow-fields, and make better estimations of local field values taking into account the sum data that is available to the practitioner. Special consideration is given to the application of the random function model to a calculated flow-field, in which errors are not aleatoric but epistemic, and comprise unknown chaotic processes and higher-order error terms. The kriging estimator was useful for the characterisation of the spatial datasets considered, and may be expected to extend quite generally to other fluids problems. In particular, meaningful blending of experimental and numerical data was achieved by cokriging, and is demonstrated in situations where experimental data is missing or sparse but may be inferred by the secondary numerical data with which it is well correlated. A statistic describing whole-field correlation on the basis of functional covariance was also proposed for fluids problems, with reference to which it is demonstrated that traditional pointwise measures of disparity are inadequate for spatial problems.

# Acknowledgements

I wish there was a better way of thanking and remembering the crowd of characters who have variously humoured or harangued me for the last six odd years, but I guess I'll do it in the normal way: old songs are good songs, and a million Elvis fans can't be too far wrong, and anyway, I'm scared of getting busted for flouting UNSW formatting guidelines.

Firstly an apology. Sorry Eddie that I could not get this one finished before you passed away, I never got the chance to thank you for your advice and unswervingly kicking my arse into action, especially in the early days when I needed it most, and heeded it least. To my surviving supervisors Tracie and Don, thank you for your infinite reserves of patience and good-humour and all the best into the future: Don, stay away from golf – it takes time away from structural analysis, and Tracie, best fun being a mum for the *second* time – you'll be great with all the practice we've given you over the years! Thanks also to all the other wise ones around the department for all of your input – a big cheers to John for 'getting it', improving my work and correcting my ingles, cheers Dick for crucifying Beethoven in the summertime, thanks Victoria for providing as-it-turned-out-not-so-temporary digs, a big thankyou to Carl for bailing me out and to Ganga for putting me back in. My gratitude is also due to everybody in the Lab, including my erstwhile journeymen in the old 505 – a special thanks to Sam Diasinos and Chris Beves for their kind permission to use their datasets for algorithm testing and development: this thesis would have been impossible without them. Many thanks to team CFD; Graham (c), Osama, Oscar, JP, Jono, Ross, Mark, Sam L., Sam D., and even the randoms for advice, restorative social soccer and field trips to Coogee and the Donkey. On that note, thanks to my learned colleagues Simon, Gareth, Garth, Luke, Rob, Raju, Rozetta, Mahmoud, Jad, Zoltan et al for much needed research-sympathy. Thanks Dave for your crantastic insights into, well, everything. A big Aufbonwiederjourciaobelli! to all the crazy Eurokids we've had through the office; Vince, Delphine, Ale (G. and extras...), Esther, Basti and the teufels, Fortran-buddy Max, Francule, Lea and Fabien: research never sleeps, but sometimes it takes breaks! I also owe a debt of gratitude (and cash) to the dedicated and professional crews at Kensington

Physio, St. Vincent's, Royal North Shore, and Prince of Wales hospitals; with a special thanks to Dr. Parker and Dr. Levi – thanks in advance to Dr. Maher! You guys are the real doctors.

But seriously, and as is traditional, thankyou to my family – cheers to dad for talking boundary elements in the pub, and thankyou mum for all the fish, and even the career advice: with parents like these, who needs children! Thankyou grandpa Watson for providing an early model of stoicism and basic bloody-mindedness which is generally useful in research. Thanks to all the brothers and sisters Lee for stupid generosity, and understanding why me and James do what we do. Thanks James for comic relief – take care, and just say no to PhDs. Finally thanks to my old friends and good friends, from all walks, who've delivered me thus far – sorry it took so long. It's been great fun watching you sleet off into the professional world, crafting the future successes etc., and embarrassing you from the sidelines, but now I fear, it may be time for me to join you! You know who you are and I know where you live, so I won't bother naming names, but you can add yourselves below, if you like. Thanks...



# Table of Contents

<b>Abstract .....</b>	<b>iv</b>
<b>Acknowledgements.....</b>	<b>v</b>
<b>Table of Contents .....</b>	<b>viii</b>
<b>Table of Figures.....</b>	<b>xiii</b>
<b>Nomenclature.....</b>	<b>xviii</b>
<b>Chapter I - Introduction .....</b>	<b>1</b>
I - 1.        State of the Art.....	3
I - 2.        Numerical Uncertainty.....	6
I - 3.        Original Contribution.....	8
I - 4.        Structure of the Dissertation .....	10
<b>Chapter II - Literature Review .....</b>	<b>13</b>
II - 1.        Verification Activities .....	13
II - 2.        Validation Activities .....	19
II - 3.        Sensitivity Analysis .....	22
II - 4.        Response Surfaces and Surrogate Models .....	25
II - 5.        Kriging Interpolation .....	27
II - 6.        Filtering for Meteorological Simulations.....	29
II - 7.        Motivation for Geostatistics and Kriging .....	30
<b>Chapter III - Theoretical Overview.....</b>	<b>33</b>

III - 1.	Random Functions and Estimation .....	33
III - 2.	Kriging.....	41
III - 3.	Variogram Modelling .....	46
III - 4.	Universal Kriging .....	52
III - 5.	Structure Identification .....	55
III - 6.	Multivariate Estimation – Cokriging .....	61
III - 6.1.	Independent drift functions.....	63
III - 6.2.	Algebraically dependent drifts.....	65
III - 7.	Cross-covariant Structure Identification .....	66
III - 7.1.	Cross-covariant statistics .....	67
III - 7.2.	Linear model of coregionalisation .....	69
<b>Chapter IV - Implementation .....</b>		<b>72</b>
IV - 1.	Overall Architecture .....	73
IV - 2.	Auto-covariant Statistics.....	77
IV - 3.	Cross-covariant Statistics.....	81
IV - 4.	Calculation of Statistics .....	82
IV - 4.1.	Lag generation and splitting scheme .....	83
IV - 4.2.	Spatial statistics .....	86
IV - 5.	Fitting a Covariance Model .....	89
IV - 5.1.	Basic variogram model fitting.....	92
IV - 5.2.	Fitting the linear model of coregionalisation.....	98
IV - 5.3.	Coregionalisation modelling options.....	100
IV - 6.	Estimation .....	105
<b>Chapter V - Case Study 1: Numerical investigations .....</b>		<b>112</b>
V - 1.	Convection in a Square Cavity .....	114

V - 1.1.	Description of flow.....	115
V - 1.2.	Covariance function behaviour.....	118
V - 2.	Generalised Covariance Theory.....	122
V - 3.	The Poisson Equation.....	125
V - 3.1.	Problem description.....	125
V - 3.2.	Generalised covariance behaviour.....	127
V - 4.	Afterword.....	136

**Chapter VI - Case Study 2: Inverted wing and rolling wheel..... 140**

VI - 1.	A Validation Scenario.....	141
VI - 2.	Kriging and Smoothing.....	146
VI - 2.1.	Univariate estimation.....	147
VI - 2.2.	Cross-validation.....	154
VI - 3.	Spatial Similarity.....	156
VI - 4.	Cokriging and Blending.....	168
VI - 4.1.	Cokriging in estimation of the raw field data.....	168
VI - 4.2.	Cokriging for the inference of modified field data.....	176
VI - 5.	Variance Reduction.....	179
VI - 6.	Alternative Schemes.....	182
VI - 6.1.	Common drift functions.....	182
VI - 6.2.	Interpolated drift functions.....	186
VI - 7.	Afterword.....	188

**Chapter VII - Conclusions..... 190**

VII - 1.	Findings.....	191
VII - 2.	Significance.....	193
VII - 3.	Future Work.....	194

VII - 3.1.	Covariance modelling .....	194
VII - 3.2.	Extension.....	196
VII - 3.3.	Numerical uncertainty.....	198
<b>References .....</b>	<b>199</b>	
<b>Appendix A - Demonstration: MATLAB listing.....</b>	<b>206</b>	
<b>Appendix B - Code Functionality .....</b>	<b>214</b>	
B - 1.	Program krige .....	214
B - 1.1.	Raster interpolation.....	215
B - 1.2.	Structure identification .....	218
B - 1.3.	Cross-validation.....	219
B - 1.4.	Input keywords .....	220
B - 2.	Program datagen .....	224
<b>Appendix C - Algorithm Details .....</b>	<b>227</b>	
C - 1.	k-means Clustering Algorithm.....	227
C - 2.	Levenberg-Marquardt Algorithm.....	229
<b>Appendix D - Derivations .....</b>	<b>233</b>	
D - 1.	Derivation of Ordinary Kriging Variance .....	233
D - 2.	Unbiasedness for Ordinary Kriging .....	234
D - 3.	Derivation of Universal Kriging Variance .....	235
D - 4.	Unbiasedness for Universal Kriging .....	236
D - 5.	Derivation of Cokriging Variance.....	237
D - 6.	Jacobian for Basic Variogram Model Fit.....	239
D - 7.	Jacobian for Coregionalisation Model Fit.....	242

D - 8.	Jacobian for Penalty Functions .....	244
D - 9.	Laplacian Variance in Terms of Generalised Covariance.....	245
<b>Appendix E - Program Notes .....</b>		<b>249</b>
E - 1.	Style .....	249
E - 2.	Module dataset.....	251
E - 3.	Module antekrige .....	263
E - 4.	Module inter .....	266
E - 5.	Module krige.....	275
E - 6.	Module optimisation.....	300
E - 7.	Module sample .....	302
<b>Appendix F - Publications .....</b>		<b>310</b>

# Table of Figures

II—1: Comparing continuous functions.....	31
III—1: A Random Walk.....	34
III—2: Raw coin-tossing events.....	36
III—3: Measurements of streamwise air velocity in a turbulent wake .....	38
III—4: Sample covariance function for LDA data .....	39
III—5: Kriged interpolation of LDA data.....	42
III—6: Schematic of estimation domain .....	43
III—7: Variogram models.....	46
III—8: Fitting model covariance functions.....	48
III—9: Kriged interpolation of LDA data (detail) .....	49
III—10: Sample variogram for LDA data.....	56
III—11: Discrete variogram for the random walk .....	58
III—12: ALC-1 Laplacian ‘template’ .....	59
IV—1: Architecture of the kriging algorithm .....	74
IV—2: Schematic pairings for auto-covariant statistic.....	78
IV—3: Schematic pairings for cross-covariant statistic.....	81
IV—4: Splitting scheme for spatial statistics.....	83
IV—5: Splitting algorithm and criteria.....	84
IV—6: Examples of sample auto-correlation functions.....	86
IV—7: Examples of sample cross-correlation functions .....	87
IV—8: Fitting a Gaussian function to sample points.....	89
IV—9: Generating a covariance model from spatial statistics.....	90
IV—10: Pseudo code for basic model identification .....	93

IV—11: Pseudo code for coregionalisation fitting .....	104
IV—12: Generating local subsets .....	106
IV—13: Local subset merging sequence .....	108
V—1: Convection in a differentially heated square cavity .....	114
V—2: Convection in a square cavity; $x$ -velocity .....	115
V—3: Convection in a square cavity; $y$ -velocity .....	115
V—4: Convection in a square cavity; pressure .....	116
V—5: Convection in a square cavity; temperature .....	116
V—6: Correlation function for $x$ -velocity .....	118
V—7: Correlation function and variogram for temperature .....	119
V—8(a) Total Gaussian component vs. grid-size; $Ra=10^3$ .....	120
V—8(b,c) Total Gaussian component vs. grid size; $Ra=10^5, 10^6$ .....	121
V—9: Laplacian template at different scales .....	124
V—10: Boundary conditions and contours for simple heat problem.....	126
V—11: Variance of Laplacian template for $20 \times 20$ grid .....	127
V—12: Generalised variogram over different grids for case one .....	128
V—13: Generalised variogram over different grids for case one (3D).....	129
V—15: ALC frequency distribution for case one; $50 \times 50$ .....	131
V—16: ALC frequency distribution for case one; $20 \times 20$ .....	131
V—14: Generalised variogram over different grids for case two .....	130
V—17: Lag vector generation.....	131
Table V-1: Fit coefficients for case one .....	132
V—18: ALC frequency distribution for case two; $50 \times 50$ .....	133
V—19: ALC frequency distribution for case two; $20 \times 20$ .....	133
Table V-2: Fit coefficients for case two .....	134
V—20: Convergence plot for generalised variogram .....	135

V—21: Alternative ALC-1 .....	138
VI—1: Schematic drawing of wing/wheel geometry.....	141
VI—2: 3D Schematic of wing-wheel configuration .....	142
VI—3: Raw $x$ -velocity for LDA and $k-\omega$ CFD; slice 1 .....	143
VI—4: Raw $x$ -velocity for LDA and $k-\omega$ CFD; slice 2 .....	143
VI—5: Raw $x$ -velocity for LDA and $k-\omega$ CFD; slice 3 .....	144
VI—6: Raw $x$ -velocity for LDA and $k-\omega$ CFD; slice 4 .....	144
VI—7: Raw $x$ -velocity for LDA and $k-\omega$ CFD; slice 5 .....	144
VI—8: Smoothing properties and the nugget effect; estimates .....	146
VI—9: Smoothing properties and the nugget effect; standard deviation.....	146
VI—10: Triangulation for contours on slices 1 and 2.....	147
VI—11: Sample statistics and covariance function fit.....	148
VI—12: Kriged contours of LDA and $k-\omega$ CFD $x$ -velocity; slice 1 .....	149
VI—13: Kriged contours of LDA and $k-\omega$ CFD $x$ -velocity; slice 2 .....	149
VI—14: Kriged contours of LDA and $k-\omega$ CFD $x$ -velocity; slice 3 .....	149
VI—15: Kriged contours of LDA and $k-\omega$ CFD $x$ -velocity; slice 4 .....	150
VI—16: Kriged contours of LDA and $k-\omega$ CFD $x$ -velocity; slice 5 .....	150
VI—17: LDA and $k-\omega$ CFD kriging standard deviation; slice 1 .....	151
VI—18: LDA and $k-\omega$ CFD kriging standard deviation; slice 2 .....	151
VI—19: LDA and $k-\omega$ CFD kriging standard deviation; slice 3 .....	152
VI—20: LDA and $k-\omega$ CFD kriging standard deviation; slice 4 .....	152
VI—21: LDA and $k-\omega$ CFD kriging standard deviation; slice 5 .....	152
VI—22(a,b) Cross-validation for univariate kriging; slices 1, 2 .....	154
VI—22(c-e) Cross-validation for univariate kriging; slices 3-5 .....	155
VI—23: Kriging correlation coefficient $\eta$ .....	156
VI—24: CFD $x$ -velocity (other models); slice 1 .....	157

VI—25: CFD $x$ -velocity (other models); slice 2.....	158
VI—26: CFD $x$ -velocity (other models); slices 3 .....	159
VI—27: CFD $x$ -velocity (other models); slice 4.....	160
VI—28: CFD $x$ -velocity (other models); slice 5.....	161
VI—29: Comparison of $z$ -velocity on slice 1.....	162
VI—30: Average sum-of-squares difference ( $SOS$ ).....	163
VI—31: Total pointwise correlation ( $RHO$ ) .....	164
VI—32: Comparison of $y$ -velocity on slice 5 .....	166
VI—33: Comparison of $z$ -velocity on slice 5.....	166
VI—34: LDA estimates cokriged with $k-\omega$ CFD; slice 1 .....	167
VI—35: LDA estimates cokriged with $k-\omega$ CFD; slice 2 .....	167
VI—36: LDA estimates cokriged with $k-\omega$ CFD; slice 3 .....	169
VI—37: LDA estimates cokriged with $k-\omega$ CFD; slice 4 .....	169
VI—38: LDA estimates cokriged with $k-\omega$ CFD; slice 5 .....	169
VI—39: LDA estimates cokriged with Spalart-Almaras CFD .....	170
VI—40: LDA estimates cokriged with $k-\varepsilon$ realisable CFD.....	172
VI—41: LDA estimates cokriged with $k-\varepsilon$ RNG CFD .....	174
VI—42(a-d) Cross-validation for cokriging (independent drifts); slices 1-4 .....	175
VI—42(e) Cross-validation for cokriging (independent drifts); slice 5.....	176
VI—43: Cokriged reconstructed velocities on slice 2 with LDA blanking .....	176
VI—44: Reconstructed velocities under LDA blanking and independent drifts .....	177
VI—45: Reconstructed velocities on slice 2 with coarse LDA sampling.....	178
VI—46: Local reduction in kriging variance for $k-\omega$ cokriging .....	179
VI—47: Average reduction in kriging variance $v$ .....	180
VI—48: LDA estimates cokriged with $k-\omega$ CFD; independent drift model .....	183
VI—49: Spurious cokriging estimation under a common mean.....	184

VI—50: Cross-validation for cokriging (common drifts); slices 1-5.....	185
VI—51: LDA estimates with $k$ - $\omega$ CFD as an interpolated drift function .....	187
B—1: Format for krigout.txt (raster interpolation) .....	215
B—2: Format for krigdia.txt .....	216
B—3: Format for krigrasta.txt .....	217
B—4: Format for krigdr.txt.....	218
B—5: Format for krigout.txt (cross-validation).....	219
B—6: Format for Tecplot files.....	225
C—1: Spatial clustering .....	227
C—2: Lloyd’s algorithm flowchart.....	228
C—3: Damping scheme .....	231
D—1: The Laplacian ALC-1 and lag generation.....	245
Table D—1: Generation of lag (separation) vectors over Laplacian template .....	246
E—1: Passing first-byte addresses in Fortran .....	250
E—2: Unstructured lists in subroutine meercat. ....	257
E—3: Merging pairs of groups in unstructured lists.....	258
E—4: Arrangement of VARM(:, :, : ).....	277
E—5(a) Array COV in krigery; independent drifts .....	283
E—5(b) Array COV in krigery; dependent drifts .....	284
E—6: Swapping scheme for splitting a tile in kr02.bin.....	305

# Nomenclature

Here follows a list of generally used nomenclature throughout the thesis. Note that there are occasional exceptions to the general nomenclature, introduced as *special* nomenclature. This is to aid readability by introducing physical or practical examples in familiar terms.

$\mathbb{R}^N$ .....	$N$ -space of real numbers
$\mathbf{x}, \mathbf{y}$ .....	general spatial vectors in $\mathbb{R}^2$ or $\mathbb{R}^3$
$i, j, k, \alpha, \beta$ .....	general indices
$P(\mathbf{x})$ .....	a stationary continuous random function
$E(\cdot)$ .....	the expectation function
$\text{var}(\cdot)$ .....	the variance function
$\text{cov}(\cdot)$ .....	the covariance function
$\mu$ .....	an expected value or mean
$\sigma^2$ .....	the variance of a random process
$\mathbf{h}$ .....	a lag vector connecting two nodes
$h$ .....	Euclidean length of the lag vector, <i>or</i> more generally – a scale factor for the lag vector template in an allowable linear combination.
$C(\mathbf{h})$ .....	a stationary covariance function
$C(\mathbf{x}, \mathbf{y})$ .....	effectively $C(\mathbf{y} - \mathbf{x})$
$N(\mathbf{h})$ .....	number of extant vector lags $\mathbf{h}$
$N(\cdot)$ .....	number of items in a set
$\mathbf{x}_k$ .....	nodal locations
$p_k$ .....	values at nodal locations $\mathbf{x}_k$ , or a realisation of the regionalised random variable $p_k \sim P(\mathbf{x}_k)$

**p** .....a vector of such nodal values  
 $w_k$ .....the weight corresponding to  $p_k$   
**w**.....a vector of the weights  
 $\mathbf{x}_0$  .....the point where an estimate is produced  
 $\hat{p}_0$  .....the estimate/estimator of the value  $p_0 \sim P(\mathbf{x}_0)$   
 $\frac{2}{K}$  .....kriging variance (or estimation variance)  
 $\frac{2}{P}$  .....total variance of the random process  $P(\mathbf{x})$   
**C**.....a left-hand-side (LHS) covariance matrix relating nodal values  
**c**<sub>0</sub>.....a right-hand-side (RHS) covariance vector expressing covariance with the point  $\mathbf{x}_0$   
 $\gamma_X(\cdot)$ .....a normalised canonical variogram model of type  $X$   
 $\gamma^i(\cdot)$ ..... $i^{\text{th}}$  normalised canonical variogram used in a model  
 $t_i$ .....coefficient multiplying  $i^{\text{th}}$  canonical variogram  
 $\Phi_i$ ..... $i^{\text{th}}$  vector norm on the lag input  $\mathbf{h}$   
 $\mathbf{T}^i$  .....positive semi-definite matrix for vector norm  $\Phi_i$   
 $\mu(\mathbf{x})$ .....underlying mean of random function or *drift*  
 $f^i(\mathbf{x})$ .....basis function in drift  $\mu(\mathbf{x})$   
 $a_i$ .....coefficient multiplying basis function  $f^i(\mathbf{x})$   
**a**.....vector of such coefficients  
 $Q(\mathbf{x})$  .....a second-order stationary random function, mean  $\mu(\mathbf{x})$   
 $q_i$ .....a realisation of regionalised random variable  $q_i \sim Q(\mathbf{x}_i)$   
 $\hat{q}_0$ .....estimate/estimator of  $q_0 \sim Q(\mathbf{x}_0)$   
 $m^i(\mathbf{x})$  .....the monomial basis functions of order  $K$   
 $\mathbf{f}^i$ .....vector of basis functions evaluated at  $\mathbf{x}_i$

**F** ..... matrix comprising a concatenation of such vectors  $\mathbf{f}^i$   
**Z**..... an intrinsically random function  
**Z( $\lambda$ )** ..... a metric on **Z**  
 $\lambda_i$  ..... coefficients in the allowable linear combination  
 $\mathbf{h}_i$ ..... vector lags in the allowable linear combination  
**ALC- $k$** ..... an allowable linear combination of order  $k$   
**IRF- $k$** ..... an intrinsically random function of order  $k$   
**Poly $^k(\cdot)$** ..... polynomial function of order  $k$   
**K( $\mathbf{h}$ )**..... generalised covariance function  
 $C_0, b_1, b_2, b_3$ ..... coefficients in canonical, isotropic **K( $\mathbf{h}$ )** form  
 $\delta(h)$  ..... nugget effect discontinuity function  
 $\Gamma(h)$ ..... generalised variogram on a Laplacian operator  
 $\rho(\mathbf{h})$  ..... stationary autocorrelation function  
 $P^\alpha(\mathbf{x})$  .....  $\alpha^{\text{th}}$  stationary random function in a cokriging  
 $\mathbf{x}_k$ , abbreviated  $\mathbf{x}_k$ ..... nodal locations in a dataset pertaining to  $P^\alpha$   
 (abbreviated as dataset  $\alpha$  is evident from context)  
 $p_k$  ..... values at nodal locations  $\mathbf{x}_k$ ; realisations  $p_k \sim P(\mathbf{x}_k)$   
 $\mathbf{p}^\alpha$  ..... a vector of such realisations  
 $w_k$  ..... the weight corresponding to  $p_k$   
 $\mathbf{w}^\alpha$  ..... a vector of such weights  
 $\hat{p}_0$  ..... estimator/estimate of the value  $p_0 \sim P(\mathbf{x}_0)$   
 $C_{\alpha\beta}(\mathbf{x}, \mathbf{y})$ ..... cross-covariance function between  $P^\alpha(\mathbf{x})$  and  $P^\beta(\mathbf{y})$   
 $C_{\alpha\beta}(\mathbf{h})$ ..... stationary cross-covariance function where  $\mathbf{h} = \mathbf{y} - \mathbf{x}$   
 $\overset{2}{C}_{CK}$  ..... cokriging variance  
 $\mathbf{C}_{\alpha\beta}$ ..... LHS covariance submatrix relating  $\mathbf{p}^\alpha$  and  $\mathbf{p}^\beta$

$\mathbf{c}_0$  .....RHS covariance vector relating  $\mathbf{p}^\alpha$  to point of estimation  
 $\mu_\alpha(\mathbf{x})$  .....drift function for  $P^\alpha$   
 $f^{ai}(\mathbf{x})$ .....basis function for the drift of  $P^\alpha$   
 $a_{ai}$ .....coefficient multiplying  $f^{ai}(\mathbf{x})$   
 $\mathbf{a}_\alpha$  .....vector of such coefficients  
 $\mathbf{f}_k$  .....vector of basis functions for  $P^\alpha$  evaluated at  $\mathbf{x}_k$   
 $\mathbf{F}^\alpha$  .....matrix concatenating  $\mathbf{f}_k$   
 $\Delta_k^j$  .....Kronecker delta  
 $\mathbf{a}^*$  .....vector of coefficients multiplying a common or dependent set of basis functions  
 $\mu_\alpha$ .....mean value of  $P^\alpha$   
 $\sigma_\alpha^2$  .....total variance of  $P^\alpha$   
 $\sigma_{\alpha\beta}^2$  .....total cross-covariance of  $P^\alpha$  with  $P^\beta$   
 $\rho_{\alpha\beta}(\mathbf{h})$  .....stationary cross-correlation function  
 $\gamma_{\alpha\beta}(\mathbf{h})$ .....stationary cross-variogram function  
 $s^i$  .....coefficient multiplying  $i^{\text{th}}$  variogram model in  $C_{\alpha\beta}(\mathbf{h})$   
 $\hat{C}(\mathbf{h})$  .....a statistic for  $C(\mathbf{h})$   
 $\hat{C}_{\alpha\beta}(\mathbf{h})$  .....a statistic for  $C_{\alpha\beta}(\mathbf{h})$   
 $\mu_{\pm\mathbf{h}}$  .....mean statistic calculated over nodal values at the  $\pm\mathbf{h}$  end of a lag vector  
 $\hat{\rho}(\mathbf{h})$  .....a statistic for  $\rho(\mathbf{h})$   
 $\sigma_{\pm\mathbf{h}}$ .....standard deviation statistic calculated over nodal values at the  $\pm\mathbf{h}$  end of a lag vector  
 $I(\mathbf{h})$  .....a statistic for  $\gamma(\mathbf{h})$

$\hat{P}^2$	.....	a statistic for $P^2$
$\bar{p}$	.....	a statistic for the mean value of the data $p_i$
$\bar{p}_{\pm \mathbf{h}}$	.....	mean statistic on $P^\alpha$ calculated over nodal values at the $\pm \mathbf{h}$ end of a lag vector
$\sigma_{\pm \mathbf{h}}$	.....	standard deviation statistic on $P^\alpha$ calculated over nodal values at the $\pm \mathbf{h}$ end of a lag vector
$\hat{\rho}(\mathbf{h})$	.....	a statistic for $\rho_{\alpha\beta}(\mathbf{h})$
$H$	.....	a list of all possible node pairs between two, possibly congruent, sets of data.
$\mathbf{h}_{ij}$	.....	lag vector $\mathbf{x}_i - \mathbf{x}_j$ connecting two nodes in $H$
$H_X$	.....	binary tree subset of $H$
$\hat{\mathbf{n}}_{\max}$	.....	ordinate direction in which to split the subset $H_X$
$n_{crit}$	.....	critical value of this ordinate along which $H_X$ is split
$T^e$	.....	final subsets $H_X$ after all splits, called tiles
$\bar{\mathbf{h}}^e$	.....	average lag vector over each tile
$S^e$	.....	associated spatial statistic generated on each tile
$\bar{\mathbf{h}}^e$	.....	average lag vector for a spatial statistic relating $P^\alpha$ and $P^\beta$
$S^e$	.....	associated spatial statistic for a spatial statistic relating $P^\alpha$ and $P^\beta$
$\mathbf{r}$	.....	a pseudo-vector of parameters for optimisation
$g(\mathbf{x}, \mathbf{r})$	.....	a spatial function in $\mathbf{x}$ with flexible parameters $\mathbf{r}$
$\{\mathbf{x}_i, G_i\}$	.....	set of nodes $\mathbf{x}_i$ and associated values $G_i$ to fit to $g(\cdot)$
$\mathbf{G}$	.....	$G_i$ expressed as a vector
$\mathbf{g}$	.....	$g(\cdot)$ evaluated at the nodes $\mathbf{x}_i$

$\mathbf{g}_i$  .....  $\mathbf{g}$  at  $i^{\text{th}}$  iteration  
 $\mathbf{r}'$  ..... optimal set of parameters, or solution  
 $\mathbf{r}_i$  .....  $\mathbf{r}$  at  $i^{\text{th}}$  iteration  
 $\delta \mathbf{r}_i$  ..... solution step at  $i^{\text{th}}$  iteration  
 $\mathbf{R}$  ..... vector of residuals  
 $\mathbf{R}_i$  .....  $\mathbf{R}$  at  $i^{\text{th}}$  iteration  
 $E_{rr}$  ..... a sum of squares;  $\mathbf{R}^T \mathbf{R}$   
 $\mathbf{J}$  ..... Jacobian matrix  
 $\mathbf{J}_i$  .....  $\mathbf{J}$  at  $i^{\text{th}}$  iteration  
 $\kappa$  ..... Levenberg-Marquardt damping factor  
 $\mathbf{v}_k^i$  ..... vectors to reparameterise the matrix  $\mathbf{T}^i$   
 $v_{jk}^i$  ..... vectors in indicial notation  
 $\theta_i$  ..... new variables to reparameterise the coefficients  $t_i$   
 $\mathbf{u}_k^i$  ..... vectors to reparameterise the matrix  $\mathbf{S}^i$   
 $u_{jk}^i$  ..... vectors in indicial notation  
 $V_{\alpha\beta}$  ..... limit on total (co-)variance between  $P^\alpha$  and  $P^\beta$   
 $i^{pen}$  ..... penalty function of type  $i$   
 $^{pen}$  ..... penalty function limiting the total variance of  $C_{\alpha\beta}(\mathbf{h})$   
 $f_{con}$  ..... factor to scale the penalty function  
 $\mathbf{R}^{obj}$  ..... residuals relating to the original objective function  
 $\mathbf{R}^{pen}$  ..... residuals relating to the penalty functions  
 $\langle \cdot \rangle$  ..... Macaulay function  
 $A, B, C$  ..... sets of nodes and associated values  
 $N_D$  .....  $N_D$  nearest nodes required in any estimation

$N_{max}$ ..... maximum set size  
 $P_{max}$ ..... maximum number of associated estimations  
**C**..... consolidated LHS covariance matrix  
**B**..... consolidated off-diagonal LHS submatrices  
constraining minimisation of kriging variance  
**c**..... concatenated RHS subvector of covariances  
**b**..... concatenated RHS subvector constraining minimisation  
of kriging variance  
**c\***..... concatenated RHS subvector of covariances filtered for  
the nugget effect  
 $C^0, C^1, \dots C^\infty$ ..... to describe function continuity or smoothness  
 $\eta_{\alpha\beta}$ ..... coregionalisation or *kriging* correlation between  $P^\alpha$  and  
 $P^\beta$   
*SOS*..... a pointwise variance metric on field data similarity  
*RHO*..... a pointwise correlation metric on field data similarity  
 $v(\mathbf{x})$ ..... local reduction in kriging variance upon cokriging

**Special Nomenclature used in Section I - 2 Numerical Uncertainty.**

$\psi$ ..... true solution to a differential equation  
 $\psi^*(g)$ ..... ordered numerical approximation to  $\psi$   
 $g$ ..... grid spacing (or a factor thereon)  
 $n$ ..... order of convergence of the numerical approximation  
 $a_n, a_{n+1}, \dots$ ..... coefficients in a Taylor Series expansion for error

**Special Nomenclature used in Section III - 1, Random Functions and Estimation; and III - 5, Structure Identification.**

$\Delta t$ ..... a one dimensional lag (separation) in time  
 $\{t_i, v_i\}$ ..... a series of velocities and time measurements

$\bar{v}$  .....the mean value of the velocities  $v_i$   
 $\sigma$ .....the standard deviation of the velocities  $v_i$

**Special Nomenclature used in Section III - 6, Multivariate Estimation –  
Cokriging.**

$p$ .....pressure  
 $\rho$ .....density  
 $v$ .....specific volume  
 $R$ .....specific ideal gas constant  
 $T$ .....temperature

**Special Nomenclature used in Section V - 1, Convection in a Square Cavity.**

$\mathbf{x}$ .....spatial coordinate system in  $\mathbb{R}^2$   
 $\mathbf{u}$  .....velocity  
 $p$ .....pressure  
 $T$ .....temperature  
 $\rho$ .....density  
 $\mu$  .....dynamic viscosity  
 $k$ .....thermal conductivity  
 $C_p$ .....specific heat capacity  
 $\beta$ .....coefficient of thermal expansion  
 $g$ .....gravity  
 $T_R$ .....reference temperature for Boussinesq approximation  
 $T_H$ .....hot wall temperature  
 $T_C$ .....cold wall temperature  
 $L$  .....side length of a square enclosure  
 $Pr$  .....Prandtl number

Ra .....	Rayleigh number
$\tilde{\mathbf{x}}$ .....	non-dimensionalised position
$\tilde{\mathbf{u}}$ .....	non-dimensionalised velocity
$\tilde{p}$ .....	non-dimensionalised pressure
$\tilde{T}$ .....	non-dimensionalised temperature
$\tilde{\nabla}$ .....	non-dimensionalised grad( $\cdot$ ) operator
$\tilde{\nabla}^2$ .....	non-dimensionalised div( $\cdot$ ) operator

**Special Nomenclature used in Sections V - 2, Generalised Covariance Theory;  
V - 3 The Poisson Equation; and V - 4 Afterword.**

$\hat{\mathbf{i}}, \hat{\mathbf{j}}$ .....	basis unit vectors in $\mathbb{R}^2$
$Z(\mathbf{x})$ .....	a numerical solution for temperature
$Y(\mathbf{x})$ .....	the source term in the numerical solution
$g$ .....	grid spacing in the numerical solution
$B_1, B_3$ .....	final coefficients in simplified generalised variogram model

**Special Nomenclature used in Chapter VI - Case Study 2: Inverted wing and rolling wheel.**

$X, Y, Z$ .....	global coordinates
Re .....	Reynolds number
$u_i^{TYPE}$ .....	$i^{\text{th}}$ nodal <i>TYPE</i> velocity measurement
$\bar{u}^{TYPE}$ .....	mean value of nodal <i>TYPE</i> velocity measurements
$\sigma^{TYPE}$ .....	standard deviation of nodal <i>TYPE</i> velocity measurements

# Chapter I - Introduction

This is a thesis about fluid dynamics and information, and their role in modern engineering. Historically, fluid dynamics has often been considered as something of a dark art by engineers. Whereas the understanding of structures and kinetics has long been based on established theories and physical mechanisms, the understanding of fluids behaviour is still putative. An introductory course in stress-analysis typically introduces a closed mathematical analysis of beam-bending, however the analogous example of viscous flow through a pipe must quickly be allayed by correction factors and a Moody chart. The empirical touch is due to the exceptional complexity of the underlying physical processes, which preclude analytic treatment.

The complexity of fluid physics has in the last twenty years, found something of a match in consumer electronics. The personal computer has reduced the price of processing power to a level where it is feasible for ordinary engineers to perform complex and “realistic” fluid simulations. This hardware capability has been matched by the development of proprietary and open-source software – complete, sophisticated and quite flexible packages for computational fluid dynamics, ubiquitously known as CFD. However these numerical models are still limited by the understanding of the theoretical physics, which is still in many respects incomplete.

The majority of CFD models are aimed at solving differential transport equations, usually the Navier-Stokes equations, using some spatiotemporal discretisation of the continuum mechanics model. If the discretisation scheme is well-posed, then an approximate solution is guaranteed. However, this reliability is something of a veneer – whether or not there always exist smooth solutions to the three dimensional Navier-Stokes equations, and whether these solutions even have bounded kinetic energy remain open questions in pure mathematics [1]. Even at the level of theoretical physics, a complete model for chaotic turbulent behaviour has remained elusive. This in particular is a source of frustration in numerical modelling, as one must use either a possibly unrealistic turbulence model, or endure the computational burden of explicit transient modelling. Further uncertainties are introduced by the inevitable existence of numerical errors, which persist and *will* persist in spite of the aforementioned improvements in processing power.

The abundance of modelling capability has been mirrored to some extent by the emergence of advanced “off-the-shelf” systems for flow-field metrology. These encompass optical systems such as Schlieren photography, Laser Doppler Anemometry (LDA) and Particle Image Velocimetry (PIV); medical imaging techniques such as Doppler sonography; and extend the use of more traditional flow measurement devices such as hot-wire anemometers, pitot tubes and the like. In particular the trend has been towards non-invasive flow metrology, especially LDA and PIV, which allow detailed spatial and temporal resolution of the vector flow-field. While more classical wind-tunnel testing focussed on just a few important variables such as lift or drag, the modern treatment results in large sets of spatial or configuration data to interpret [2]. Furthermore, it is reasonable to expect that as experimental capabilities improve and become increasingly automated, even larger datasets will be produced.

The direct consequence of both the modern analysis and experimental environments – is spatial data. By whichever means, large fields of results are produced that are at least nominally attached to spatial nodes. In detailed studies these nodes may be located not only by spatiotemporal coordinates, but also by configuration variables such as angle-of-attack or Reynolds number. For the purposes of comparison – and critically for engineering applications, to build

confidence in a model or a set of measurements – these results are typically produced in parallel across a number of different platforms or models. For example measurements relating to say, operational performance may be reproduced in a wind-tunnel at a similar dimensional regime, and then simulated using a number of numerical models. The common elements to each model are the non-dimensional operating conditions and the domain – or rather, elements of its geometry. A crucial aspect to the mature understanding of many fluid dynamics scenarios is the consideration of these disparate datasets. Whilst they are hardly ever in perfect agreement, one hopes that collectively they indicate some overarching behaviour. Certainly it would be remiss to accept the predictions of any one platform without comparing it with others, regardless of how sure one might be of its veracity.

The aim of this work has been to develop a synthetic method or framework that ties together the spatial data which are produced in the course of a modern aero- or hydrodynamic investigation. The flavour of this tool has been skewed towards engineering applications where data is frequently messy or incomplete, and it cannot be guaranteed that procedures or regimens relating to the initial data generation will be followed. Synthesis is a post-processing step, and the less which needs to be assumed about the data on which it works, the more useful and general it becomes. In this way, the engineer is aided in seeing the bigger picture that the data reflects.

## I - 1. State of the Art

To achieve the above aims, the similarity of the spatial datasets must be assessed, which means also that the possible reasons for their *dissimilarity* must be addressed. This introduces elements of what has traditionally been called verification and validation [3]. These terms are commonly used in numerical modelling to describe how accurately the numerical model represents the theoretical model – verification, and how accurately the theoretical model represents the physical reality – validation. It is worthwhile to note that these ideas have grown from more numerical than experimental origins. In classical verification and validation, it is tacitly assumed that the goal of engineering simulation is to approximate the physical reality as best as possible, and common techniques therein all serve this aim. This is

reasonable, but sometimes threatens to overlook the complexities of both the physics and the model. As previously mentioned, there is still significant theoretical uncertainty as to what the approximating equations are, and apparently deterministic numerical models can still exhibit chaotic behaviours at scales larger than the grid, or a single timestep [4].

Typical approaches in verification and validation serve to quantify or bound errors in numerical solutions, in order to build confidence in the numerical model. Whilst these techniques are grouped together, they often bear little relation to each other apart from their common goal. Because all such exercises are directed towards improving the model, it is posited that there is an absolute “truth” with respect to which the aim of all models or experiments is to achieve parity. The present author views this as a scientific approach to controlling and quantifying error, where the issues to be resolved are ones of truth and accuracy. However, while engineers might aim for these ideals, frequently the constraints of the larger project take precedence: these methods work well in situations where the amount of uncertainty or disparity is relatively low, but may not be constructive or realistic away from this ideal. Furthermore, classical verification and validation is classically informed in that it is not forcibly designed for spatial data: often, specific model outputs are concentrated upon – for example lift or drag, and the added value of obtaining complete flow-field information is lost.

A frequent source of irritation is the comparison of non-statistical and statistical uncertainties [5]. The errors in a numerical solution are largely due to the propagation of local approximation errors, whereas experimental errors are statistical in nature. One may handle the numerical errors in the same way as bias errors are treated in experimental results, but it is often difficult to bound the error and so create a useful confidence interval – especially on the raw flow-field results. It is nonsensical in any case to express numerical error in terms of a confidence interval because this error is described theoretically as a deterministic, albeit unknown, higher-order function. Put simply, there can be no replications of the numerical experiment, as repeating it can only return the same results. Thus, if one constructs a tolerance from the numerical result, within in which one imagines there is a 95% chance that the correct solution may be found, one is ever making a qualitative

assertion. Adopting consistent practices to choose this interval according to some error estimate merely codifies this assertion. This has caused a great deal of angst amongst practitioners, and such unproductive debates as whether error estimates ought to be factored by two, or three [5]. The present author maintains that the numerical solutions are the results of a deterministic process and so do not comprise randomness in any *aleatoric* sense. However, it is still useful to model the *epistemic* uncertainty – that is, the unknowable error committed at each timestep or grid-spacing, and bona-fide chaotic variation in iterative or real time, as spatial random processes.

Verification and validation, such as they are represented in the CFD community involve the systematic breakdown and where possible, quantification of error. There are however other, less obvious models for uncertainty that have emerged in other specialisations. A very close cousin to CFD for aerodynamics, or perhaps rather grandfather, is numerical weather prediction. In developing short to medium term meteorological forecasts, it is necessary to integrate large numbers of spatial observations and measurements with a numerical model. This activity is generically termed data assimilation [6], and includes the so-called 3DVAR and 4DVAR algorithms which model the uncertainty in the model and measurements. The techniques used are forcibly tailored towards dynamic updating of the error model as new information comes to hand, but nonetheless there are many similarities with the author's present concerns. The field measurements from satellites, weather balloons and stations all exhibit varying degrees and types of statistical uncertainty and/or bias. Meanwhile, the state of the system is also informed by the last forecast, polluted as it may be by numerical error and modelling inaccuracies.

Another specialism in which spatial data of differing origins and types must be assimilated is in the earth sciences. *Geostatistics* is the modelling of spatial uncertainty and its role in estimation, as originally applied to surveying, cartographic and reserve estimation problems [7]. Typical of these studies, is the generation of data at points, perhaps by boreholes or a topographic survey, from which the full extent of the reserve or geological formation must be inferred. Like spatial data in flow measurement, this can be further complicated by extra information – additional surveys, or in the case of an underground reserve, information from mining records

indicating how much ore was actually found there. Critically, the effects of spatial averaging on estimation variability are modelled. Often, all of the data that can be gathered at a site is accrued and cross-correlated in what is termed a Geological Information System (GIS), so that better predictions and explorations can be made.

## I - 2. Numerical Uncertainty

The majority of commercial CFD programs – for example, Fluent or CFX – attempt to solve partial differential equations describing the microscopic flow physics using the Finite Volume (FV) method. The FV method is a relatively simple discretisation scheme. It can be thought of as a flux conservative Finite Difference (FD) method that generalises orthogonal grids to unstructured arrangements of finite volumes or *cells*. The partial differential relations are approximated at nodes in these cells and in the limit, the numerical solution approaches the true solution as more cells are used.

Importantly, no functional form is adopted over FV cells and there is a strong physical interpretation to the nodal values on them [8]. This is unlike many other numerical methods of partial differential equation solution, such as spectral methods, Boundary Element (BE) methods and notably Finite Element (FE) methods. This is noteworthy, if only for the reason that many practitioners consider that FE analysis – universally used in solid mechanics, is the same thing as FV analysis. Whilst cosmetically there are resemblances, mathematically there are important differences [9]. The FV method has come to the fore in fluid mechanics largely as it is easier to implement and program than the FE method, and subsequently allows greater flexibility addressing the more abstruse transport equations. However more fundamentally, a sensible and meaningful functional form for the solution of the Navier-Stokes equations is simply not obvious. For the equations of stress, it is relatively clear what sort of behaviours are required of such a form – before even considering the differential relationships, there exists some knowledge of behaviours such as simple tension, compression, shearing and bending. The advantage of proposing meaningful functional forms to interpolate the nodal solution, is that it

allows the analyst to work in the *coarse*. A useful structural FE analysis can be constructed with just one element, which would be a nonsense using the FD method.

The verification therefore, of FD and FV solutions is particularly important as the solution is *not* necessarily reasonable away from the asymptotic range of the method. In other words, where  $\psi^*(g)$  is a  $g^n$  ordered approximation for the solution  $\psi$ , there is a Taylor series expansion for the discretisation error;

$$\psi^*(g) = a_n g^n + a_{n+1} g^{n+1} + a_{n+2} g^{n+2} + \dots$$

and  $g$  must be small enough to make the leading term dominant. Unfortunately, because one cannot know all of the coefficients  $a_n, a_{n+1}, \dots$  it is very difficult to verify that this is indeed the case. Practically, Richardson's extrapolation [10] can be used with two or more solutions at different resolutions  $g$  to estimate the magnitude of the leading term or terms, assuming that the order  $n$  of the method is known. However one must still make the assumption that the higher order terms can be neglected, and even the order of the method  $n$ , should be verified against the observed order of convergence – usually requiring more than two solutions at some integer refinement [11]. Typically, different parts of the flow-field solution will converge at different rates, even though a similar discretisation scheme is used everywhere. Functional values such as lift or drag, which are integrated over some or all of the solution, may converge at different rates or before point velocities in the flow [11]. Although important structural elements in the flow-field may be unresolved, the integrated functional values may still appear convergent [12]. Certainly one might be doubtful of the robustness of such a solution, as the physics of the flow is not really represented.

In spite of the above difficulties, the greatest source of disparity between experimental and numerical results often remains in the physical model itself. Micro-mechanical properties such as viscosity and density are extrapolated to a scale at which their behaviour can be quantified, measured and compared with the real world. However, how this gross behaviour emerges from the smaller scales is very complex and still under theoretical examination: fluids exhibit totally different gross spatiotemporal behaviours as the non-dimensional regime; for instance Reynolds number, changes [13]. In particular, turbulent behaviour properly requires a time-dependent model such as Direct Numerical Simulation (DNS) or Large Eddy

Simulation (LES), which are significantly more numerically expensive than steady-state approaches. Frustratingly, engineers are most often interested in steady-state operating conditions, which leads to the widespread use of turbulence models and the Reynolds Averaged Navier Stokes equations (RANS), as a middle road [8]. Whilst convenient and dimensionally justified, these models necessarily simplify the actual turbulent time-dependent fluctuation – parity with “reality” may be achieved on average, but probably not in the minutiae of the flow-field. In general, turbulence models work best for the specific circumstances they are designed to address, but may be unreliable outside of those, or where turbulent fluctuations are large.

### I - 3. Original Contribution

As a consequence of the modern analysis and experimental environment, it is now possible to scrutinise the flow-field and check the macroscopic behaviour of microscopic assumptions made in the numerical modelling. In the past, the technical segregation of these activities has largely left the detailed comparison of numerical and experimental results to theoretical physicists. But this is changing, particularly as engineers seek to use numerical models to drive down development times and costs [14]. The only way to do this effectively in the midst of uncertainty is to consider all of the data synergistically and develop useful measures of similarity. Whilst such measures are relatively obvious when performance is judged on only a few criteria (lift, drag, etc.), there has been little consideration given to the comparison of *fields* of data that are intrinsically inter-related.

A metric describing the spatial similarity between experimental and numerical nodal data is proposed in Chapter VI. This metric is derived from spatial statistics commonly used in geostatistics and is used to compare the numerical results of different turbulence models with respect to a set of experimental results. Use is also made of the statistical model to inform interpolation between nodes, and more importantly, meaningful blending of the numerical and experimental datasets. In this process, spatial statistics need to be generated on numerical solutions of the Navier-Stokes equations. The implications of such statistical interpretation of such non-random numerical data are therefore examined in Chapter V. It is found that local

deterministic errors due to the truncation of the Taylor series expansion necessary in the PDE solution may be idealised as the result of a stationary random process, at least at higher orders, for non-trivial solutions.

Therefore, tools are proposed and demonstrated for the comparison of field data that are typically generated in the course of an aero- or hydrodynamic study. Specifically, the comparison of spatially related data is addressed, as this detail is typical of modern fluid dynamics. The algorithms used are established spatial estimation techniques in the field of geostatistics, but are not current for the assessment of spatial data in fluid mechanics. The tool is regarded as a synthetic method to help develop a broader appreciation of how disparate datasets may be related, where little can be presupposed about the data and their means of collection – the analysis is made *a posteriori*.

What is apparent in the foregoing discussion is that regarding modelling or experimentation as a narrowly-defined search for ‘truth’ is possibly counter-productive – after the statistician George Box; “all models are wrong, but some are useful” [15] (p. 424). The usefulness of numerical modelling and experimentation, is that they inform a greater picture and identify behaviours.

In the course of this work the following publications have been produced:

Kelly, D.W. and Watson I.T., *Towards a Real-time Interactive Environment for Finite Elements* Symposium – (Prof. Valliappan’s retirement from UNSW), Sydney, NSW, Australia, 21 February 2004.

Watson, I.T., Barber, T.J. and Leonardi, E., *Validation of Numerical and Experimental Results Using the Kriging Estimator*, in 25<sup>th</sup> AIAA Aerodynamic Measurement Technology and Ground Testing Conference. 2006, AIAA, San Francisco, California (AIAA-2006-3301).

Watson, I.T., Barber, T.J. and Leonardi, E., *Discovering Interpolation Functions in Experimental Fluids Data*. Australian Workshop on Fluid Mechanics 2007, Melbourne University, Melbourne.

Watson, I.T., Barber, T.J. and Leonardi, E., *Whole Field Validation of Numerical and Experimental Results*. Under review, Computers and Fluids, submitted Oct 2008, accepted pending revisions.

Barber, T.J., Doig, G., Beves, C., Watson, I.T., and Diasinos, S., *Running on all cylinders – the integration of CFD and EFD for ground-effect aerodynamics studies* in 4<sup>th</sup> Symposium for Integrating CFD and Experiments in Aerodynamics. 14-16 September, 2009, von Karman Institute, Rhode-Saint-Genese, Belgium.

## I - 4. Structure of the Dissertation

During the candidature, the established algorithms of geostatistics were implemented in such a way as to make them accessible to the specialism of fluid mechanics, and useful for the particular problems that spatial datasets in fluid mechanics present. This took the form of a prototypical program – program `krige`, that performs basic spatial structure identification and kriging estimation. This program was applied to data assimilation scenarios that were supplied by colleagues in the School of Mechanical and Manufacturing Engineering, with an emphasis on practicality and meeting the demands of those with a background in engineering fluid mechanics.

**Chapter II** presents a broad review of the diverse techniques often found or recommended in the Verification and Validation literature. Whilst many of the recommendations to be found in this field are sensible, as has been remarked there is very little commonality between procedures, and very few techniques are universally applicable. Data assimilation methods are also covered here, from the general; response surfaces, interpolation and curve-fitting, to specific techniques in the geological sciences and meteorology. At the start of the candidature, it was hoped to answer two basic questions: how might one compare spatial datasets, and how then might this inform a best estimate of the reality of the situation. The literature survey charts how the investigation was led towards specific tools to answer very general questions. This leads to a short afterword that explains the author's conception of the problem in more concrete terms, and why geostatistical methods were pursued.

**Chapter III** describes the basic estimation theory that supports the geostatistical tools whose use is proposed to address the data assimilation problem in ‘classical’ aerodynamics. If the reader is familiar with these techniques, a detailed reading of this chapter may not be necessary. The univariate, stationary random function model is introduced with the kriging estimator, and a simple one-dimensional time-series is used to demonstrate the effect of modelling assumptions on estimation. The concepts are extended to the multivariate case, cokriging.

**Chapter IV** details how the methods of structure identification and estimation outlined in Chapter III have been implemented in program `krige`. It was decided to program an in-house kriging algorithm to allow greater flexibility in its range of applicability and also during its development. Whilst there exist open libraries of geostatistical routines, many of them are specifically aimed at geological problems. These tend to involve smaller numbers of points (sometimes by orders of ten), located exclusively in two or three dimensions. In particular, the restriction to spatial independent variables precludes the generation of multidimensional response surfaces, useful in many engineering applications. Furthermore, the algorithms in Chapter IV have been developed with the intent of automating aspects of geostatistical practice as much as is possible; simply as it cannot be expected that fluids specialists should adopt techniques that are apparently outside of their area of expertise or interest.

**Chapter V** presents a case-study and investigation, which examines the treatment of deterministic computed results by statistical methods, and the resulting implications for the principles of spatial stationarity. Clearly, treating the replicable, deterministic output of a computer program as a random function seems unreasonable – in spite of its usefulness for the purposes of quantifying spatial correlation. This chapter proposes an interpretation of such a model wherein the random function characterises the spatial variation, hence smoothness, of the numerical solution. This solution is not in any case free from uncertainty, although the uncertainty here is epistemic, not aleatoric. These ideas are explored in a series of progressively simpler numerical experiments, as simplicity implies determinism in this context. The theory of higher order *intrinsic* random functions is proposed to extend the usefulness of the stationary model.

**Chapter VI** forms a case-study in which program `krige` is finally used to develop correlations and estimations relating to an experimental and numerical aerodynamic study of an open-wheel configuration often encountered in motorsports. A basic measure of spatial similarity is proposed, and the usefulness of cokriging to develop better estimates of field behaviour is examined. Variations to the estimation methods are also presented and discussed.

**Chapter VII** concludes the dissertation. Areas for future work and improvement are identified and the strengths and weaknesses of the demonstrated spatial techniques are appraised.

# Chapter II - Literature Review

The difficulties mentioned in the previous chapter have been partially addressed from many angles by a multitude of techniques. Whilst there are a great many approaches to verification and validation problems, there is generally very little commonality between them: they have been developed in different fields for different ends and are constrained by different working environments. What results in the literature are numerous guidelines and recommendations [3, 11, 16] that define semantics and methodology, accompanied by a host of supporting tools and techniques – few of which bear any direct relation to each other. This is apparent in the basic literature dealing with broader verification and validation methodologies which is discussed below, along with the principal techniques used in verification and validation.

## II - 1. Verification Activities

The term verification has been coined in the last ten or twenty years to express the variety of procedures used to assess the accuracy of a given numerical model. These are diverse and depend heavily upon the numerical methods employed to solve the conceptual model – a brief exposition of the principal approaches which

have been proposed and used, shall be attempted. The American Institute of Astronautics and Aeronautics (AIAA) defines verification as:

“ **Verification:** The process of determining that a model implementation accurately represents the developer’s conceptual description of the model and the solution to the model. ” [3]

Other organisations, notably the Institute of Electrical and Electronics Engineers [17], use the term verification (and later, the term validation) in a rather different context – relating specifically to software development. The AIAA’s definitions, essentially similar to the definitions adopted by Roache [11, 16], are adopted in this work.

Verification in CFD typically assesses the accuracy of a numerical solution to a set of possibly time-dependent partial differential equations (PDEs). In the main, the fluid dynamics community has pursued numerical solutions to PDEs by employing variations on differencing methods. These have evolved into the Finite Volume (FV) method which is generally implemented in most mainstream CFD packages such as Fluent and CFX. Like Finite Differences (FD), the PDE solution is approximated at points only, but FV benefit from the consideration of an integral form of the PDE over collocated sub-domains (cells or elements) [8, 9, 16] – thus permitting a flux conservative formulation, and greater topological flexibility for practical problems. Both FD and FV methods are constructed around a truncated (and generally multidimensional) Taylor series that locally approximates the actual PDE solution [18]. As a result, error estimation is usually achieved by bounding the value of the higher order terms, as first articulated by Richardson [10]. These terms tend asymptotically to zero as grid size approaches zero and for a meaningful error estimate, the grids must be at least within this asymptotic range of convergence – that is, the leading term in the error expansion is assumed to be dominant. Simply attaining this level of discretisation for many real problems can prove difficult, especially as Richardson’s extrapolation originally proposes a halving of the mesh spacing. More practical are methods including those proposed by Roache [11] and others [19, 20], who extend the same principle without this requirement, called non-

integer refinement. Roache's Grid Convergence Index (GCI) [21] has emerged as a popular, simple and flexible tool to study the convergence of numerical schemes.

Strictly speaking, Richardson's extrapolation (or its variants) is applied directly to the solved quantities to obtain a correction or error bound to say, the velocity or pressure field [10]. However, the technique is also widely applied to functionals such as lift, drag or heat transfer coefficients – values integrated from the raw simulated quantities. It is noted that not all integrated or solved quantities converge at the same rate [11] – indeed, in the presence of strong singularities critical solved quantities may not converge at all under successive refinement [3]. The great advantage of Richardson's extrapolation is its ready application to various solution methods outside of FD and FV – any scheme involving a discretisation that can be successively refined is a candidate, or indeed any convergent sequence.

Finite Volumes are by no means an exclusive method of solution for PDEs describing fluid dynamics [8, 22]. Historically they have found favour in the field of Computational Fluids Dynamics largely because they may be adapted to treat a very broad class of problems; particularly those articulating complex and *realistic* physical equations. Furthermore, the physics of said equations is also less well understood and so less may be assumed about the solution – there is of course a “millennium prize” offered by the Clay Mathematics Institute for merely proving the *existence* of a smooth solution to the Navier-Stokes equations [1]. Nevertheless, the FV method is significantly less complex than some of its counterparts in the Finite Element (FE), spectral and Boundary Element (BE) methods, and this has made it easier to program and implement – perhaps at the expense of robustness and stability [9]. As one moves towards more elaborate methods, the mathematical and programming complexities become increasingly pronounced. This typically means that the physical models are restricted – for example the very first practical simulations of inviscid and potential flow solved Laplace's ‘heat’ equation using BE [23] or panel methods. Various FE formulations [22] and spectral representations [24] of the Navier-Stokes equations have also been attempted. The provision of a variational principle in each case provides an error norm on the solution which can theoretically be determined using various adjoint solution methods [17-19]. Bounding a norm on the error function is achieved in a post-processing solution step

in these methods, which can be almost as numerically intensive as the problem itself [25]. Techniques such as the Zhu-Zinkiewicz error indicator [18-21] provide a middle road by considering some terms to be small. In general, a more accurate error bound is necessarily more numerically expensive and often inclement on the mathematical tractability of the specific problem.

There exist alternative error indicators that are more independent of the solution method. A significant proportion of these make use of the physical aspects of the solution to offer some indication of feasibility. Sometimes the conservation of higher moments such as energy, angular momentum or force balance are used [18-20]. Of course, this is only applicable if the solution scheme does not expressly conserve these quantities – the Finite Volume scheme is meant to be flux (momentum, pressure, etc.) conservative. Alternatively one can examine the preservation of laws that theoretically must be observed, but are not explicitly written into the equation solution – for example the second law of thermodynamics [26].

In general, most CFD simulations are of non-linear phenomena, and sometimes spectacularly so. Computers are of course fundamentally restricted to linear operations, and so solution schemes for such PDEs are almost always iterative at some level. Historically, finite difference solutions of the Poisson equation were effected by iterative hand calculations [18], even though Poisson's equation is a linear PDE. More recently, perhaps by virtue of the fact that CFD has 'boomed' in a period of cheap RAM, solvers have also been designed to operate iteratively in core memory, even for linear matrix problems. Typically, such matrix solution schemes are cast as iterative residual minimisation problems [16] and will not solve the given equations exactly or even to machine precision. This is in contrast to FE solvers which evolved for structural analyses in the 1960s and 1970s, when RAM was exceptionally expensive, and permanent storage was relatively cheap. These typically use variants of Gaussian elimination and operate out-of-core for large problems using skyline, frontal and sparse solvers [9] – outright solver performance is sacrificed to optimise input/output overheads. Therefore, for most CFD solvers there is the added uncertainty of iterative convergence.

Typically, most solvers will be stopped when a given norm on equation residuals falls below a user defined value – thus managing the error due to

incomplete iterative convergence is relatively simple. The norm is simply set several orders of magnitude smaller than the right hand side of the matrix expression [27]. Note however that this does not precisely bound the error in the solution vector, which is usually dependent on the matrix condition number [28]. In practice, the solution is performed at several levels of residual norm to guarantee the solution does not change significantly [3]. The iterative solution often mimics the fluid dynamics of the physical problem – both are non-linear, dynamic, multi-dimensional systems. The discretised version of this problem theoretically may exhibit some very exotic behaviours such as oscillatory and chaotic orbits, in addition to outright convergent and divergent paths in the phase space [29]. Difficulty achieving convergence can sometimes be attributed to these behaviours, especially when solving in flow regimes that are inherently chaotic or oscillatory [4]. Hopefully though, at least stability is guaranteed by a von Neumann analysis [30].

A particularity of especially fluids modelling is the general inscrutability of behaviours that are observed in quite ordinary (Newtonian) fluids. It is not hard to achieve physical flows that display an incredible amount of physical complexity, in spite of the apparent homogeneity of the physical laws and materials. Turbulence is an excellent example of this [29]. At the macro-scale the behaviour appears to be random about a steady average, yet the micro-modelling of vortices and transition layers reveals highly bifurcatory and time-dependent ordered behaviour. The understanding of such non-linear dynamic systems – so-called ‘chaos theory’ [31] – has been an area of intense interest in applied mathematics in the last half century. The modern field is popularly attributed to Lorenz [4] but the basic ideas date back to Poincaré and Kolmogorov, among others. The theory essentially categorises the dynamic behaviour of systems by how they evolve in time. In addition to familiar convergent, divergent and oscillatory behaviours (called *orbits*) chaos theory assigns another category, the *strange attractor* – discrete examples in ‘pop’ literature include the famous Mandelbrot set and the Lorenz attractor; turbulence is an obvious continuous example. The importance of such structures is that they characterise an invariant sub-set of the phase space of a dynamic system, into which time dependent trajectories may fall and subsequently remain in a stable, though chaotic, orbit. These orbits appear stable (quasi steady-state or ergodic), but never exactly repeat so one cannot know that they are stable. The importance of this to a numerical model, is that

an ordered time-marching, or false time-marching scheme can conceivably result in outwardly random behaviour.

Finally, it is noted that often problems of code confidence are grouped into verification activities – following the recommendations and definitions of Roache [11]. By this interpretation the analyst also takes responsibility for the proper working of the code itself – verifying it by means of testing, benchmarking and notably the method of manufactured solutions. The last technique involves testing of the code against analytic solutions of the approximating PDEs. Where these equations are too complicated to permit any sort of analytic solution, such a solution is ‘manufactured’ through the addition of appropriate source terms. Whilst this certainly tests the core functionality of the code, it does not ensure that it will work on actual problems for which functional variation may not be so contrived. In the context of CFD for commercial application and the pervasive use of CFD software packages, most researchers consider this level of verification too parsimonious or difficult [32], and outside of the usual scope of verification. Certainly though, studies of software quality assurance do highlight the usefulness of scepticism in ‘black box’ simulations [33].

Some approaches outlined above might seem to give the impression that the quantification of numerical error is almost straightforward or procedural. In reality, it is frequently problematic. In particular, achieving ‘grid independence’ can require very fine meshes and significant computational effort – certainly for many practical geometries and problems it is impossible with current processing power. Engineers typically assay the most complex model that the hardware can handle, and so this difficulty is likely to persist in spite of continuing improvements in computing power. From experience, the time taken for analyses fits into human timeframes, regardless of the hardware’s capability. To circumvent the difficulties in providing realistic error estimates for all analyses, the AIAA consider the activity of verification to be dependent on the objectives of the analysis and scope of the overall study [3].

## II - 2. Validation Activities

Whereas verification concerns the correct solution of the approximating PDEs, validation ensures that the appropriate equations are solved. Naturally, this is with the aim of inferring physical behaviour from the modelled behaviour, thus the scope of the desired inferences also drives the selection of the mathematical model. Validation has thus far been performed almost on an ad hoc basis, with various approaches considered depending on the degree of agreement or strength of inference required. The AIAA defines validation as:

“ **Validation:** The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model. ” [3]

This definition and the conventions adopted by the AIAA in this area are again similar to the definitions adopted by Roache [11, 16].

Perhaps the most widespread means of validation is via the comparison of just those aspects of the flow that are considered relevant to the performance of the numerical model. Often the replication of integrated functional values such as lift or drag is considered to demonstrate model validation [34-39]. It is natural then to consider the statistical dispersion of the physical data to quantify the performance of the numerical model [11, 34]. This approach is a natural extension of ideas that have been current in experimental data analysis for some time [2]. It is however, not a complete model as it confers a confidence in the numerical result that may not be warranted. Attempts to address this shortcoming assign some uncertainty to the numerical result. How, and even whether, this ought to be done is controversial.

Wilson and Stern [36, 38, 39] use arguments native to Experimental Fluid Dynamics (EFD) [2, 40, 41], and extend them to include errors in the simulation, which are represented as a quasi variance quantity. This is of course purely notional, as usually there are no differing repetitions of the numerical results from which to calculate a variance – a bracket is chosen such that it is 95% certain that it contains

the correct numerical result. This is achieved by ‘correcting’ (properly, factoring) the adjustment offered by a Richardson’s extrapolation with a factor gained loosely from experience. Indeed, as Wilson and Stern’s factor [36] is based on a particular solution of the one-dimensional wave equation, it is questionable whether it is applicable to differing solution methods and orders, domain dimensionalities and problem types. This weakness is roundly criticised by Roache [5], whose proposal for a general purpose grid convergence index [21] their numerical uncertainty estimate somewhat resembles. In Wilson and Stern’s work, it is also made clear that the transition from the comparison of scalar point values to fields of values is to be achieved on a point-wise basis – that is, treating each point in the field as a particular instance of the proposed framework.

Other attempts to introduce a statistical element to the computed results include  $N$ -version techniques [39, 42]. In this, replications of the computational “experiment” are achieved by solving ostensibly the same problem with different codes, and perhaps even with different constituent equations (turbulence models and the like). However, this approach requires the existence of multiple CFD codes – again, it is difficult in any case to draw statistical inference from the computed results.

Where more advanced experimental techniques permit the spatial resolution of the flow-field, the comparison can be purely qualitative, or centre around a given flow feature such as a stagnation, separation or re-attachment point. Whilst, some less arbitrary schemes have been proposed, most comparisons are still made on a point-wise basis. Oberkampf and Trucano [34] suggest a point-wise comparison over a one-dimensional domain, defining a metric for similarity and incorporating it and experimental statistical uncertainty in a so-called validation metric. Whilst sensible, there are still arbitrary choices that must be made in this model – the method is obviously complicated by the possibility of interpolation errors. These are inevitable whenever computational and experimental nodes do not coincide in space. Further, there is a specified mechanism for uncertainty which differs fundamentally for the numerical data and the experimental data. A marked point of difference with Wilson and Stern [32, 39] is that the numerical errors are not to be interpreted in a statistical manner. Oberkampf and Trucano aim to meet the following criteria, summarised as:

“(1) We agree... that the metric should incorporate an estimate of the numerical error in the computational simulation. However, we do not agree that this estimate should be grouped with the experimental uncertainty nor that it should be represented probabilistically. ...

(2) The metric should not exclude any modeling assumptions or approximations used in the computation of the simulation result. ...

(3) ...the metric should incorporate an estimate of the random errors in the experimental data, e.g., an estimate of the variance... the metric should also include an estimate of the correlated bias errors in the experimental data...

(4) The metric should depend on the number of experimental replications of a given measurement quantity...

(5) ...the metric should use nondeterministic methods to propagate uncertainty through the computational model... .” [34]

These criteria, although rigorous, are very difficult to satisfy in practice – especially as the fifth requirement presumably necessitates a Monte-Carlo simulation on the numerical solution scheme.

Wilson and Stern’s [36] approach instead defines a level or bracket of validation that may be obtained in spite of inaccuracies in the numerical and experimental results. It is somewhat more flexible than the above criteria, as has been commented on in discussion articles [5, 32, 43]. Moreover, there exist many similarities between the two approaches. Although it is not explicit, in both it is assumed that error enters numerical and experimental results via known mechanisms and what the analyst or experimentalist desires is the “*Truth*”. Whilst Oberkampf and Trucano [34] tacitly consider the numerical solution verified and compare directly to that, they still essentially choose an arbitrarily constant standard ‘truth’ with which the results are compared. The standard is in some sense monolithic – whether it actually is the truth, or a corrected simulation, and the mechanisms for error are all prescribed. Wilson and Stern combine their error terms as a sum-of-squares treated as quasi-variances [39] in the manner of EFD, whereas Oberkampf and Trucano [34]

use the triangle inequality (sum of absolute values) and normalise via the  $\tanh(\cdot)$  function. Furthermore, in each method the error sources are considered to be independent – a safe, if unverifiable assumption.

A shortcoming of point-wise comparison is that it does not appreciate the possible distortion of otherwise useful results. This is most obviously manifested as spatial (or phase) shifting between results sets. So far, very little has been mentioned with regard to how this might be sensibly handled. Carrica et al [44] use a Fast Fourier Transform on their data to effect a functional description with which they can apply the methods of Wilson and Stern [39]. Notably they examine the amplitudes only of the lowest two frequency components – imaginably because these are the principal contributors in the spectral representation.

## II - 3. Sensitivity Analysis

To the basic V&V frameworks outlined above, some important annexes must also be added. The appropriateness of modelling assumptions such as boundary conditions and material properties must also be addressed. Very often, small perturbations to these may be of great consequence to the physics of the computed solution [11]. What is generally called upon to quantify the effect of these assumptions is generically termed *sensitivity analysis*. Indeed, many aspects of ‘verification’ can be recast as sensitivity problems – for example, sensitivity to meshing, time-step size or residual limits. The approach in all such cases is to characterise the model behaviour with respect to the ‘sensitive’ parameters or design variables – the domain of which is called the *design space*. Pelletier et al [45] examine computational uncertainty in terms of sensitivity analyses. Sensitivity analysis of the Navier-Stokes and related equations is a formidable specialisation in its own right, which is why it is not considered to be within the direct scope of this work. However, it is of extreme usefulness for characterising the broader behaviour of simulations as it can offer valuable derivative information on the behaviour of any given response surface. The state-of-the-art is outlined below, with attention to derivative estimation techniques in use for aerodynamic optimisation.

Sensitivity analyses come in many different guises. Principally, one may distinguish response surface approaches; where the system response is evaluated at points in the design space and sensitivities are inferred, from gradient-based approaches; where some attempt is made to calculate the partial derivatives at a point in the design space. Optimisation algorithms such as genetic and evolutionary algorithms fall into the former category, as do the statistical interpretations of sensitivity [46, 47]. Statistical appraisals of sensitivity connect the variances of model outputs to model inputs. Response surface approaches are also encompassed by Design of Experiment methodologies which again attempt to relate the sensitivity of the output to that of the input [48].

Gradient based methods have a much more numerical flavour, especially as they form the basis for a great many numerical shape optimisation schemes in aerofoil and hull design. In this area, the complexity of the governing differential equations often determines what may be achieved – there is a trade-off between the improvements provided by an efficient optimisation algorithm, and choosing a realistic model with which to simulate those improvements. For heat flux problems, it is relatively simple to develop the partial differentiations manually and incorporate them into a code [49, 50] – a so-called continuous method for sensitivity. Similar approaches have been taken for the Euler equations [51]. When the sensitivity of relatively few model outputs is desired (in comparison to the number of inputs) so-called adjoint methods [52] of gradient determination become more efficient than direct methods. This is demonstrated [53, 54] again in relation to optimisation algorithms requiring the sensitivity of the Euler equations, and Kim et al [55] perform a sensitivity analysis on the Reynolds Averaged Navier-Stokes equations with various two equation turbulence models. Adjoint methods are typically useful for aerofoil optimisation, where there are many shape parameters as inputs and perhaps only lift and drag coefficients as outputs.

The above authors have adopted quasi-analytical approaches to calculate the partial derivatives required in a sensitivity analysis – they have coded the analytic differentiation steps. This is a very time consuming activity, especially as one utilises more complex and realistic mathematical descriptions. So-called discrete methods, on the other hand, use external differencing techniques around the numerical solver

to obtain the partial differentiations required to solve both the direct and adjoint gradient formulations. This involves propagating small perturbations “epsilon” to the design variables through the solver, and saves man-hours at the expense of computational efficiency. Whilst this is an obvious and flexible way to obtain gradient information, it is often difficult to balance the mathematical accuracy gained using a smaller step size, with the competing limitations of numerical precision. A more robust formulation is the so-called complex derivative [56-58], which propagates an imaginary epsilon in the dependent variables through the solution. From this, a second order approximation of the first derivative can be made from the imaginary part of the solver output, without massive cancellation errors. Whilst not strictly a discrete method, Automatic Differentiation [59, 60] achieves largely the same thing by propagating derivatives with sensitive variables as the solution is effected, in an automatic fashion. In all of the cases mentioned above, the changes to the code are generally cosmetic – necessitating the definition of new types and/or operators, but the ease of calculation is gained at the expense of numerical efficiency.

Traditional differencing is the only method listed above that operates as a true algorithm extension, without modification to code and internal algorithms. For this reason, it remains pre-eminent in practical sensitivity analyses, particularly when assessing variable parameters such as the suitability of far-field conditions, boundary conditions and material properties. In any case, for many experimental and field measurements, gradient measurement is simply impossible. In chapter three of his doctoral thesis Michalek [61] incorporates numerical gradient information to guide the choice of sensitive parameters in experimentation – however, the uncertainty of the numerical results and model is not considered. The usefulness of propagating input uncertainties through a numerical model in the context of a validation experiment has also been pointed out [11, 34]. This approach is hindered however, by the numerical expense of conducting statistically meaningful Monte Carlo simulations – which leads one back to the use of response surface methods and constructing surrogate models from which gradients and uncertainties can be inferred. In the vast majority of cross-disciplinary studies involving CFD and detailed experimental surveys, the data relates to points in either a physical, or a design space. Thus in some way, one generally returns to the use of a response surface to characterise behaviour.

## II - 4. Response Surfaces and Surrogate Models

The re-construction of response surfaces from point information is a common and deceptively complex problem in many areas of engineering science and technology. Response surfaces are of particular interest to engineers, as they characterise system behaviour and trends, and map the relation between independent and dependent variables. The simplest response surfaces are the common linear regressions that are typical of classical frequentist statistics [62-64]. Global basis functions over the independent data are proposed and then weighted so as to produce a least squares fit to the dependent data. Maximum likelihood estimation is similar, and can achieve essentially the same result under the adoption of (often) a Gaussian model. In either case, a statistical model is assumed and parameters in it are estimated. Confidence intervals can be calculated on the local estimate and the weighting scheme giving some idea of the certainty with which individual model parameters are estimated, as well as the variation of the underlying random process.

To this basic class, non-linear regressors may be added – these build upon the same principles but the complications of non-linearity mean that a simple linear solution to the regression problem is impossible. Usually, the response surface parameters are determined by optimisation algorithms, for example where the problem can be cast as least squares, Gauss-Newton iteration and the Levenberg-Marquardt algorithm [65, 66] are often used. For estimation of the random model properties and confidence intervals for the regressors, the system is linearised around the solution point and treated as if it were an equivalent linear regression. Naturally, the statistics so obtained are used with caution as local behaviour may not be representative. To obtain a more realistic idea of the uncertainty one must resort to Monte Carlo techniques and numerical simulation [67].

The weakness of the above regression tools lies in the choice of global basis functions. Whilst a great many such functions and decompositions exist; ordinary, Lagrange or Chebyshev polynomials, Fourier and wavelet decompositions to name a few, it is not immediately obvious which of them provide a natural characterisation of the phenomenon under observation. Furthermore, it is difficult to obtain an interpolated surface that interpolates “exactly” all of the known data – an “exact”

interpolator is one that passes through the known data points [68]. Certainly one may for example, use a sufficiently high order polynomial to fit all the points, but the associated solution time is prohibitive and the resulting interpolation may be quite erratic away from the nodes – called Runge’s phenomenon [69] (p. 101) after Runge’s original paper [70]. To avoid making an arbitrary choice and achieve exact interpolation, a non-parametric interpolation may be used; such as local polynomial interpolation, inverse distance methods, splines or more generally the kriging estimator.

Non-parametric interpolations propose a basis (or kernel) for the estimation that is intrinsically defined by the data itself – global models for the response surface are avoided. The simplest such method is nearest neighbour interpolation, where the response surface simply assumes the value of the nearest (by some metric) node. The resulting interpolations are *not* continuous. A  $C^0$  continuous exact interpolation is produced by a Delaunay triangulation that connects an arbitrary set of nodes and then uses linear patches to define the response surface [71] – note that this interpolation is  $C^0$  continuous, but it is not  $C^1$ . Less generally, the topology and connectivity of the node points may suggest an interpolation – when data is arranged on a grid in a given coordinate system, it is common to use bi-linear patches or bi-cubic splines [72]. In all cases, higher order patches can be used [73], but it becomes increasingly tricky to organise good continuity between the patches, both at  $C^0$  and at higher derivatives.

A general way of enforcing continuity is to introduce b-splines (short for *basis-splines*) [74, 75]. These introduce the idea of basis functions for the parameterisation of a geometric surface. Indeed, spline curves and surfaces may be thought of as a generalisation of shape functions, as found in Finite Elements [9] for parameterisation of the FE solution and domain. However, spline curves are still dependent on the particular topology of the nodal results and the arbitrary definition of their connectivity. Furthermore, organising said connectivity becomes more complicated as one abandons the three spatial dimensions for general interpolations in  $\mathbb{R}^N$  – splines have been developed with physical geometrical description in mind. Inverse distance weighting for data interpolation overcomes these problems. It was first expounded by Shepard [68] and later modified by Liszka [76], who incorporated statistical uncertainty in the data. However, there still remain largely arbitrary

decisions to be made about important parameters in the interpolation – notably there are no clear statistical interpretations.

In general, response surfaces are important to engineers because they characterise trends and behaviours across multiple variables. This is at the heart of all engineering experimentation, often expressed as (Modern) Design of Experiment (MDoE). This is introduced in Fisher’s seminal text [64] and in more modern counterparts [2, 77]. Deloach presents DoE’s application in wind-tunnel testing and introduces the use of neural networks to interpolate the non-linear relations so obtained [48, 78, 79]. More traditionally, polynomial fits are sought – though it is noted that this can result in erratic fits, especially as the topology of the experimental design becomes more convoluted.

## II - 5. Kriging Interpolation

Kriging differs somewhat from the methods listed above as it proposes a model for spatial uncertainty upon which interpolation (in a statistical context, called estimation) is then based. In particular, it models the spatial data with variations on the stationary random function. It was developed in the earth sciences by its namesake D.G. Krige [80, 81], and later formalised by Matheron [82, 83] in the 1960s and ‘70s. It is now integral to the discipline known as *geostatistics*. However, perhaps due to the specialised nature of the application or the apparent inscrutability of the theory underlying it, kriging has not received a large amount of detailed attention from the broader scientific community until relatively recently.

Like the methods listed above, kriging is a non-parametric interpolator that can exactly fit a response surface through a set of nodes, although now more generally the data can be in  $\mathbb{R}^N$  and a behaviour is specified for spatial continuity (i.e. the smoothness of the interpolate). As described in numerous textbooks [7, 84-86] on kriging, a realisation of a stationary random function is assumed to interpolate the known data. By considering multiple correlated random functions, disparate data-sets may be employed in the interpolation, a technique known as *cokriging*. In an early study [87] cokriging was used to improve estimates of uranium ore grades by their correlation with measurements of radioactivity, and the technique is generally

presented alongside univariate kriging in most basic introductory textbooks [7, 84]. Most importantly in all such methods is the existence of a model for spatial randomness and covariance – as expressed by a *covariance* function. The type of random function that is modelled informs the type of kriging estimation and covariance function to use; *simple* kriging assumes a zero mean, stationary covariance function, *ordinary* kriging assumes a constant unknown mean, possibly incrementally stationary covariance function (randomness at up to the first derivative), *universal* kriging accommodates a trend in the data, and most generally, *intrinsic* kriging assumes a higher order random function in which the randomness occurs at higher derivatives [84, 86]. *Block* kriging is used to model the diminution of randomness at larger scales. This is a fundamental part of kriging as geologists (and geostatisticians) often need to estimate spatially averaged data, such as overall grades. Nonlinearity and distribution estimation are introduced to the model via *disjunctive* kriging, and *indicator* kriging [84, 86].

In the sphere of fluids simulation and experimentation, kriging has found application as a surrogate model for optimising response surfaces [88-90], a graphical interpolation method [91], characterising summary integral functional quantities [92, 93], interpreting meteorological surveys [94-96], and surveys of aquifer and petroleum reservoir fluid dynamics [97, 98]. In the last example, and also in the work of Laurenceau [89], one notes that via cokriging, it is possible to assimilate statistical data collected at dissimilar differential orders into an interpolation – these data are typically quantities connected via a differential relationship, for example fluid pressures and velocities, or temperature and heat flux. In an unpublished example used in Chilès' textbook [84] (pp. 319-323), Neumann (derivative) boundary conditions are imposed on an aquifer bounded by impervious geological structures. Also outlined in this textbook, is how the equations of potential fluid flow can be used to derive the covariance and cross-covariance relationships in this case – more detailed information is available in earlier work [99]. Cokriging thus forms a framework for the estimation of some classes of stochastic differential equations [100].

As a computational tool, kriging has also been used to construct shape functions for various meshless Galerkin methods of PDE solution [101, 102]. It has also long been noted that kriging also generalises spline interpolations [72, 103].

## II - 6. Filtering for Meteorological Simulations

It was mentioned that kriging has been used for data assimilation in meteorological studies [94-96]. More broadly, the data assimilation methods used in the meteorological sciences and in weather prediction, are quite close to the aims of the present authors. In both there is a need to assimilate information from numerical predictions and physical measurements, which in turn drive the numerical simulations.

These methods are very well established and there exist textbooks [6] that describe them – many authors [6, 104] attribute the original ideas to Gandin’s classic textbook [105] (originally in Russian, translated in 1965). The essential approach is similar to Kalman filtering [106], or more generally recursive Bayesian estimation [107], though due to the size and complexity of the evolving state vector, various simplifications are made. Recursive Bayesian estimates are used to predict the possibly multivariate, *state* of a system. This ‘state-vector’ is then updated on the basis of new information which allows one to better infer the random model – the covariance matrices, behind the prediction. When the error in the state vector is assumed to be a multivariate Gaussian distribution, the Bayesian estimates devolve upon the Kalman filter. As noted by Cressie [104], meteorological data assimilation is also very similar to kriging insofar as it is an “optimal interpolation” method, especially in its earlier guises where the covariance matrices were not updated, and essentially static.

The methods of data assimilation eventually evolved to their present forms, which are geared around guiding a numerical simulation of the phenomenon to provide timely prediction. These *variational methods* are often called 3DVAR and 4DVAR [108, 109]. In these the numerical model is almost continuously being updated with new information, and the compatibility of the incoming data with the model is of crucial importance. However, there is still a fundamental updating of the

model, which is a dynamic entity. Importantly, the chaotic non-linear behaviour of weather systems fundamentally impose a prediction horizon regardless of the quality of the model – the incoming data is used to steer a numerical model.

## II - 7. Motivation for Geostatistics and Kriging

In the above survey, it has been attempted to give an overview of the techniques and even opinions that are current in formal Verification and Validation. Additionally, a number of other fields have been reviewed apart from the traditional folds of fluids experimentation and analysis, because they display close synergies with the broader aims of this thesis. The present author has made use of tools in geostatistics to address the data synthesis problem. Whilst these tools do not yet handle all of the issues raised thus far, they do at last introduce a basic framework for the comparison of spatial data in fluid studies.

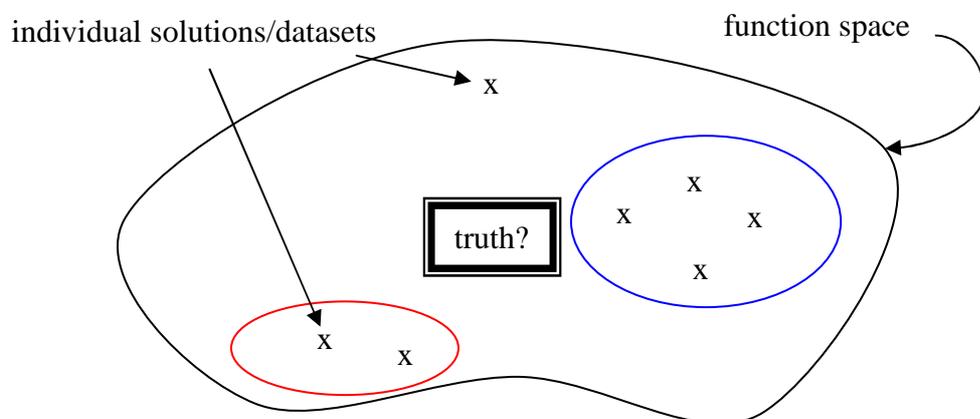
When analyses or experiments are conducted across different configurations of design variables – at different points in the design space, it is often useful to interpolate the results using a response surface model. To these design space variables, one can add spatial variables, for whose interpolation kriging was originally developed. Using a response surface model, it is possible to estimate the behaviour at points in the design space that have not yet been analysed, and guide the exploration of the design space by identifying the areas in it where inference is relatively bad. The use of response surfaces is an important part of Design of Experiment, and kriging performs this function well [90], especially when the pattern of data collection is irregular and not arranged on a grid or experimental design template. This is especially useful in multidimensional design spaces or ad hoc experimental schemes in which irregular data might be expected.

To the basic interpolation problem described above, one may add related sets of spatial data. This is the comparison problem that was introduced in the first chapter, where two questions were raised: how might one sensibly compare different sets of spatial data, and how can they act collectively to make better estimates? The present author has deliberately steered away from regarding this as an exercise in paring away error from the original data until all such datasets attain parity,

subsequently regarded as the “truth”. This is the usual approach in classical verification and validation [11]. Instead the fundamental means of comparison of spatial datasets has been addressed – if a legitimate correction to a set of results exists then it ought to be used in the first instance.

The extension of kriging to multiple datasets is called cokriging. In cokriging, the cross-correlation of data between similar datasets is modelled and used to improve estimation. In this thesis these basic cross-covariance models are used to develop relationships between datasets and define “whole-field” correlation. This informs a relativistic view of individual datasets. If one imagines each dataset as being representative of an underlying spatial function, then this becomes akin to a clustering exercise wherein the functions exist in some function space. Functions that are far apart by some metric or norm are dissimilar, whereas functions that are close together are similar. This is illustrated in Figure II—1 in which there are seven such spatial functions: four of them are proximal or self-similar, one of them is completely different from all the others, and two of them are alike but different to the other groupings. If similarity was judged with respect to a central point only, the “truth” in classical verification and validation, the outlying function would be judged no more remarkable than the furthest points in the two groups. It is this more complete picture of similarity that this thesis aims to inform – instead of a model for disparity the aim is to develop a model for correlation.

Spatial data in fluids are frequently presented as a static summary, sometimes due to limitations of measurement and analysis, but also because engineers are



**Figure II—1: Comparing whole-field solutions – truth vs clustering.**

usually interested in steady state conditions. In experimentation, the average behaviour over a certain time interval is reported – or when modelling, a steady state approximation is employed. There are many models for uncertainty that use Bayesian inferences or Kalman filters to update an underlying covariance model dynamically, but they are not directly applicable to static datasets. To get around this problem, most covariance models in geostatistics make use of a stationarity assumption: the underlying model describing the variation of the spatial data is, at some level, similar throughout the domain. This assumption is also necessary for this thesis because a covariance model must be inferred from a single *static* set of measurements. A very extreme example of this is the construction of stationary covariance models on numerical solutions approximating non-random functions, for which covariance may not strictly exist. Interpretations of this are explored by this thesis, in which the numerical data is considered to be the result of a numerical ‘experiment’, and the stationary spatial covariance serves to describe the smoothness of the numerical solution.

The broad reasons why geostatistical tools and kriging were adopted to address the data assimilation problem in modern aerodynamic studies have been outlined. It is by no means the only way in which the problem can be addressed, but in the absence of any established synthetic method, it covers the core issues of spatial uncertainty present in fluids metrology and simulation. Most attractive is the generality of the methods, and their robustness and simplicity.

# Chapter III - Theoretical Overview

The estimation theory behind the techniques used in this thesis may be unfamiliar to many engineers – especially those with a specialist interest in fluid dynamics and Computational Fluid Dynamics. Therefore, a brief outline of the theory is presented in this chapter. The concept of the random function is described with examples, and its use as an estimation model is introduced – ordinary kriging. These concepts and their notation are then extended to cover the more general case; universal kriging, and the multivariate case; cokriging. The concomitant spatial statistics to characterise the estimation models are developed in parallel.

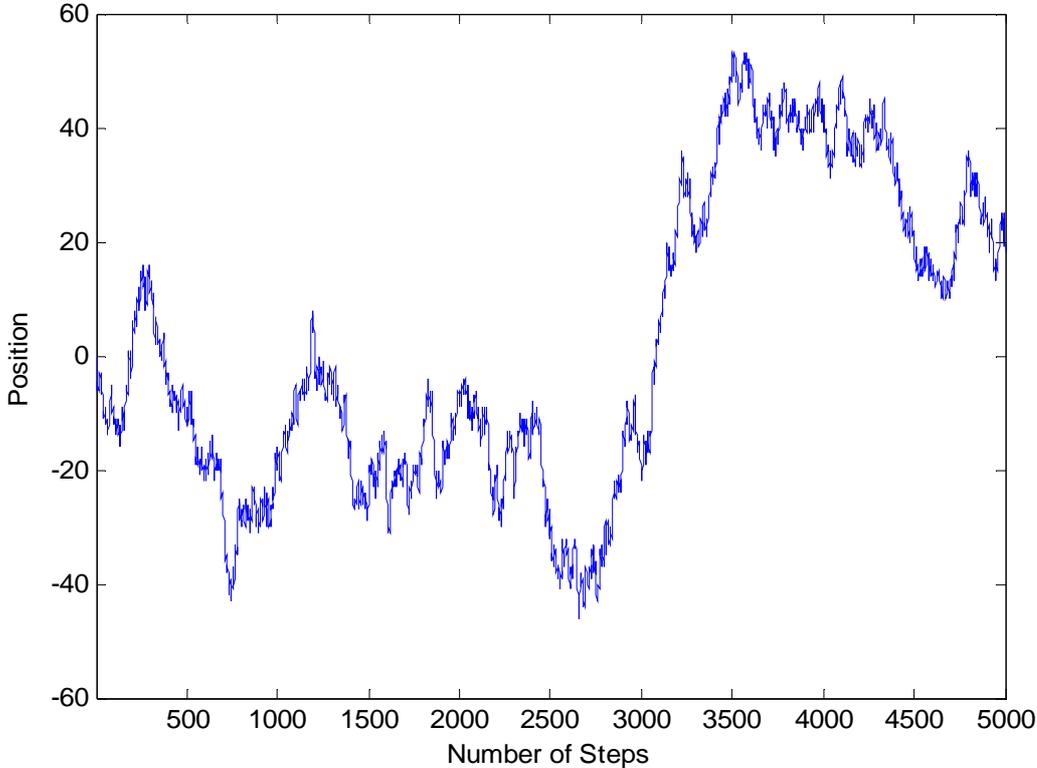
For a reader familiar with the kriging estimator, this chapter may only require attention for notation and the like. However, should the theory not be familiar, there are also good textbooks on the subject. The present author recommends Isaaks and Srivastava [7] or Cressie [110] for the uninitiated – the interested reader may find a more extended theoretical discussion in Chilès and Delfiner [84].

## III - 1. Random Functions and Estimation

Central to kriging is the random function model – this section describes what random functions are, and how they may be described. Random functions and

processes are like random variables, except that they take on their realisations over a domain in space, time or indeed generally in  $\mathbb{R}^N$ . Similarly, an observed instance of such a function is its realisation – a given random function may have a countable number of discrete realisations, or an infinity of continuously many realisations. In either case, a realisation forms a topology or field of values over the domain of the random function. In *universal* kriging used in this thesis, interpolations are achieved by modelling phenomena which are assumed to be second order stationary random functions with an unknown non-stationary trend. In these sorts of random functions, the variance is the same at all points in the domain, but the expected value varies according to the trend. Whilst this is a big assumption, it is ultimately useful and necessary given that often there are not enough data to infer better models. The so-called *intrinsic* hypothesis is used to extend the usefulness of the stationary random function model.

A useful way of visualising a random function is to imagine a so-called ‘random walk’. If one was to spin a coin a number of times and take steps to the left for heads, and steps to the right for tails, one would be performing a random walk –



**Figure III—1: A random walk, generated by a coin-tossing sequence.**

which given that all steps are equal, is essentially a *discrete* random function. Such a walk is plotted in Figure III—1. The walk is statistically self-similar – the same random process is driving the variation of displacement at all the points in the discrete function’s one-dimensional domain. Thus on average, that is over many realisations, one can imagine that the walk will not move away from zero displacement – the expected value of displacement is zero at every point along the walk. However, more often than not one *will* move away from the starting point and the longer the random walk progresses, the greater will be the dispersion of possible destinations: thus the variance of displacement from the starting position will increase with time. Moreover the expected value of the displacement at a point is strongly dependent on those values around it as one cannot move faster than one step at a time and the probability of a large jump in just a few steps is low.

A random function is something like a continuous version of a random walk, except now there is a continuum of points at which it will take a value – there are still rules for how the phenomenon varies from point to point. Generally, the continuity of a random function is characterised by its covariance function, which describes the covariance between two points on the random function. For example, say there is a continuous random function  $P$ , that takes values over a domain in  $\mathbb{R}^N$ . A realisation of it at a point, say  $P(\mathbf{x})$ , is called a *regionalised random variable* and the covariance function simply describes the covariance between values at two points it is defined as

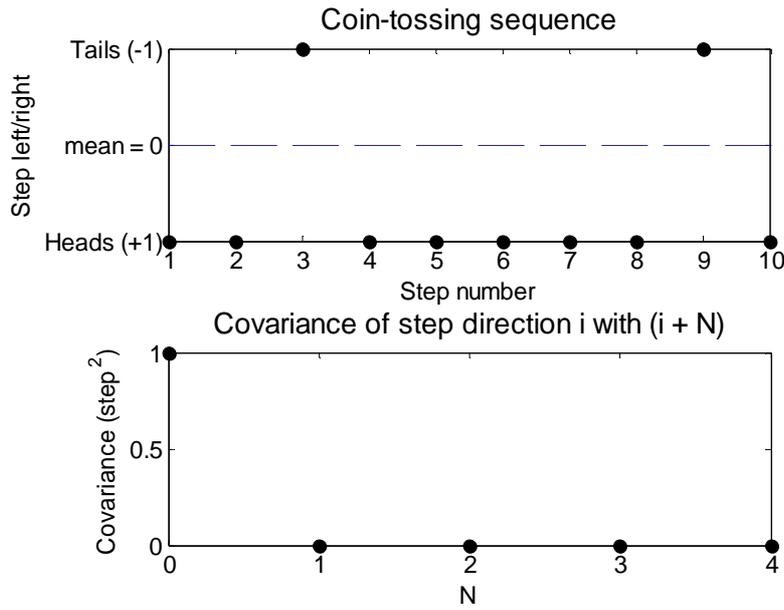
$$\text{cov}(P(\mathbf{x}), P(\mathbf{y})) = E\left[\left(P(\mathbf{x}) - E(P(\mathbf{x}))\right)\left(P(\mathbf{y}) - E(P(\mathbf{y}))\right)\right] \quad (3.1)$$

where  $E(\cdot)$  is the expectation function. This function is often written in an expanded form

$$\text{cov}(P(\mathbf{x}), P(\mathbf{y})) = E(P(\mathbf{x})P(\mathbf{y})) - E(P(\mathbf{x}))E(P(\mathbf{y})) \quad (3.2)$$

Considering again the example of the random walk in Figure III—1, it was noted that the expected value of displacement over all possible realisations was zero at all points in the domain. Thus one can say that where

$$E(P(\mathbf{x})) = 0 \quad \forall \mathbf{x} \quad (3.3)$$



**Figure III—2: Raw coin-tossing events and associated (discrete) stationary covariance function.**

the constant,  $\mu = 0$ . This means that the random walk is *first-order stationary* – that is, its expected value is equal to a constant  $\mu$  over its domain. The concept of stationarity also extends to the covariance function – strictly, *second-order stationarity* requires that the second moment of the random function is stationary:

$$E(P(\mathbf{x})P(\mathbf{y})) = E(P(\mathbf{x} + \mathbf{h})P(\mathbf{y} + \mathbf{h})) \quad \forall \mathbf{h} \quad (3.4)$$

Note that if  $P$  is zero mean, this expectation function is by definition, the covariance. Importantly, because of Equation (3.4), Equation (3.1) can be written in terms of just the separation between  $\mathbf{x}$  and  $\mathbf{y}$ ,  $\mathbf{h}$ ;

$$\mathbf{h} = \mathbf{y} - \mathbf{x} \quad (3.5)$$

Using (3.4), the product in Equation (3.2) may be written as

$$\begin{aligned} E(P(\mathbf{x})P(\mathbf{y})) &= E(P(\mathbf{y} - \mathbf{x})P(\mathbf{0})) \\ &= E(P(\mathbf{h})P(\mathbf{0})) \end{aligned} \quad (3.6)$$

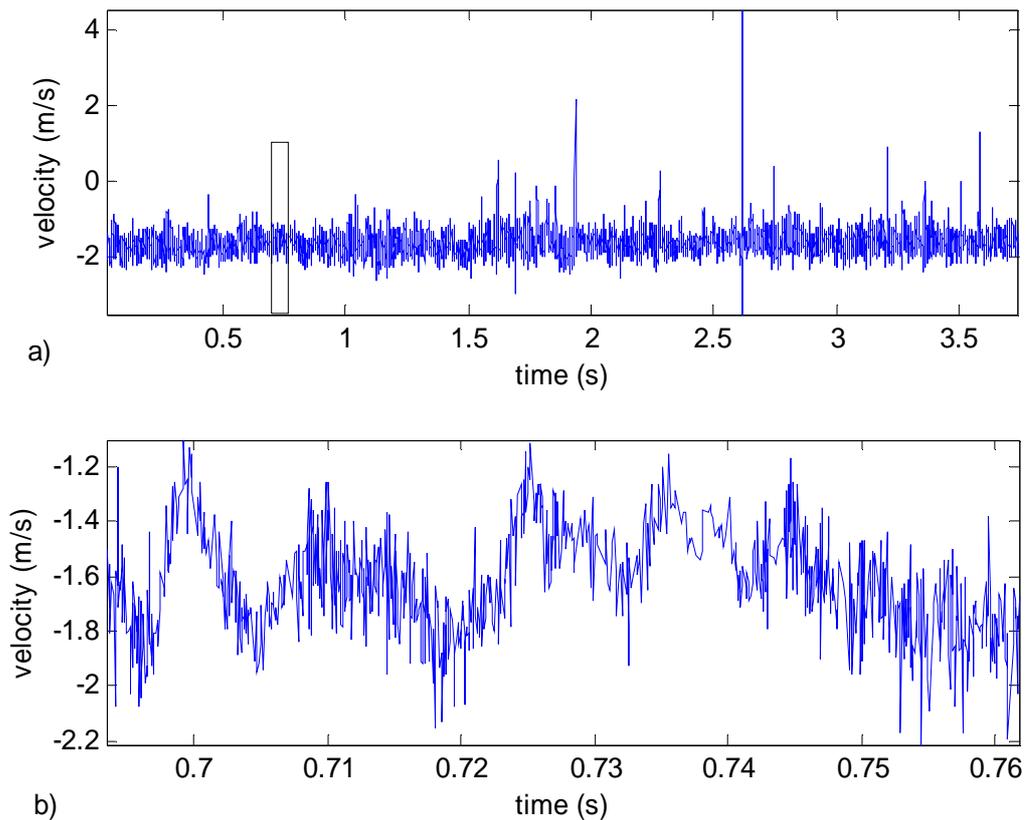
and given that the function is also first order stationary, the mean may be substituted for the remaining terms in Equation (3.2) so that

$$\begin{aligned}
\text{cov}(P(\mathbf{x}), P(\mathbf{y})) &= E(P(\mathbf{x})P(\mathbf{y})) - \mu^2 \\
&= E(P(\mathbf{h})P(\mathbf{0})) - \mu^2 \\
&= C(\mathbf{h})
\end{aligned}
\tag{3.7}$$

leaving a function in just  $\mathbf{h}$ . Thus for a *second order stationary* random function, the covariance does not depend on the absolute location of  $\mathbf{x}$  and  $\mathbf{y}$  in the domain, but only upon their relative separation  $\mathbf{h}$ . As a shorthand, the stationary covariance function will usually be denoted  $C(\cdot)$  hereafter.

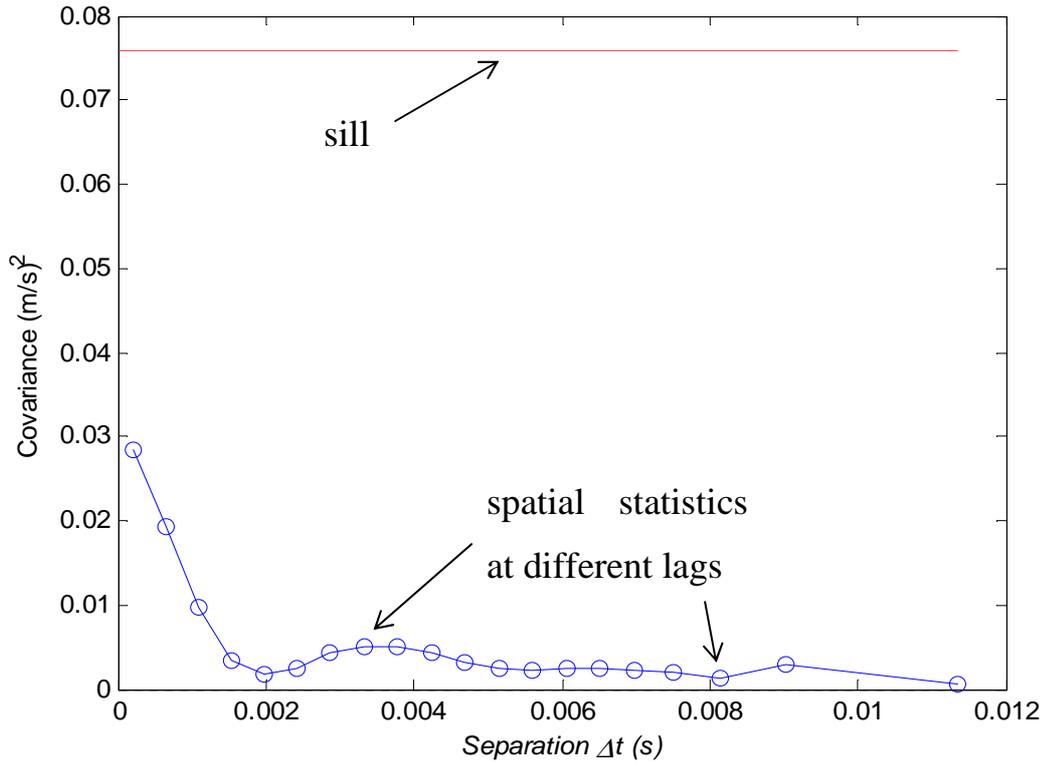
Unfortunately one sees immediately that this is *not* the case for the random walk in Figure III—1 because dispersion increases with time: if one looks at a pair of points separated by some time  $\Delta t$  early in the walk and compares them with another pair again separated by  $\Delta t$  later in the walk one can imagine immediately that as the dispersion of values is less for the first pair, the expected value of their product will *also* be less. Thus, the random walk is not a second order stationary random process. However, the events driving it are. If the series of coin tosses is viewed as a (discrete) random function with two values; +1 for heads and -1 for tails – the expectation is constant throughout the domain (and equal to zero), and the covariance function is independent of the absolute event timing. One is as likely to toss a heads at the beginning of the sequence as at the end. As the events are statistically independent coin tosses, the covariance function is equal to zero at all time separations except for at zero separation where it is equal to one – this is because an event’s correlation with itself must be one, the relationship is non-independent. As the covariance function goes straight from perfect correlation to zero correlation at adjacent separations, this behaviour is characterised as pure *noise*.

The above definitions and ideas also apply to continuous random functions. Realisations of these form a spectrum of continuous functions over a now continuous domain, and behaviour at a given point is likewise described by a now continuous probability distribution. Again, second order stationarity implies that the mean is constant throughout the domain and the covariance function only depends on relative separation, as expressed in Equation (3.5). Second order stationary processes are also commonly referred to as *weak* or *wide-sense* stationary or *covariance* stationary.



**Figure III—3: Time-series measurements of streamwise air velocity in a turbulent wake over; (a) 3.5 second interval, (b) inset taken between ~0.69s to ~0.76s.**

A good example of such a process is provided by the measurement of air velocity at a stationary point in a ‘steady-state’ turbulent flow. Figure III—3 shows such a series of such measurements taken using a Laser Doppler Anemometer behind a wingtip, in the wake of a trailing turbulent vortex. These results were produced in the UNSW low-speed wind tunnel by Vincent Galoul during his Masters practicum [111]. The velocity measurements will vary in time due to turbulent behaviour comprising both high frequency noise, the Kolmogorov spectrum and the large scale chaotic movement of vortices. However, given that the wind tunnel was operating at steady state conditions, the average value of the velocity would not be expected to change or trend upwards or downwards. Furthermore, it would not be expected that the physical processes behind the turbulent shedding would cause the velocity to vary in a different manner as the measurements progressed. Thus it is reasonable to view this as a second order stationary function.



**Figure III—4: Sample covariance function for LDA data.**

For the set of data  $\{t_i; v_i\}$  in Figure III—3, the one-dimensional “lag”  $\mathbf{h}$  will simply be the separation in time  $\Delta t$ , between pairs of measurements. The covariance between pairs of data separated by  $\Delta t$  is calculated using;

$$C(\mathbf{h}) \equiv C(\Delta t) = \frac{1}{N(\Delta t)} \sum_{(i,j)|\Delta t_{ij}=\Delta t} v_i v_j - \bar{v}^2 \quad (3.8)$$

where the  $(i, j)$  pairs in the summation term are chosen such that they are within some tolerance of  $\Delta t$  apart,  $N(\Delta t)$  is the number of such pairs, and

$$\bar{v} = \frac{1}{N} \sum_{i=1}^N v_i \quad (3.9)$$

The stationary covariance function is thus evaluated and plotted at a number of different separations  $\Delta t$  (indicated by circles) in Figure III—4. Each point in Figure III—4 is a statistic calculated from numerous pairs of data separated by  $\Delta t$ . Figure III—4 will be used as a particular, one-dimensional example of a covariance function to guide the following discussion.

As discussed in the context of the coin tosses, a stationary covariance function will reduce from the *total variance* of the random function at zero separation  $\mathbf{h} = \mathbf{0}$ , to zero covariance some distance away from the origin. Where  $\mathbf{h} = \mathbf{0}$ ,

$$\text{cov}(P(\mathbf{x}), P(\mathbf{x})) = \text{var}(P(\mathbf{x})) \quad (3.10)$$

and the distance at which the covariance function then reaches zero is called the *range*. This behaviour can be observed in Figure III—4. What it means is that the random function's values at points that are close together tend to be similar – they have high covariance, whereas points that are far from each other are essentially independent variables – they possess zero covariance. The covariance function's value exactly at the origin; the total variance of the data, may be estimated by the usual statistics for the total variance

$$^2 = \frac{1}{N} \sum_{i=1}^N (v_i - \bar{v})^2 \quad (3.11)$$

noting that the unbiased statistic that factorises by  $(N - 1)$  instead of  $N$  is hardly any different when  $N$  is relatively large. The function in Figure III—4 actually extends in both directions:  $\Delta t$  can be negative – in which case the same pairs will appear in the summation term in Equation (3.8) leading to the same covariances. Thus the covariance function is an even function. More generally it may be demonstrated that because of second order stationarity expressed in Equation (3.4)

$$\begin{aligned} \mathbf{C}(\mathbf{h}) &= \mathbf{E}(P(\mathbf{x} + \mathbf{h})P(\mathbf{x})) - ^2 \\ &= \mathbf{E}(P(\mathbf{x})P(\mathbf{x} - \mathbf{h})) - ^2 \\ &= \mathbf{C}(-\mathbf{h}) \end{aligned} \quad (3.12)$$

Again, the set of all values separated by  $+\mathbf{h}$  is the same as the set of all values separated by  $-\mathbf{h}$ , hence the expectation term is the same. In higher dimensionalities an anisotropy may be present, meaning that the range at which the covariance function goes to zero, will depend on the direction  $\mathbf{h}$ . It remains however, an even function as indicated by Equation (3.12).

## III - 2. Kriging

The statistical models introduced in the last section can be used to inform estimation and interpolation. Given the sampling in Figure III—1 of a particular realisation of a random function, the covariance function may be used to estimate the value of the random function at unsampled locations, thus interpolating the points. In one dimension this is a curve fitting exercise, and basing the estimation on the covariance function is called *kriging*. A kriged interpolation is indicated with the solid black line in Figure III—5, on a small subset of the original LDA data. The same technique is extensible to higher dimensions, for which it is essentially a response surface fitting. For example, in three dimensions one may regard the temperature distribution throughout a room as a continuous random function. A particular realisation of it might then comprise measurements from thermocouples at points in the room. In either case, the problem is to estimate the complete random function from a particular realisation of it.

Imagine a set of points  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \dots$ , at which there are the regionalised random variables  $P(\mathbf{x}_1), P(\mathbf{x}_2), P(\mathbf{x}_3) \dots$  as illustrated in Figure III—6. The actual realisations of the regionalised random variables are denoted by the lower case  $p_1, p_2 \dots$ , concatenated in a vector  $\mathbf{p}$ .

$$\mathbf{p} : p_k \sim P(\mathbf{x}_k) \quad (3.13)$$

Kriging assigns linear weights to these point values to produce an estimate, so where  $\mathbf{w}$  forms the weights and an estimate of the value at  $P(\mathbf{x}_0)$  is desired, there is an estimator of the form

$$\hat{p}_0(\mathbf{x}_0) = \sum_i w_i p_i \quad (3.14)$$

where the summation is performed over all elements in the vector  $\mathbf{p}$ . The hat indicates an estimated value, and note also the shorthand summation notation used above, that will be used subsequently. This is akin to the usual ‘Einstein’ notation, except that it makes explicit which subscripts are being summed over.

Kriging is a Best Linear Unbiased Estimator (BLUE), for appropriate classes of random functions. It is best, in the sense that it minimises the modelled variance.

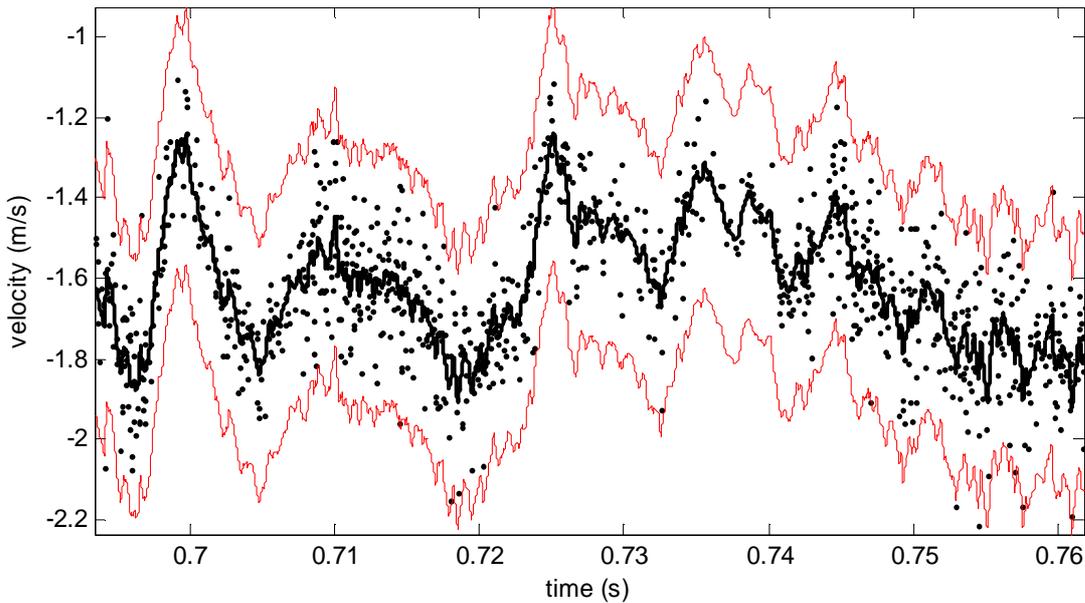
Given a model for covariance, i.e. – the covariance function  $C(\mathbf{x}, \mathbf{y})$  used to characterise the underlying random function  $P$ , a result may be derived for the variance of the estimation error  $\hat{p}_0 - p_0$ , where  $p_0$  is the true value of the random function at  $\mathbf{x}_0$ :

$$\begin{aligned}
\text{var}\{\hat{p}_0 - p_0\} &= \text{var}\left\{\sum_i w_i P(\mathbf{x}_i) - P(\mathbf{x}_0)\right\} \\
&= \text{var}\left\{\sum_i w_i P(\mathbf{x}_i)\right\} + \text{var}\{P(\mathbf{x}_0)\} - 2 \text{cov}\left\{\sum_i w_i P(\mathbf{x}_i), P(\mathbf{x}_0)\right\} \\
&= \sigma_p^2 + \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) - 2 \sum_i w_i \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_0))
\end{aligned} \tag{3.15}$$

This derivation is presented in full detail in Appendix D - 1. Strictly, it is applicable to the second order stationary random functions only, but essentially identical results may be obtained for situations in which there are possible spatial variations in the mean value – these variations are commonly referred to as a *drift*. In summary, the variance of the estimation error expressed above, is called the kriging variance and is notated as  $\sigma_K^2$ . Equation (3.15) may then be rewritten more neatly as

$$\sigma_K^2 = \sigma_p^2 + \sum_{i,j} w_i w_j C(\mathbf{h}_{ij}) - 2 \sum_i w_i C(\mathbf{h}_{i0}) \tag{3.16}$$

where



**Figure III—5: Kriged interpolation in black of LDA time-series in Figure III—3(b),  $\pm\sigma_K$  root kriging variance bracket in red, raw data as black points.**

$$\mathbf{h}_{ij} = \mathbf{x}_i - \mathbf{x}_j \quad (3.17)$$

and the stationary covariance function  $C(\mathbf{h})$  is used.

For a *best* estimate, this variance needs to be minimised with respect to the weights in its definition. However, there are also unbiasedness conditions on the weights – which will act as constraints on this minimisation. The type of constraint depends on the behaviour of the random function under examination. To achieve an unbiased estimate  $\hat{p}_0$ , the requirement

$$E(\hat{p}_0 - p_0) = 0 \quad (3.18)$$

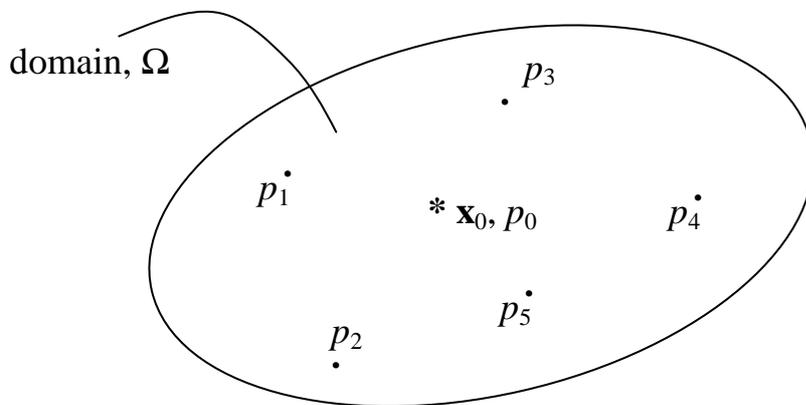
must be satisfied. If the random function is second order stationary, and additionally the mean is equal to zero (or is already known) at all points in the domain, this condition is satisfied automatically, and the minimisation is unconstrained. Thus to find the weights, one solves just:

$$\nabla_{\mathbf{w}} \frac{\partial}{\partial \mathbf{w}} = \mathbf{0} \quad (3.19)(a)$$

or,

$$\begin{aligned} \frac{\partial}{\partial w_k} \left[ \frac{1}{P} + \sum_{i,j} w_i w_j C(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_0) \right] &= 0 \\ 2 \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_k) - 2 C(\mathbf{x}_k, \mathbf{x}_0) &= 0 \\ \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_k) &= C(\mathbf{x}_k, \mathbf{x}_0) \end{aligned} \quad (3.19)(b)$$

In matrix notation, this can be re-written as:



**Figure III—6: Schematic of estimation domain.**

$$\mathbf{C}\mathbf{w} = \mathbf{c}_0 \quad (3.19)$$

Using the above equation to determine the weights in Equation (3.14) and so produce an estimate is called *simple kriging* estimation.

However, if the mean of the random function  $\mu$  is unknown but stationary, there will be extra conditions arising from the requirement in Equation (3.18). For an unbiased estimate in this case

$$\begin{aligned} E\{\hat{p}_0 - p_0\} &= E\left\{\sum_i w_i P(\mathbf{x}_i) - P(\mathbf{x}_0)\right\} = 0 \quad \therefore \\ \sum_i w_i &= 1 \end{aligned} \quad (3.20)$$

so the weights must sum to unity – a more complete proof is offered in Appendix D - 2. Again, for a *best* linear unbiased estimate the modelled variance is minimized with respect to the weights in its definition, but now subject to the new constraint in Equation (3.20). A Lagrange multiplier  $2\mu$  is introduced to take care of the constraint (it is factored by two so that conveniently, it becomes the unknown mean).

$$\nabla_{\mathbf{w}} \frac{2}{K} = 2\nabla_{\mathbf{w}} \left\{ \sum_i w_i \right\} \quad (3.21)(a)$$

This yields  $N$  equations, and the  $(N+1)^{\text{th}}$  unknown is solved by the constraint posed by Equation (3.20). Substituting for  $\frac{2}{K}$  one arrives at the following  $N$  equations, indexed by  $k$ ;

$$\begin{aligned} \frac{\partial}{\partial w_k} \left[ \frac{2}{P} + \sum_{i,j} w_i w_j C(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_0) \right] &= 2 \frac{\partial}{\partial w_k} \left[ \sum_i w_i \right] \\ 2 \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_k) - 2 C(\mathbf{x}_k, \mathbf{x}_0) &= 2 \\ \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_k) + \mu &= C(\mathbf{x}_k, \mathbf{x}_0) \end{aligned} \quad (3.21)(b)$$

Again, this can be re-expressed in matrix form, including the constraints in Equation (3.20):

$$\begin{bmatrix} C(\mathbf{x}_1, \mathbf{x}_1) & \cdots & C(\mathbf{x}_N, \mathbf{x}_1) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ C(\mathbf{x}_1, \mathbf{x}_N) & \cdots & C(\mathbf{x}_N, \mathbf{x}_N) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \\ 0 \end{bmatrix} = \begin{bmatrix} C(\mathbf{x}_1, \mathbf{x}_0) \\ \vdots \\ C(\mathbf{x}_N, \mathbf{x}_0) \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{c}_0 \\ 1 \end{pmatrix} \quad (3.21)$$

This system of equations, relating to the unknown stationary mean case, is often called the *ordinary kriging* equations. The covariance matrix on the left hand side effectively declusters the randomly sampled data, screening the effect of uncorrelated sample locations, and producing smooth interpolates.

Normally more than one estimate is produced at a time, solving the same matrix in Equation (3.21) with different right hand side vectors. The right hand side vector is dependent only on the location of  $\mathbf{x}_0$  in the estimation domain in Figure III—6, whereas the matrix on the left hand side depends on the location of the data-points  $\mathbf{p}$  in the domain. Writing the final estimate as

$$\hat{p}_0(\mathbf{x}_0) = \begin{bmatrix} \mathbf{p}^T & 0 \end{bmatrix} \left\{ \begin{bmatrix} \mathbf{C} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}^{-1} \begin{pmatrix} \mathbf{c}_0(\mathbf{x}_0) \\ 1 \end{pmatrix} \right\}, \quad (3.22)$$

it is seen that the covariance functions in  $\mathbf{c}_0$  effectively act as basis or shape functions for interpolation. As a result the interpolations will be as smooth as the covariance functions used to produce them.

Examining Equation (3.21), it is apparent that the same matrix inversion (or LU decomposition) can be used on multiple  $\mathbf{c}_0$  vectors. This is exploited when producing estimates over a field, say for the production of contour rasters or inputs to integration schemes – in any case, the matrix in Equation (3.21) is generated over just the points that are considered local to the point of estimation. In this way, the size of the matrix is greatly reduced and solution time is reduced. The exact scheme for the *windowing* is described later in Section IV - 6, but essentially the aim is to reduce the number of matrix solutions that are required, at the same time as limiting the matrix size of each solution and maintaining a minimum number of points in each sub-set of nodes considered local to the interpolation points. Having decided

upon these sub-sets, a local covariance matrix is generated for each and the value is decided upon.

### III - 3. Variogram Modelling

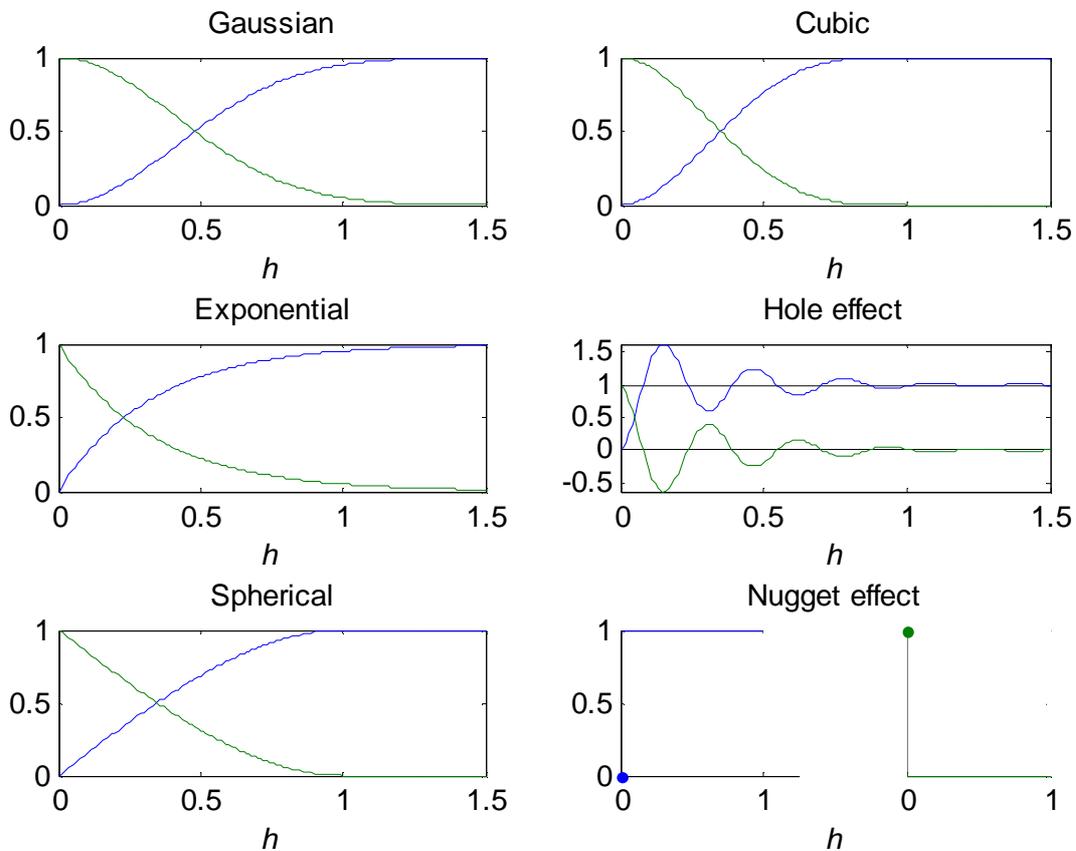
In the estimation algorithm described in the previous sections, the covariance function as it appears in Figure III—4, calculated by Equation (3.8), cannot be used directly. This is because it is required that the matrices in Equations (3.19) and (3.21) are *positive definite*. A given matrix  $\mathbf{A}$  is positive definite if and only if:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} > 0 \text{ for all } \mathbf{x}.$$

It is positive *semi*-definite if and only if:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0 \text{ for all } \mathbf{x}.$$

This amounts to a requirement that the modelled variance  $\frac{2}{k}$  is positive, and thus



**Figure III—7: Variogram models in blue, with equivalent model covariance function in green.**

$\kappa$  is real. Furthermore as can be seen in Figure III—4, Equation (3.8) only estimates the covariance function at points, so for generality a complete and continuous function over the whole domain of possible lag vectors  $\mathbf{h}$  ( $\Delta t$ ) must be available. For these reasons, a model must be fitted to the *sample* covariances in Figure III—4. This model is composed of canonical positive definite functions called *variogram models*.

The model for spatial continuity used in this thesis consists of linear combinations of the following functions, which are also presented in Figure IV—7.

Gaussian: 
$$\Gamma_G(h) = 1 - e^{-3h^2} \quad (3.23)(a)$$

Cubic: 
$$\Gamma_S(h) = \begin{cases} 7h^2 - \frac{35}{4}h^3 + \frac{7}{2}h^5 - \frac{3}{4}h^7 & h \leq 1 \\ 1 & h > 1 \end{cases} \quad (3.23)(b)$$

Exponential: 
$$\Gamma_E(h) = 1 - e^{-3h} \quad (3.23)(c)$$

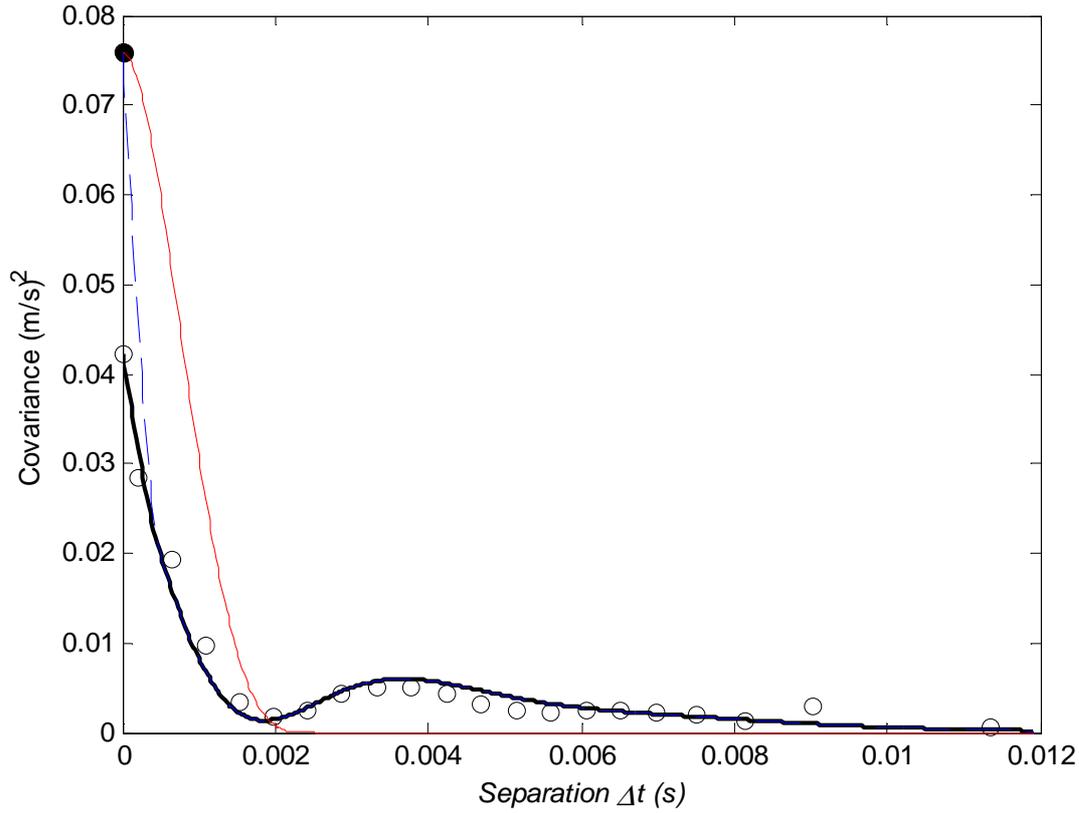
Hole effect: 
$$\Gamma_H(h, \ ) = 1 - e^{-3h} \cos( \ h) \quad (3.23)(d)$$

where  $h > \sqrt{3}$ .

Spherical: 
$$\Gamma_S(h) = \begin{cases} \frac{3}{2}h - \frac{1}{2}h^3 & h \leq 1 \\ 1 & h > 1 \end{cases} \quad (3.23)(e)$$

Nugget effect: 
$$\Gamma_N(h) = \begin{cases} 0 & h = 0 \\ 1 & h > 0 \end{cases} \quad (3.23)(f)$$

This list is far from exhaustive, but these functions have been chosen as they characterise particular spatial behaviours. Note that each of these functions is scaled to reach a maximum value of unity and a range of unity, and unlike the function in Figure III—4, they rise from zero instead of decreasing from total variance – they are in effect upside-down covariance functions. The reasons for this will be presented later, in the context of another structural tool, the *variogram*. It will suffice for the moment to imagine instead that functions  $(1 - \Gamma(\mathbf{h}))$  are being fitted to the points in Figure III—4. The solid black line in Figure III—8 shows one such fit. The actual function it describes is:



**Figure III—8: Fitting model covariance functions; best overall fit in black, best fit without nugget effect in blue, best Gaussian fit in red.**

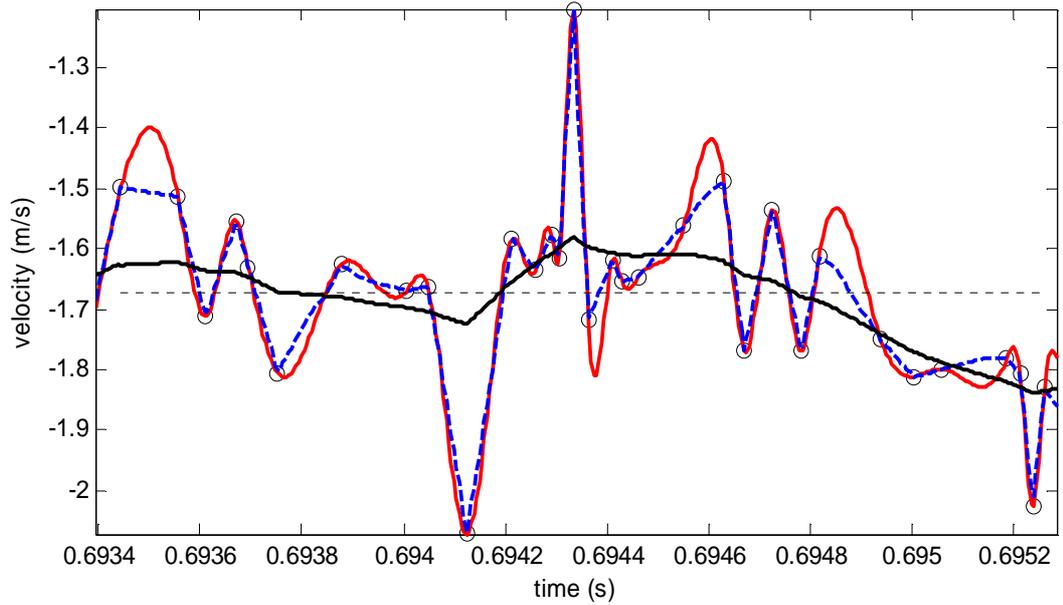
$$\begin{aligned}
 C(h) = C(\Delta t) = & 0.033666[1 - \Gamma_N(\Delta t)] + 0.01 \left[ 1 - \Gamma_S \left( \frac{\Delta t}{0.0005} \right) \right] + \\
 & + 0.0072 \left[ 1 - \Gamma_G \left( \frac{\Delta t}{0.011} \right) \right] + 0.025 \left[ 1 - \Gamma_H \left( \frac{\Delta t}{0.0038}, 1461.2 \right) \right]
 \end{aligned} \tag{3.24}$$

In this equation there is a linear combination of the functions in Equation (3.23), wherein  $h$  is scaled to fit the observed range in Figure III—4. The fit was obtained graphically in MS Excel™.

Thus in general, the form of the modelled covariance function is given by

$$C(\mathbf{h}) = \sum_i t_i \left[ 1 - \Gamma^i(\Phi_i(\mathbf{h})) \right] \tag{3.25}$$

where the functions  $\Phi_i(\mathbf{h})$  scale the lag vector (or in this case time separation  $\Delta t$ ). In this expression, each of the models may be used more than once, i.e. several of the  $\gamma^i$  may be spherical models (of type  $\gamma_S$ ), nested inside of each other with differently scaled ranges and different coefficients  $t_i$ . Importantly, these coefficients must be



**Figure III—9: Interpolation of LDA data (in blue), kriged using; best  $C(h)$  fit in black, zero-nugget  $C(h)$  fit in blue, cubic model  $C(h)$  fit in red.**

positive and the sum of the coefficients  $\sum t_i$  should be equal to the total variance of the underlying random function.

$$\sum_i t_i = 2 \quad (3.26)$$

Each canonical model characterises a type of spatial behaviour or *continuity*. The Gaussian model, based on the Gaussian distribution, characterises exceptionally smooth behaviours, so much so that it is actually a ‘pathological’ model. It infers that the random function’s value and all of its derivatives are known at a point (it is  $C^\infty$ ), thus it is no longer a random function as it may then be estimated via its Taylor Series at all other points in the domain. Using this model exclusively for estimation can lead to near singular covariance matrices in Equation (3.21). The exponential model is typical of less smooth behaviours, as is the spherical model. The spherical model is a generic model displaying linear behaviour at the origin, and zero covariance outside of the range. This effectively means that it disregards points outside of a given range – for which there will be zero entries in the covariance matrices. Its behaviour is similar to the exponential model, which does the same thing asymptotically as the separation goes to infinity. Finally, the hole effect model is designed to model sinusoidal variation in space, or layering.

At the other extreme, the nugget effect represents incoherent noise – where the data varies greatly over a very small distance. This is akin to the discrete coin tossing experiment which demonstrated zero covariance away from the origin, with a sharp peak at the origin (Figure III—2). This variance is often called *unstructured* variance, in contrast with the other variogram models that uncover some ‘structure’ in the data.

The modelling of the particular covariance function has a marked effect on the estimation properties of the kriging estimator. This is demonstrated using the LDA time-series data in Figure III—5. The data here are interpolated using ordinary kriging, as described previously, over a shorter window. The solid black lines in Figures III—5 and III—9 are the result of kriging estimation carried out at regular intervals of ten microseconds – far shorter than the average sampling rate of the LDA. Importantly, the kriging in both cases has been performed using the covariance model in Equation (3.24), indicated by the black line in Figure III—8. Inspecting this figure, it is seen that this covariance model fits the ‘sample’ statistics in Figure III—4 closely. Because it incorporates a large contribution from the nugget effect model, the interpolated line in Figure III—9 does not pass ‘exactly’ through the data-points: the nugget effect introduces a core uncertainty in the data, at each data-point. In actual fact, it does pass through these points, but discontinuously: the interpolation jumps discontinuously at the interpolated data points to the data-value at those points before returning discontinuously to its prior value.

If one was to decide that the interpolated line must pass through the points one would need to model the covariance function without a nugget effect. This would imply that there is no inaccuracy or uncertainty in any of the raw data, or at least that the statistical variation from this is tiny as compared with the physical fluctuation of actual windspeed in the tunnel. It was indicated to me by Vincent Galoul that this was indeed the case and that the data collected by the LDA were very exact measurements of instantaneous velocity. If this is true, it may be sensible to use the covariance model indicated in blue in Figure III—8. This has zero nugget effect, although as can be seen in this figure, it does not fit the sample statistics as closely as Equation (3.24). In a way, this interpretation permits what one expects to see, to change the interpretation of what is actually seen! This is not of course totally

legitimate scientific practice, but surprisingly common anyway – it is impossible to disavow oneself of any prior knowledge of the phenomenon as often the entire experiment is shaped by this ‘knowledge’ in the first place. It is observed that the interpolate or rather, expected value of it, changes – in Figure III—9 the original data-points are interpolated exactly by the blue line, corresponding to the covariance function without a nugget effect.

Indeed, one may go a step further and hypothesise that the turbulent velocity fluctuation is actually a  $C^2$  smooth function that is best represented by a cubic spline model. One of these, in red, is also fitted to the sample covariance points in Figure III—8. Agreement with the statistics is rather poor, but a simplistic appreciation of Newtonian kinetics and continuum mechanics might yet lead one to use this model, which when applied to the estimation of velocity produces the interpolate in Figure III—9. This passes through all the points with  $C^2$  smoothness, but implicitly makes a lot of assumptions about the phenomenon under observation. Noting the similarity between the cubic model and the Gaussian model, it is natural to investigate whether one may be substituted for the other. However, using a pure Gaussian covariance function leads to very badly conditioned and/or near singular covariance matrices. This is because the Gaussian model is effectively  $C^\infty$ , and therefore a pathological model. If the expected value is infinitely differentiable and smooth, the ‘random’ function is determined by its Taylor series and not in fact, random. From a modelling perspective however, this problem is overcome if the Gaussian model is used in conjunction with other models.

At the other extreme again, one could consider that the fluctuation observed in the data is purely random ‘white’ noise. If this were the case, the covariance function could be modelled with a pure nugget effect; zero everywhere except for at the origin (see Figure III—7). The interpolating function in this case is simply the average of all the data used in the estimation, also indicated in Figure III—9. In some cases, this will be an appropriate model – as previously noted, a certain amount of behaviour is read into a set of data and this influences its interpretation. The advantage of examining spatial statistics is that this behaviour can be observed and verified.

In Equation (3.25), the function  $\Phi_i(\mathbf{h})$  was introduced to scale the lag vector  $\mathbf{h}$  to fit the particular range of each simple variogram structure. This is straightforward to see in a one-dimensional time domain, but the situation becomes more complex in higher dimensions, where  $\mathbf{h} \in \mathbb{R}^N$ . In this case, each of the models may also exhibit some sort of anisotropy. To preserve the positive definite properties of the covariance function, this amounts to a stretching and shrinking in some given principal, orthogonal directions. The anisotropy may be expressed as the following norm on  $\mathbf{h}$ :

$$\Phi_i(\mathbf{h}) = \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \quad (3.27)$$

where  $\mathbf{T}$  is a positive semi-definite matrix (ensuring positivity of the term under the square root). Therefore, Equation (3.25) can be rewritten as:

$$\mathbf{C}(\mathbf{h}) = \sum_i t_i \left[ 1 - I^r \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right] \quad t_i > 0 . \quad (3.28)$$

The coefficients in the matrices  $\mathbf{T}^i$  and their respective weighting coefficients  $t_i$ , are tuned to fit statistics estimating the stationary covariance function in Equation (3.7). Equation (3.8) is a one dimensional example of such a statistic - more general statistics in  $\mathbb{R}^N$  are introduced in the next chapter, in Sections IV - 2 and IV - 3.

We have pursued a least squares fit for Equation (3.28). Firstly, statistics on the covariance function are generated at points, as they were in Figure III—4. Then a form (Equation (3.28)) for the model is chosen and the parameters in it are tuned so that the sum of the squared differences between the statistics and the model is minimised. This requires a non-linear least squares algorithm, for which purpose the present author has used the Levenberg-Marquardt algorithm. This algorithm and its particular implementation are elaborated upon in Section IV - 5. Having determined the model covariance function, the covariance matrices in Equation (3.21) can be calculated and one may solve for the kriging weights and the kriging variance – so producing an estimate.

### III - 4. Universal Kriging

The same principles may also be used to derive the *universal kriging* system, the more general form that has been programmed for this thesis. In universal kriging,

it is assumed that the random function under examination possesses a *drift*: the value of the mean is not stationary and changes in a structured fashion over the domain. The drift is assumed to be composed of a linear sum of underlying basis functions  $f^i$ :

$$f(\mathbf{x}) = \sum_i a_i f^i(\mathbf{x}) \quad (3.29)$$

where  $a_i$  are the coefficients. Importantly, although its form is specified, the drift itself is unknown – meaning that the values of the coefficients also need to be determined in the estimation procedure. This is rather like when for ordinary kriging, the mean was unknown, and determined in the estimation. Therefore, a function  $Q(\mathbf{x})$  of the form

$$Q(\mathbf{x}) = P(\mathbf{x}) + f(\mathbf{x}) \quad (3.30)$$

is estimated where  $P(\mathbf{x})$  is a zero mean random function component. The estimator is again a sum

$$\hat{q}_0 = \sum_i w_i q_i \quad (3.31)$$

Often the basis functions used will be simple monomials, leading to a polynomial representation of the drift – but these are by no means exclusive. Any function in  $\mathbf{x}$  is acceptable, including orthogonal decompositions, Fourier decompositions, and even secondary interpolated response surfaces. Obviously, the functions must not be linear combinations of each other, as then singular solutions would arise from the non-unique representations. The present author has used the very simplest drift functions – the monomial basis functions. The monomial basis functions of order  $K$  in  $\mathbb{R}^N$  will consist of functions of the form:

$$m^i(\mathbf{x}) = x_1^{I1} x_2^{I2} x_3^{I3} \dots x_N^{IN}$$

where  $I1 + I2 + I3 + \dots + IN \leq K$  . (3.32)

Put more simply, they are the individual terms in a polynomial of order  $K$  in  $\mathbb{R}^N$  – thus the second order monomials in  $\mathbb{R}^2$  are  $1, x, y, x^2, y^2$  and  $xy$ .

Whatever basis is chosen, the unbiasedness condition expressed in Equation (3.18) now results in the following set of equations.

$$\begin{aligned}
E\{\hat{q}_0 - q_0\} &= E\left\{\sum_i w_i [P(\mathbf{x}_i) + \mathbf{f}(\mathbf{x}_i)] - [P(\mathbf{x}_0) + \mathbf{f}(\mathbf{x}_0)]\right\} \\
&= \sum_i w_i E\left\{P(\mathbf{x}_i) + \sum_j a_j f^j(\mathbf{x}_i)\right\} - \sum_j a_j f^j(\mathbf{x}_0) \quad (3.33)(a)
\end{aligned}$$

which because  $P(\mathbf{x})$  is zero mean and Equation (3.18) sets the above expression to zero, can be rearranged to yield

$$\sum_i w_i \mathbf{f}^j(\mathbf{x}_i) = \mathbf{f}^j(\mathbf{x}_0) . \quad (3.33)$$

The full derivation of Equation (3.33) is presented in Appendix D - 4. In matrix form, Equation (3.33) may be written as

$$[\mathbf{f}_1, \mathbf{f}_2, \dots] \mathbf{w} = \mathbf{F} \mathbf{w} = \mathbf{f}_0 \quad (3.34)$$

where the vectors  $\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \dots$  are the basis functions  $f^j$  evaluated at the points  $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots$ , respectively, and  $\mathbf{f}_0$  is evaluated at the point of estimation  $\mathbf{x}_0$ .

$$\mathbf{f}_i = \begin{pmatrix} f^1(\mathbf{x}_i) \\ f^2(\mathbf{x}_i) \\ \vdots \end{pmatrix} \quad (3.35)$$

The derivation of kriging variance expressed in Equation (3.15) is slightly modified, but results in exactly the same relation on the *detrended* random function  $P$  of the original function  $Q$ :

$$\begin{aligned}
\text{var}\{\hat{q}_0 - q_0\} &= \text{var}\left\{\sum_i w_i Q(\mathbf{x}_i) - Q(\mathbf{x}_0)\right\} \\
&= \text{var}\left\{\sum_i w_i [P(\mathbf{x}_i) + \mathbf{f}(\mathbf{x}_i)] - [P(\mathbf{x}_0) + \mathbf{f}(\mathbf{x}_0)]\right\} \quad (3.36) \\
&= \frac{2}{P} + \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) - 2 \sum_i w_i \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_0))
\end{aligned}$$

The complete derivation of the above is presented in Appendix D - 3.

When Equations (3.34) are used as constraints on the minimisation of Equation (3.36), this results in the following set of equations, expressed in matrix form as

$$\begin{bmatrix} \mathbf{C} & \mathbf{F}^T \\ \mathbf{F} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{f}_0 \end{bmatrix}. \quad (3.37)$$

These are the universal kriging equations, that have been implemented for this thesis. Note that these equations reduce to the ordinary kriging equations when a single basis function, consisting of  $f^1 = 1$  is used.

### III - 5. Structure Identification

Having described univariate estimation, the covariance functions used in it, and their determination, are now described in more detail. This activity is often termed structure identification, and was touched on before in “Variogram Modelling”, where the *model* covariance function was introduced. The structures identified here were second order stationary, but more generally one encounters random functions which may contain first order drifts and non-stationary second order behaviour.

It has so far been remarked that the covariance function is properly expressed by Equation (3.1), but as there is usually a lack of repeat realisations to determine this form, an assumption of stationarity is made. Whilst this assumption may not be strictly true, it is acceptable inasmuch as its violation tends only to affect the determination of the covariance function over large separations. Because of the relation in Equation (3.10), its behaviour approaching the origin should approach that of the actual covariance function. Furthermore when using kriging for point estimation as described above, it is the data-points closest to the estimated point that will carry the largest weights – thus it is the covariance function’s behaviour over small separations that will dominate the final estimation. Intuitively, points that are far away are not important to the estimation. Therefore, the assumption of stationarity is difficult to verify, but it is very useful. Without it, it would simply be impossible to characterise the covariance functions on the basis of a single “static” set of measurements.

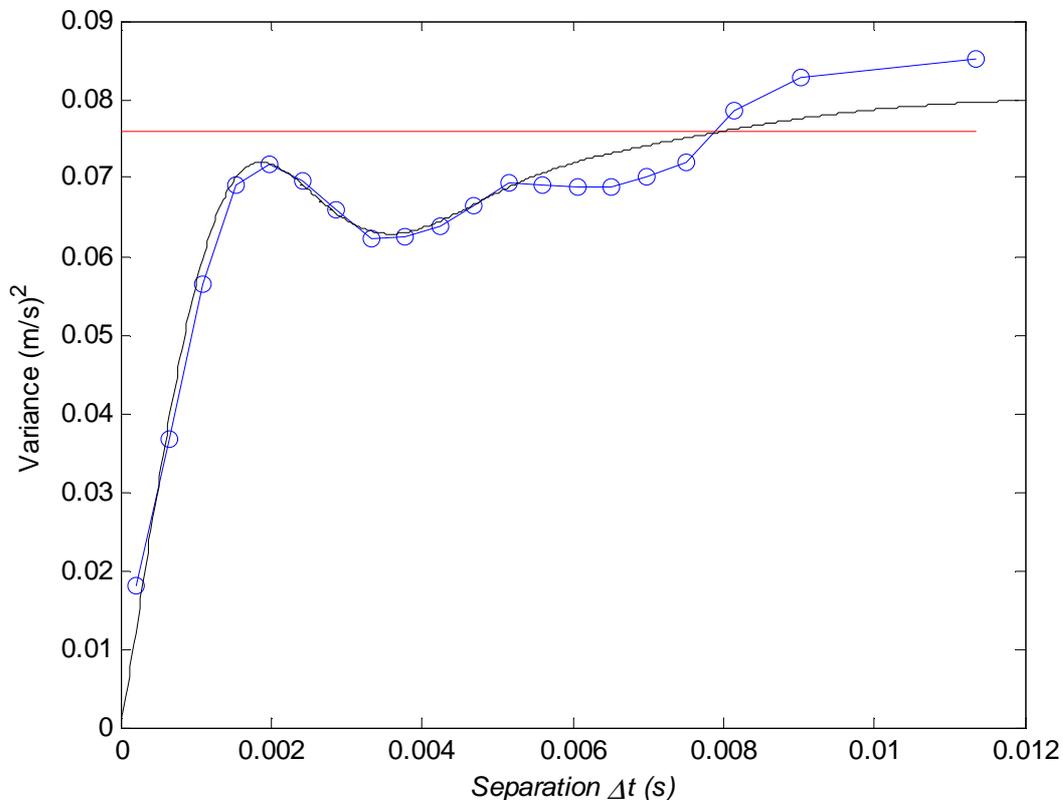
There exist other, milder forms of the assumption of second order stationarity. Notably, in geostatistics it has been customary to use the *variogram*

function instead of the covariance function. The variogram is related to, but differs from the covariance function in that it is a variance calculated on the *difference* in random function value between two points. Properly called the semi-variogram, it is described by

$$\Gamma(\mathbf{h}) = \frac{1}{2} \text{E} \left[ (P(\mathbf{x} + \mathbf{h}) - P(\mathbf{x}))^2 \right] \quad (3.38)$$

which is effectively half the variance of the first order difference operator as defined on the lag  $\mathbf{h}$ . Throughout this thesis, this function shall be referred to as simply the *variogram* (its common name). Because it is a squared difference, it is insensitive to drift in the mean value of the random function and is suitable for a broader class of random functions.

Specifically, the variogram treats the family of zero-th order *intrinsically* stationary random functions – for which the increments, if not the values, are stationary. A simple, discrete example of one such function is the random walk that was introduced earlier in Figure III—1. It was previously concluded that the function



**Figure III—10: Sample variogram for LDA data; sample points as circles, fitted variogram model in black.**

was not second order stationary, or covariance stationary, by conducting a simple thought experiment. However, there is certainly a self-similar random process that drives this function's variation in time as it has been designed into it. The process is unearthed from the raw data by examining the variogram of the data. This is estimated over lags of  $\Delta t$  using

$$\Gamma(\Delta t) = \frac{1}{2N(\Delta t)} \sum_{(i,j)|\Delta t=t_j-t_i} [v_i - v_j]^2 \quad (3.39)$$

and its value at different lags  $\Delta t$  is plotted in Figure III—10. Just as for Equation (3.8),  $N$  is the number of data pairs at a separation of  $\Delta t$  and the summation is over all such pairs, indexed by  $i$  and  $j$ .

Like the covariance function, the variogram is symmetric. Because they are subsequently squared, the difference terms in Equation (3.38) may be reversed, so that

$$\Gamma(\mathbf{h}) = \frac{1}{2} \mathbf{E} \left[ (P(\mathbf{x}) - P(\mathbf{x} + \mathbf{h}))^2 \right] \quad (3.40)$$

The expectation function operates over all possible values of  $\mathbf{x}$ , so the above statement is equivalent to;

$$\begin{aligned} \Gamma(\mathbf{h}) &= \frac{1}{2} \mathbf{E} \left[ (P(\mathbf{x} - \mathbf{h}) - P(\mathbf{x}))^2 \right] \\ &= \Gamma(-\mathbf{h}) \end{aligned} \quad (3.40)$$

thus symmetry is proven. Alternatively, one may assert that all the lags  $+\mathbf{h}$  in the random function domain must also exist for lags  $-\mathbf{h}$ . Unlike the covariance function, note that the variogram always rises from zero at the origin because:

$$\begin{aligned} \Gamma(\mathbf{0}) &= \frac{1}{2} \mathbf{E} \left[ (P(\mathbf{x}) - P(\mathbf{x}))^2 \right] \\ &= \frac{1}{2} \mathbf{E}[0] = 0 \end{aligned} \quad (3.41)$$

Furthermore, it is possible that the variogram has unbounded variance at large separations. Figure III—11 presents the discrete variogram of position pertaining to the random walk in Figure III—1. It rises linearly and without bound as separation  $h$  increases. This is because the change in position of the random walk over  $n$  coin-

tosses is necessarily the same thing as the variance of the sum of  $n$  coin tosses – and the variance of the sum of  $n$  coin tosses is simply  $n$  (given that each toss is independent). Hence the variance of one coin toss at  $\Delta t = 1$ , two coin tosses at  $\Delta t = 2$ , three coin tosses at  $\Delta t = 3 \dots$ , are respectively equal to one, two, three..., and the variogram is thus linear.

For a second order stationary random function with finite variance, the stationary covariance function  $C(\mathbf{h})$  is related to the variogram  $\gamma(\mathbf{h})$  by

$$C(\mathbf{h}) = \sigma^2 - \gamma(\mathbf{h}) \tag{3.42}$$

Straightaway, it can be seen that there is no stationary covariance representation for the variogram in Figure III—11, as it seems that this variogram has unbounded variance which would mean that the covariance function would be negative – an impossibility. However, the variogram of the velocity data in Figure III—3 does seem to reach a finite value, as can be seen in Figure III—10. It is apparent that the relation in Equation (3.42) is more or less observed – the variogram resembles the covariance function, offset and turned upside-down. However, the relation is subject to statistical error, thus it is more faithfully observed by the statistics located at smaller separations or lags.

Thus in contrast to covariance functions which decrease from total variance to zero at some range, bounded variograms rise from zero and plateau at the total variance at some range. The value at which they level off is often called the *sill*, and it is theoretically equal to the total variance of the random function. This explains

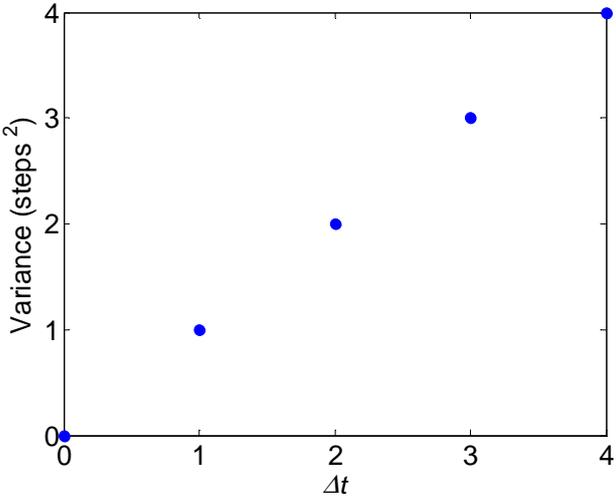
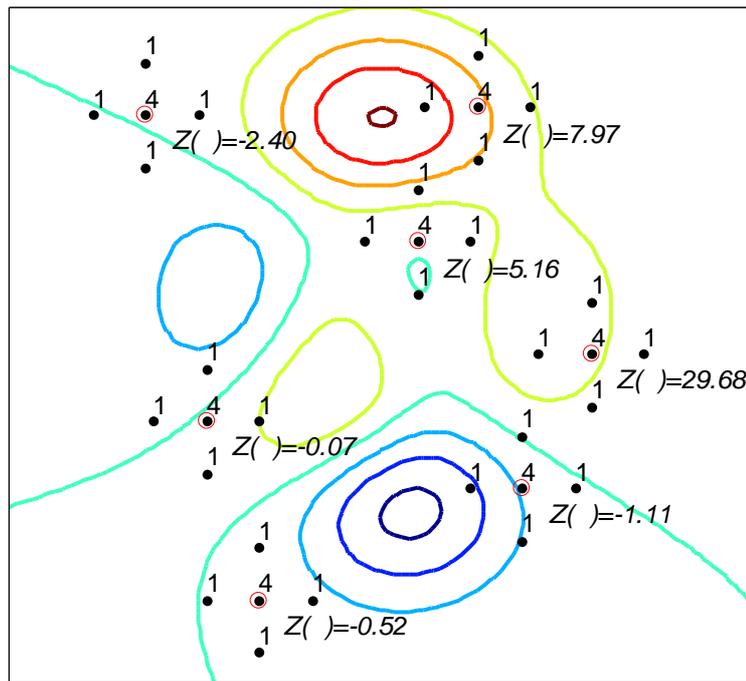


Figure III—11: Discrete variogram for the random walk.



**Figure III—12: ALC-1 Laplacian metric (template, black points) applied at different points (red circles) on a given background function.**

why all of the functions in Equation (3.23) rise from zero to a unit value at a range of one – they are variogram models, not covariance models. Because geostatisticians have traditionally used the variogram as their main structural tool, the canonical forms that are used to fit the sample structural statistics are known as variogram models, and that activity is known as variogram modelling.

Of particular interest in both the variogram and the covariance function is the limiting behaviour as the origin is approached. For data that comprises some random noise component ie; a  $C^0$  discontinuity, or high small scale variability – one should see a similar  $C^0$  discontinuity at the origin of the variogram and covariance: the variogram function will jump discontinuously from zero at the origin to another real value less than or equal to the sill, and the covariance function will fall discontinuously by the same amount, as illustrated in Figure III—7 for each variogram model. As previously discussed, the size of this jump represents the magnitude of short scale variability or ‘noise’. It is somewhat easier to calculate by means of the variogram, wherein it is directly measurable, simply because the variogram must be zero at zero separation (Equation (3.41)). For the covariance function, an accurate assessment of total variance of the random fluctuations must first be made, from which the intercept can then be subtracted.

As mentioned above, the variogram may be regarded as the variance of the first order difference operator, defined on  $\mathbf{h}$ . This introduces the means by which the variogram and the covariance function may be extended to cover higher order randomness. The first order differencing operator is a simple example of what is known as an *Allowable Linear Combination* at order zero (ALC-0), which means that it is a metric on a function that sends zero-th order polynomials (i.e. constants) to zero. A *metric*  $Z(\lambda)$  is essentially a template that samples an underlying function  $Z$  at locations relative to  $\mathbf{x}$  and weights them in some way so that it measures

$$Z(\lambda) \equiv \sum_i w_i Z(\mathbf{h}_i + \mathbf{x}) \quad (3.43)$$

as demonstrated in Figure III—12. These concepts shall be introduced more completely in Chapter V. This template, like the differencing operators, is in some sense translatable: one can slide it around the domain of a function and it will return a value, based on the topology of what it is measuring. In Figure III—12, a Laplacian metric is applied at the red circles  $\mathbf{x}$ , and where the relative locations  $\mathbf{h}_i$  are denoted with black dots, the values 2.4, 7.97, 5.16 and 12.02 are produced on the underlying function. Thus the first order difference operator is a metric where there are two points weighted by plus and minus one. If the underlying topology is constant throughout the domain, it will always return a value of zero.

More generally, an Allowable Linear Combination at order  $k$  (ALC- $k$ ) is a metric that sends  $k$ -th order polynomials to zero; for example, the Laplacian template always sends an underlying linear function to zero. The variogram assumes that the variance of the Allowable Linear Combinations of order zero on  $P(\mathbf{x})$  is stationary; dependent only on separation  $\mathbf{h}$ . Such a random function is called *intrinsically stationary* at order zero. Thus, the basic concepts can be extended to cover higher order random behaviours, called *Intrinsic Random Functions* at order  $k$  (IRF- $k$ ) by examining the variance of ALC- $k$  metrics. Although complex, the IRF- $k$  framework is what extends the definitions of stationarity in kriging, and additionally makes them tenable at lower orders – ordinary kriging is essentially intrinsic kriging at order zero.

Another useful measure of spatial continuity is provided by the (auto-) correlation function, described by

$$\mathbf{C}(\mathbf{h}) = \frac{\mathbf{C}(\mathbf{h})}{2} \quad (3.44)$$

Whilst it is only a factored form of the covariance function, its actual estimation, at a particular lag can be more robust as the result is normalised to be between zero and one. When the denominator is appropriately estimated, it helps filter out artefacts of the sampling process and drift: this will be described in Sections IV - 2 and IV - 3.

### III - 6. Multivariate Estimation – Cokriging

The foregoing discussion describes kriging for a single random function. Often situations arise in which there are a number of random functions  $P^1, P^2, P^3 \dots$  that are correlated in some way. If they are correlated, it ought to be possible to use realisations of the secondary variables,  $P^2, P^3 \dots$  to improve the estimation of the primary variable – especially if the variables are strongly cross-correlated and direct knowledge of the primary variable is poor.

In the geological sciences where kriging was initially developed, it is common to have concentrations of say, tin compounds as predicted by a borehole survey, and silver compounds as predicted by a different survey. As it is known that tin is often found with silver, one might expect to improve the estimation of the primary variable tin by considering its correlation with the secondary variable silver. It is quite easy to imagine similar scenarios within the context of fluid dynamics and anemometry. For example, one may have a room which is instrumented with thermocouples and additional devices for measuring air density, where the thermocouples are hung quite densely but the air density measurements are available at relatively few points in the room. If it was the estimation of the air density field that was of real interest, the question arises; how might one incorporate the thermocouple information to obtain a more detailed picture? After all, one would expect the two to bear some close relation, as by the ideal gas law;

$$\underline{P} = RT$$

therefore  $\frac{1}{\rho} = \nu \propto T$

By viewing temperature and specific volume, as two random functions that bear a good statistical correlation, cokriging provides a way of resolving this question.

The derivation is very similar to that of univariate estimation discussed above and similar notation to that used in the previous sections shall be adopted. In the multivariate case, there are now multiple random functions  $P^1, P^2, \dots$  for which there are particular realisations  $\mathbf{p}^1, \mathbf{p}^2, \dots$  – note that these vectors may not be the same length because there are typically different numbers of nodes at which the secondary data has been measured. As the theory can become complicated by the appearance of many different indices, greek letters shall be used to index the random functions and roman letters shall index the points within their realisations.

$$\mathbf{p} : p_k \sim P(\mathbf{x}_k) \quad (3.45)$$

A cokriged estimate of the primary quantity  $P^1$  at  $\mathbf{x}_0$  is now a linear weighted sum of both the surrounding  $\mathbf{p}^1$  values and the secondary variables  $\mathbf{p}^2, \mathbf{p}^3, \dots$ .

$$\begin{aligned} \hat{p}_0^1 &= \sum_{k,} w_k p_k \\ \hat{p}_0^1 &= \sum \mathbf{w} \cdot \mathbf{p} \end{aligned} \quad (3.46)$$

In Section III - 2, the covariance function was introduced. In the context of multivariate kriging this is properly called the *auto*-covariance function, as cokriging also requires a means of spatially cross-correlating the different variables. This functionality is provided by the *cross*-covariance function. From now on, the auto-covariances in Equations (3.1), (3.15) and (3.21) shall be denoted by

$$\text{cov}(P(\mathbf{x}), P(\mathbf{y})) = C(\mathbf{x}, \mathbf{y})$$

and to generalise, cross-covariances shall also be denoted

$$\text{cov}(P(\mathbf{x}), P(\mathbf{y})) = C(\mathbf{x}, \mathbf{y}) . \quad (3.47)$$

The cross-covariance function expresses the covariance between two points in the domains of two different random functions, and is used in a similar fashion as the auto-covariance function – which of course is a special case. Its estimation and analogous stationarity properties will be discussed later.

Just as before, the aim of multivariate kriging is to reduce the kriging variance – which is now expressed as;

$$\begin{aligned}
 \sigma_{CK}^2 &= \text{var} \left\{ \hat{p}_0^1 - p_0^1 \right\} \\
 &= \text{var} \left\{ \sum_i w_i P(\mathbf{x}_i) - P^1(\mathbf{x}_0) \right\} \\
 &= \sigma_1^2 + \sum_{i,j} w_i w_j C(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_i C_1(\mathbf{x}_i, \mathbf{x}_0) .
 \end{aligned} \tag{3.48}$$

The derivation of the above relation is presented in Appendix D - 5. If the random functions  $P^1, P^2, \dots$  are second order stationary with zero mean, the minimisation of kriging variance expressed by Equation (3.48) yields

$$\nabla_{\mathbf{w}} \sigma_{CK}^2 = \mathbf{0}$$

thus

$$\sum_j w_j C(\mathbf{x}_i, \mathbf{x}_j) = C_1(\mathbf{x}_i, \mathbf{x}_0) \tag{3.49}$$

or in matrix form;

$$\begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{21} & \cdots \\ \mathbf{C}_{12} & \mathbf{C}_{22} & \\ \vdots & & \ddots \end{bmatrix} \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{c}_0^1 \\ \mathbf{c}_0^2 \\ \vdots \end{pmatrix}, \tag{3.50}$$

where each submatrix  $\mathbf{C}_{\alpha\beta}$  is a matrix of covariances relating  $P^\alpha$  and  $P^\beta$ .

$$\mathbf{C} = \{C\}_{mn} = [C(\mathbf{x}_m, \mathbf{x}_n)] \tag{3.51}$$

Equation (3.50) can now be solved for  $\mathbf{w}_1, \mathbf{w}_2, \dots$ , the weights needed to evaluate  $\hat{p}_0$ .

### III - 6.1. Independent drift functions

Equation (3.50) is complicated somewhat should the random functions include some drift component or trend. To maintain unbiasedness the weights are now subject to sets of constraints which effectively detrend the raw data. The nature of these constraints depends on how the drifts are related. Let the drift functions for the random function  $P^\alpha$  be denoted as:

$$f(\mathbf{x}) = \sum_i a_i f^i(\mathbf{x}) . \tag{3.52}$$

We shall modify the vector form of these, previously introduced in Section III - 4 by using

$$\mathbf{f}_i = \begin{pmatrix} f^1(\mathbf{x}_i) \\ f^2(\mathbf{x}_i) \\ \vdots \end{pmatrix} \text{ and } \mathbf{F} = [\mathbf{f}_1 \quad \mathbf{f}_2 \quad \dots]. \quad (3.53)$$

When the random functions have algebraically independent (unrelated) drifts, each of them must be modelled separately. Therefore, each variable  $P$  will have a set of coefficients  $\mathbf{a}$  to model their respective drifts. To make the universal kriging equations detrend the raw weighted data and so preserve unbiasedness, the following conditions must be introduced [84]. Denoting the Kronecker delta with  $\Delta_j^i$ , there arises the following set of conditions;

$$\begin{aligned} \sum_i w_i f^j(\mathbf{x}_i) &= \Delta^1 f^j(\mathbf{x}_0) \quad \text{or} \\ [\mathbf{f}_1, \mathbf{f}_2, \dots] \mathbf{w} &= \mathbf{F} \mathbf{w} = \Delta^1 \mathbf{f}_0 \end{aligned} \quad (3.54)$$

where  $\mathbf{w}$  is a vector of the unknown weights multiplying the vector of known values  $\mathbf{p}$ . Note that the right hand side of Equation (3.54) is a zero vector for all secondary ( $i \neq 1$ ) drifts. If these constraints are imposed upon the minimisation of kriging variance in Equation (3.49), the following optimisation problem is obtained, where the drift coefficients  $\mathbf{a}$  become Lagrange multipliers, and  $\mathbf{w}$  is the vector of all weights multiplying all nodal values  $p_i$ :

$$\nabla_{\mathbf{w}}^2_{CK} = 2 \sum_j \nabla_{\mathbf{w}} \left\{ a_j \sum_i w_i f^j(\mathbf{x}_i) \right\}. \quad (3.55)$$

Just as in the derivation of the universal kriging equations (3.37), the above expression yields as many equations as weights in  $\mathbf{w}$ . The extra equations needed to solve for the Lagrange multipliers  $a_{ai}$  are furnished by the constraints in Equation (3.54), resulting in the following set of equations, in matrix form.

$$\begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{21} & \dots & \mathbf{F}^{1T} & \mathbf{0} & \dots \\ \mathbf{C}_{12} & \mathbf{C}_{22} & & \mathbf{0} & \mathbf{F}^{2T} & \\ \vdots & & \ddots & \vdots & & \ddots \\ \mathbf{F}^1 & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}^2 & & \vdots & & \vdots \\ \vdots & & & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \\ \vdots \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{c}_0^1 \\ \mathbf{c}_0^2 \\ \vdots \\ \mathbf{f}_0 \\ \mathbf{0} \\ \vdots \end{pmatrix} \quad (3.55)$$

One may solve for the weights which are then used with Equation (3.46) in the usual way to produce an estimate at  $\mathbf{x}_0$  and calculate an associated kriging variance using Equation (3.48).

### III - 6.2. Algebraically dependent drifts

The above case concerning independent drifts may be thought of as rather strange – it is difficult to imagine a ‘non-synthetic’ situation where the random component remains correlated yet the non-random components are allowed to vary independently. The other extreme case, which is perhaps more reasonable, is to consider that the random functions  $P^1, P^2, \dots$  may either share the same underlying mean or may have their means related via some linear relationship (as usual, non-linear relationships require special treatment which shall not, for simplicity, be addressed here). If the functions share the same mean, then

$$\mu(\mathbf{x}) = \sum_i a_{\mu_i} f^i(\mathbf{x}) . \quad (3.56)$$

The means  $\mu$  are still distinguished by the subscript for reasons that shall be seen later – for now it is enough to ignore the subscript and imagine that the functions  $f^{ai}$  will be the same for each random function. The unbiasedness condition that resulted in (3.54) now produces the set of constraints on the weights;

$$\sum_i w_i f^j(\mathbf{x}_i) = f^{1j}(\mathbf{x}_0) , \quad (3.57)(a)$$

or, 
$$\begin{bmatrix} \mathbf{f}_1^1, \mathbf{f}_2^1, \dots \end{bmatrix} \mathbf{w}^1 + \begin{bmatrix} \mathbf{f}_1^2, \mathbf{f}_2^2, \dots \end{bmatrix} \mathbf{w}^2 + \dots = \sum \mathbf{F} \mathbf{w} = \mathbf{f}_0^1 . \quad (3.57)$$

When the minimisation of variance is constrained by the above relations, one obtains the minimisation problem;

$$\nabla_{\mathbf{w}} \frac{\partial}{\partial C_K} = 2 \sum_j \nabla_{\mathbf{w}} \left\{ a_{*j} \sum_{i,j} w_i f^j(\mathbf{x}_i) \right\} \quad (3.58)(a)$$

which may be expressed in matrix form as

$$\begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{21} & \cdots & \mathbf{F}^{1T} \\ \mathbf{C}_{12} & \mathbf{C}_{22} & & \mathbf{F}^{2T} \\ \vdots & & \ddots & \vdots \\ \mathbf{F}^1 & \mathbf{F}^2 & \cdots & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \\ \vdots \\ \mathbf{a}_* \end{pmatrix} = \begin{pmatrix} \mathbf{c}_0^1 \\ \mathbf{c}_0^2 \\ \vdots \\ \mathbf{f}_0^1 \end{pmatrix} \quad (3.58)$$

Note that there is a common set of weights  $\mathbf{a}_*$  expressing the drift for all functions. The subscript was retained in Equation (3.56) to cater for the possibility that the drifts were related in some way. An important example of this, is when there is so-called ‘gradient’ information collected at points – for example, nodal values of temperature  $P^1$ , complemented by measurements of rate of change of temperature  $P^2$  in the direction  $\mathbf{s}$ . In this case the basis functions  $f^i$  are modified to

$$f^{2i} = \frac{d}{d\mathbf{s}} \{f^{1i}\} \quad (3.59)$$

Other relationships between the drifts can be modelled similarly. Whilst the basis functions are modified for each variable, there is still a common vector of coefficients  $\mathbf{a}_*$  tying them together.

### III - 7. Cross-covariant Structure Identification

The cross-covariance function was introduced briefly in the previous section. It is explained in more detail here, with attention to the way in which it is modelled. It is worth bearing in mind that necessarily, many of the expressions in this section actually present a more general form of the auto-covariance function, and often in this dissertation the term covariance function and indeed the following equations will refer to either function or both functions, depending on the context.

### III - 7.1. Cross-covariant statistics

Many of the definitions for the cross-covariance function are direct extensions of those encountered previously for the auto-covariance. From the definition of covariance,

$$\begin{aligned} C(\mathbf{x}, \mathbf{y}) &= E\left(\left[P(\mathbf{x}) - \bar{P}(\mathbf{x})\right]\left[P(\mathbf{y}) - \bar{P}(\mathbf{y})\right]\right) \\ &= E\left(P(\mathbf{x})P(\mathbf{y})\right) - \bar{P}(\mathbf{x})\bar{P}(\mathbf{y}) \end{aligned} \quad (3.60)$$

Note that in the above,  $\mathbf{x}$  is a point in the domain of  $P^\alpha$  and  $\mathbf{y}$  is a point in the domain of  $P^\beta$ , and these domains overlap. Just as for the auto-covariance, and for exactly the same reasons, an assumption of second order stationarity is made, thus

$$\begin{aligned} C(\mathbf{x}, \mathbf{y}) &= E\left(P(\mathbf{x})P(\mathbf{x} + \mathbf{h})\right) - \bar{P}(\mathbf{x})\bar{P}(\mathbf{x} + \mathbf{h}) \\ &= C(\mathbf{h}) \end{aligned} \quad (3.61)$$

where again

$$\mathbf{h} = \mathbf{y} - \mathbf{x} \quad \text{see (3.5)}$$

For auto-covariances the sign of  $\mathbf{h}$  did not matter as the function itself was symmetric (both  $\mathbf{x}$  and  $\mathbf{y}$  indicated point values in the same function). However for cross-covariances it is important, as the cross-covariance function is not necessarily symmetric.

$$\begin{aligned} C(\mathbf{x}, \mathbf{y}) &\neq C(\mathbf{y}, \mathbf{x}), \quad \neq \\ \therefore C(\mathbf{h}) &\neq C(-\mathbf{h}) \end{aligned} \quad (3.62)$$

A good example of how the cross-covariance may be asymmetric is the case of a spatial shift. If two random functions  $P^\alpha$  and  $P^\beta$  are in fact identical, their ‘cross’-covariance function would indeed degenerate to a simple auto-covariance function. Furthermore this function would be identical to the covariance function associated with either random function. It stands to reason therefore that if  $P^\beta$  is then offset from  $P^\alpha$  by some simple spatial shift, but otherwise remains the same function, the spatial shift will also be manifested in the cross-covariance function. The new cross-covariance function would still be identical to the original auto-covariance functions of  $P^\alpha$  and  $P^\beta$ , except that now the peak at its centre would be offset – shifted by the same amount as  $P^\beta$  was shifted from  $P^\alpha$ .

Nonetheless a symmetry condition still exists for the cross-covariance function;

$$\begin{aligned} \text{cov}(P(\mathbf{x}), P(\mathbf{y})) &= \text{cov}(P(\mathbf{y}), P(\mathbf{x})) \\ \mathbf{C}(\mathbf{h}) &= \mathbf{C}(-\mathbf{h}) \end{aligned} \quad (3.63)$$

but the indices need to be swapped. Note that the above relation means that only one cross-covariance function  $\mathbf{C}_{\alpha\beta}(\mathbf{h})$  must be determined. In Equations (3.50), (3.55) and (3.58), both  $\mathbf{C}_{\alpha\beta}(\mathbf{h})$  and  $\mathbf{C}_{\beta\alpha}(\mathbf{h})$  are used to generate the various sub-matrices, however because of Equation (3.63), the diagonally opposite sub-matrix to  $\mathbf{C}_{\alpha\beta}$  is simply its transpose, and this ensures that the matrices in these equations are symmetric.

$$\mathbf{C} = \mathbf{C}^T \quad (3.64)$$

Analogous to the auto-covariant case is the stationary cross-correlation function  $\rho_{\alpha\beta}(\mathbf{h})$ , which is the covariance function normalised this time by the standard deviations of the random functions  $P^\alpha$  and  $P^\beta$ .

$$\rho_{\alpha\beta}(\mathbf{h}) = \frac{\mathbf{C}_{\alpha\beta}(\mathbf{h})}{\sigma_\alpha \sigma_\beta} \quad (3.65)$$

This offers the same advantages as the auto-covariant case, again depending on how the standard deviations  $\sigma_\alpha$  and  $\sigma_\beta$  are calculated. There is also an analogous function to the variogram in Equation (3.38) for the cross-covariant case – called the cross-variogram.

$$\Gamma_{\alpha\beta}(\mathbf{h}) = \frac{1}{2} \text{E} \left[ (P(\mathbf{x}+\mathbf{h}) - P(\mathbf{x})) (P(\mathbf{x}+\mathbf{h}) - P(\mathbf{x})) \right] \quad (3.66)$$

However, to calculate this function realisations of both  $P^\alpha(\mathbf{x})$  and  $P^\beta(\mathbf{x})$  must be available at each data-point  $\mathbf{x}$ , which implies that the data-sets or realisations arising from the different random functions must be coincident. This is not generally the case. Unlike the cross-covariance function, the cross-variogram *is* symmetric.

$$\begin{aligned} \Gamma_{\alpha\beta}(-\mathbf{h}) &= \frac{1}{2} \text{E} \left[ (P(\mathbf{x}-\mathbf{h}) - P(\mathbf{x})) (P(\mathbf{x}-\mathbf{h}) - P(\mathbf{x})) \right] \\ &= \frac{1}{2} \text{E} \left[ (P(\mathbf{x}) - P(\mathbf{x}-\mathbf{h})) (P(\mathbf{x}) - P(\mathbf{x}-\mathbf{h})) \right] \end{aligned} \quad (3.67)(a)$$

and provided the random functions form the same lag vectors over their respective domains,

$$\begin{aligned} &= \frac{1}{2} E \left[ \left( P(\mathbf{x} + \mathbf{h}) - P(\mathbf{x}) \right) \left( P(\mathbf{x} + \mathbf{h}) - P(\mathbf{x}) \right) \right] \\ &= \Gamma(\mathbf{h}) \end{aligned} \quad (3.67)$$

Furthermore it is easy to see that the random functions  $P^\alpha$  and  $P^\beta$  can be swapped in Equation (3.66), therefore

$$\Gamma(\mathbf{h}) = \Gamma(\mathbf{h}) \quad (3.68)$$

It follows from the above that the cross-variogram is not able to pick up a pure spatial shift in the same way as the cross-covariance function. This is because it only finds the even component of the stationary covariance function in Equation (3.61). As examining spatial shifts may later be useful, and the sets of related nodal data considered in this dissertation are not necessarily coincident, this particular structural tool has not been utilised.

### III - 7.2. Linear model of coregionalisation

When the systems of equations in Equations (3.50), (3.55) and (3.58) are used to produce a cokriged estimate, there are some further restrictions on the modelled covariance functions that ensure a positive definite system of equations and non-negative modelled variance. The restriction is imposed by the coregionalisation model, for which the present author has adopted the simplest *linear* coregionalisation model.

The linear model of coregionalisation is a restriction on the sill values  $t_i$  in Equation (3.28) of the various variogram model contributions. For an allowable, positive definite model, the cross-covariance (and thus cross-variogram) functions must be composed of a linear sum of positive-definite combinations of the variogram models described in Equations (3.23). So given some models indexed by  $i$  and extending the univariate model proposed in Equation (3.28), the modelled auto and cross-covariance functions may be represented as the linear combinations

$$C(\mathbf{h}) = \sum_i s^i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right]. \quad (3.69)$$

To ensure a positive definite system of equations in Equations (3.50), (3.55) and (3.58), the matrix formed by the coefficients  $[s^i]$  must be positive definite for all models – i.e.

$$\mathbf{x}^T \begin{bmatrix} s_{11}^1 & s_{12}^1 & \cdots \\ s_{21}^1 & s_{22}^1 & \\ \vdots & & \ddots \end{bmatrix} \mathbf{x} > 0 \quad \forall \mathbf{x},$$

$$\mathbf{x}^T \begin{bmatrix} s_{11}^2 & s_{12}^2 & \cdots \\ s_{21}^2 & s_{22}^2 & \\ \vdots & & \ddots \end{bmatrix} \mathbf{x} > 0 \quad \forall \mathbf{x} \quad (3.70)$$

, ... etc.

as must the anisotropy transformation matrices  $\mathbf{T}^i$  in Equations (3.69) and (3.27).

Given that a positive definite matrix may only have zeroes on the off-diagonals, it may be concluded that if a behaviour or model is desired in a given cross-covariance model, it must also appear somewhere in the auto-covariance models of that coregionalisation model. This is reflected in Equation (3.69), where all of the models indexed by  $i$  have coefficients  $s^i$  for their auto and cross-covariant contributions to the functions  $C_{\alpha\beta}$ . The positivity constraints can be quite difficult to manage – the way in which they are handled is described in Section IV - 5.2.

For univariate kriging there was a restriction on the total value of the coefficients, outlined in Equation (3.26). In multivariate kriging the analogous relation applies, so now

$$\sum_i s^i = \sigma^2 \quad (3.71)$$

where  $\sigma^2$  is the total covariance between the random (non-drift) components of variables  $P^\alpha$  and  $P^\beta$ . In practice, this condition is relaxed somewhat simply as it is difficult to measure the total cross-covariance for non-coincident nodal data. Furthermore, it is also difficult to filter the random component to determine the covariance. So instead of an equality, it is required for the cross-covariances that;

$$\sum_i s^i \leq \quad , \quad \forall \neq \quad (3.72)$$

and a broader range of variogram models is thus admitted. This relation at least puts a sensible upper limit on the total variance of the cross-covariant components. In general, it pays to remember that whilst the estimations of total variance directly affect the modelled kriging variance, they do not affect the estimated value as they only concern scaling internal to the model. What matters is the relative shape of the covariance functions, not their scaling.

## Chapter IV - Implementation

From the outset, the aim of this thesis was to produce – at least in prototypical form – a piece of software that could in some meaningful way address the typical needs of the fluids analyst, straddling both modern experimental and computational tools. This rather ambitious aim was spurred by a commonly perceived deficiency in the analysis of spatial data; which is, data collected or generated over a field or domain. The techniques in spatial statistics described in the foregoing chapter specifically address spatial data and their comparison and interpolation. In particular, cokriging is of interest as it offers an underlying model for inter- and intra-dataset correlation, and also a means by which results can be blended.

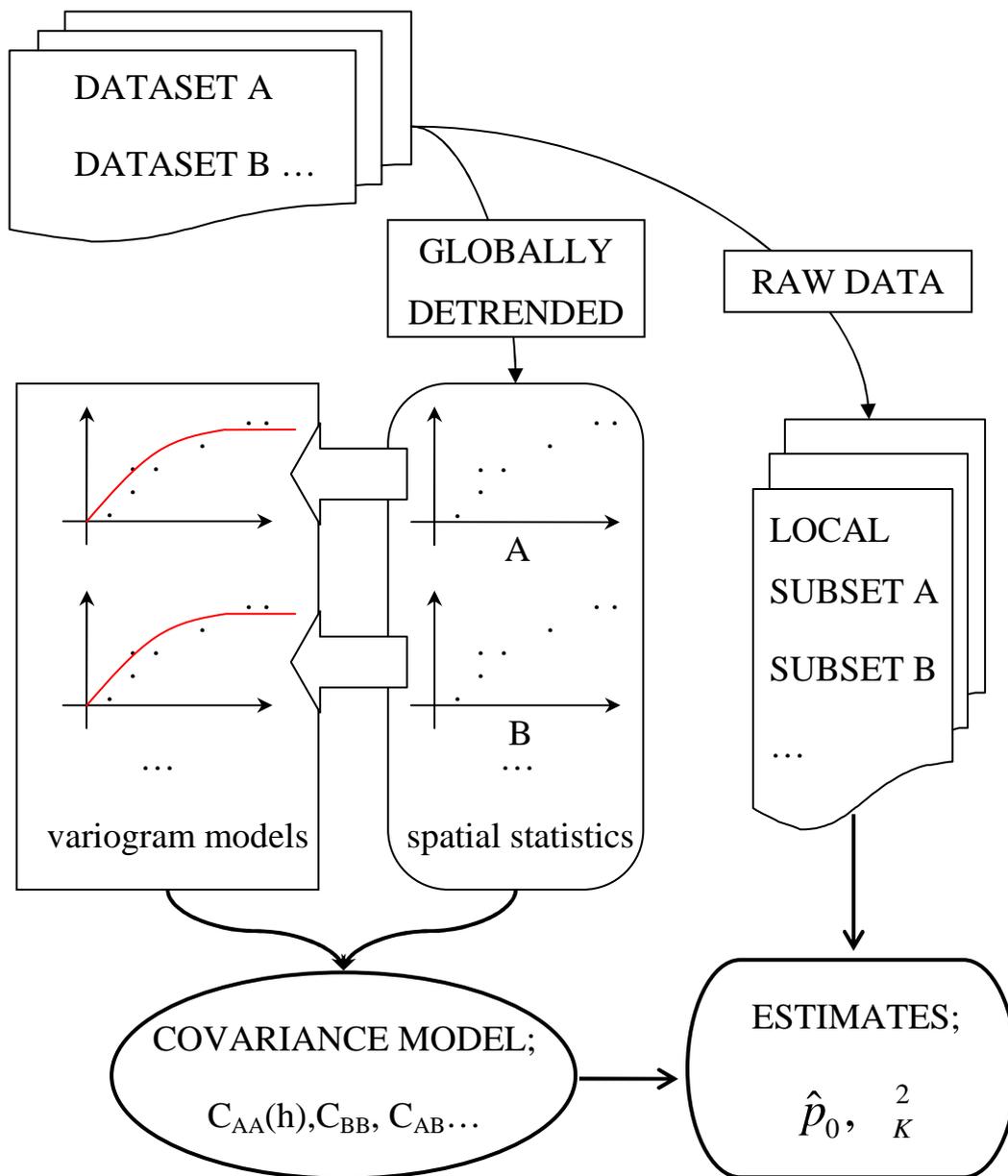
How cokriging and the supporting structure identification required for it are implemented is described in this chapter. The principal original work in this thesis was the creation of a prototype cokriging code written in FORTRAN, specifically addressing the requirements of the computational fluids dynamicist. In hindsight, it may have been easier to use a package, for example GSLIB – a free suite of geostatistical tools, but given the significantly different end application to which the original spatial statistics were being put, it was decided to code instead as it was felt that this offered greater functional flexibility. Moreover, it was also required that the

algorithms should be automated to a greater extent than was evident in a suite of tools obviously designed for use by an audience with an intimate knowledge of geostatistical theory. Although one of the attractions of geostatistics was its relative simplicity, it was unreasonable to expect those in the fluids community to become versant in an entirely unfamiliar discipline. For this reason, the code has been designed to work in an as automated, yet robust fashion as possible, sometimes at the expense of optimal theoretical performance.

Another consideration in the intended development of this software tool, was its eventual application to higher dimensional spaces than  $\mathbb{R}^2$  or  $\mathbb{R}^3$ . Whilst data is often collected over a physical domain, it is not uncommon for extra data dimensions to exist in an experimental or numerical survey. As was mentioned in the introduction and literature review, most modern engineering experiments seek to characterise behaviour over a range of parameters – in some ways different to strictly scientific experiments which address a hypothesis or theory.

## IV - 1. Overall Architecture

As previously outlined in “Chapter III - Theoretical Overview”, producing kriged and cokriged interpolations typically involves two steps – structure identification, and then subsequent estimation. Structure identification involves generating spatial statistics on the data-sets – for example the sample covariance functions and variograms introduced in Equations (3.8) and (3.39), and then from these statistics, inferring a likely form and shape of the covariance model, expressed in Equation (3.69). The estimation step involves choosing a sub-set of data local to the estimated point  $\mathbf{x}_0$ , illustrated in Figure III—6, and then generating the matrices in Equations (3.37), (3.55) or (3.58) to solve for the weights and eventually produce the estimate  $\hat{p}_0$  using Equation (3.46). The FORTRAN code ‘krige’ operates along similar lines, as illustrated in Figure IV—1; the structure identification steps are to the left on this figure, whilst the estimation steps are on the right.



**Figure IV—1: Schematic architecture of the kriging algorithm.**

Data is provided in a binary format, in twin '.kr1' and '.kr2' files. The data in these files consists of the raw nodal information comprising coordinates and values for the various datasets under examination. Significantly, to speed up searching these files for information, the data is organised into local clusters which are stored on separate binary records in the direct access file. In this way, a search for points in a given locality may be narrowed immediately to a smaller search amongst those clusters closest to it. The preparation of the raw data in this manner is detailed in Appendix C - 1. Other input information is provided in a file called 'krigin.txt',

which sets various estimation and analysis options, such as the domain of estimation or points at which estimations are required. The code can operate in several modes; “raster interpolation”, “structure identification” and “cross-validation”. The raster interpolation mode shall be explained in most detail, as it encompasses all aspects of structure identification and also estimation.

Initial structure identification involves first generating sample spatial statistics on the raw data. Auto-covariant structures such as correlograms and variograms are generated for each dataset used in the estimation, and cross-covariant statistics are generated for all pairs of datasets, to describe their interrelation. Thus for a univariate kriging, only one such structure will be generated; a correlogram or variogram, and for cokriging with two variables, three structures will be produced; two auto-covariant structures and one sample cross-covariance function. In general, for  $N$  variables there are  $N(N + 1)/2$  such sample structures.

The sample structures consist of ‘clouds’ of points that approximate the true covariance function’s surface. Naturally, these clouds are subject to statistical error: they are distorted because their estimation is based on limited information. As described in “Theoretical Overview”, a model covariance function must be fitted to these points for generality and to ensure that the point-to-point covariance relationships remain positive definite. However, there are many flexible parameters in the model proposed in Equation (3.69) – reproduced below

$$C(\mathbf{h}) = \sum_i s^i \left[ 1 - F^i \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right], \quad \text{see (3.69)}$$

such as the number and type of variogram models  $\gamma^i$ , and the coefficients that make up  $s^i$  and  $\mathbf{T}^i$ . A robust and automatic means of selecting these parameters is required. This is difficult to achieve without a human’s intuition, so it has been attempted here computationally for only a very simple model consisting of four of the variograms introduced in Equation (3.23); the nugget effect model, the spherical model, the exponential and Gaussian model. These have been selected so as to achieve some gradation of behaviour from pure noise to purely smooth behaviours.

At a first pass, each of the sample structures is considered separately. To each of these structures a model covariance function of the form in Equation (3.28) is

fitted, assaying various combinations of the three structural models and the nugget effect. In each trial the coefficients  $t_i$  and the matrices  $\mathbf{T}^i$  are adjusted to achieve a best fit, and the significant structures with large  $t_i$  values are noted. Thus at the end of the trials, one has a number of variogram models  $\gamma^i$  of various types, and as many corresponding anisotropy matrices  $\mathbf{T}^i$ . As only the significant or ‘good’ structures have been selected, there may not actually be that many variogram models from which the final model is constructed. The second part of the variogram modelling then considers all of the sample structures together to determine all of the terms in Equation (3.69) and generate the final covariance model. The transformations and models previously discovered are substituted into the form in Equation (3.69), and the coefficients  $s^i$  are adjusted so that a best fit is obtained with each sample structure formed by some  $\alpha$ - $\beta$  combination of variables. Note that when running in structure identification mode, `krige` runs only to this stage, with some modifications.

Now that all of the parameters in the covariance model (3.69) have been determined, the path is clear to produce some kind of estimate by solving one of the Equations (3.55) or (3.58) (or the other systems of equations, which are of course, sub-cases), and using Equation (3.46), viz.

$$\hat{p}_0 = \sum_{k,} w_k p_k \quad . \quad \text{see (3.46)}$$

To produce an estimate, a local window of data close to the point of estimation  $\mathbf{x}_0$  is selected. This is done for two reasons; to make the assumption of stationarity more reasonable by only considering a locally stationary sub-region, and also to reduce the size of the system of equations expressed in Equations (3.55) and (3.58). To manage the estimation in an efficient manner and avoid re-solving identical or nearly identical sets of equations, lists are compiled of proximal data-points to each estimation point. These lists are then merged if they contain similar data thus trading off the number of matrices to be solved with the size of the matrices to be solved. In this way, the weights for several estimation points can be determined from the **LU** decomposition of a single matrix – all that needs to be recalculated is the back-substitution of the left hand side vector,  $\mathbf{c}_0$ . This is a more efficient alternative to solving almost the same set of equations, several times.

Once the weights for each estimation point are determined it is a simple matter to use Equations (3.46) and (3.48) to produce the final estimate and its associated kriging variance. In the raster interpolation mode, the program krige produces a regularly spaced grid (or raster) of such estimates, output to a sequential ASCII formatted file. Visualisation of these results is then easily achieved with MATLAB, or even MS Excel. The other operational modes are described in detail in Appendix B - 1, but incorporate elements of the above estimation procedure to achieve different ends.

## IV - 2. Auto-covariant Statistics

The very first task is to generate spatial statistics that describe the continuity of the phenomenon under observation. So far, the theoretical definitions of the variogram and the covariance function in Equations (3.38) and (3.7) respectively, have been introduced without properly detailing the statistics required to estimate them. Two one-dimensional statistics were introduced in Equations (3.39) and (3.8), and these will be generalised here to estimate these functions in higher dimensions. Throughout this chapter, the statistics that estimate the true function are signified by the use of a hat; thus  $C(\mathbf{h})$  is estimated by  $\hat{C}(\mathbf{h})$  and so forth.

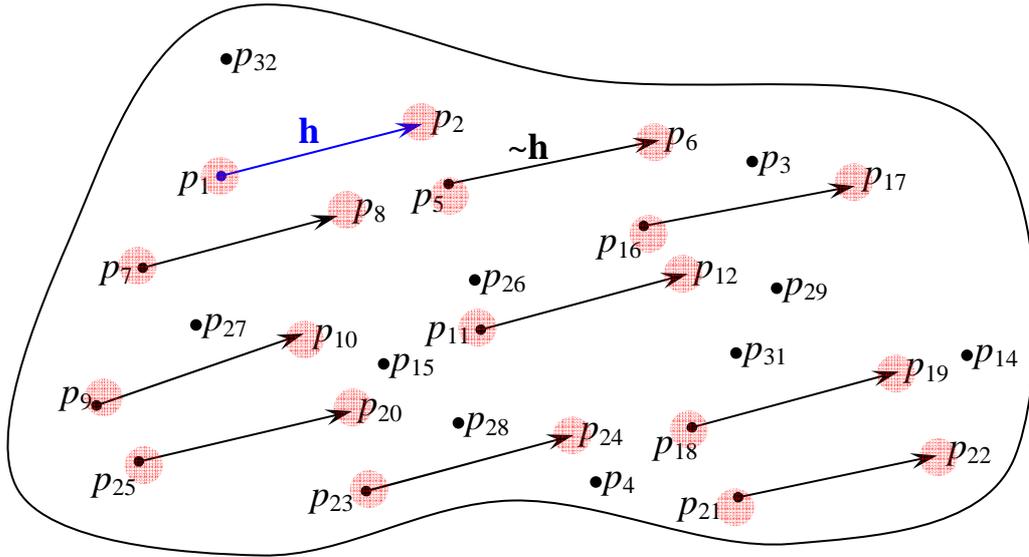
In Equation (3.2) the stationary covariance function on the random function  $P$  is defined as the expectation of the product, so that over a lag  $\mathbf{h}$

$$\text{cov}(P(\mathbf{x} + \mathbf{h}), P(\mathbf{x})) = E(P(\mathbf{x} + \mathbf{h})P(\mathbf{x})) - \mu_{+\mathbf{h}} \mu_{-\mathbf{h}} \quad (4.1)$$

As before, a realisation of  $P$  sampled at a set of nodes indexed by  $i$  is denoted by the lowercase,  $p_i$ . The above covariance function may be estimated from this set of nodal data by a statistic of the form [7]

$$\hat{C}(\mathbf{h}) = \frac{1}{N(\mathbf{h})} \sum_{(i,j) \in \mathbf{h}} p_i p_j - \mu_{+\mathbf{h}} \mu_{-\mathbf{h}} \quad (4.2)$$

where the summation term is over all pairs of data-points  $p_i$  and  $p_j$ , separated by a vector  $\mathbf{h}$ , and  $N(\mathbf{h})$  is the number of all such pairs – illustrated in Figure IV—2. The values  $\mu_{+\mathbf{h}}$  and  $\mu_{-\mathbf{h}}$  are respectively; the mean of the values located at the heads of the



**Figure IV—2: Nodal pairs separated by a given lag vector, to a given tolerance (in pink).**

lag vectors, and the mean of the values located at the tails of the lag vectors, in Figure IV—2.

$$-_{\mathbf{h}} = \frac{1}{N(\mathbf{h})} \sum_{i | \mathbf{h}_{ij} = \mathbf{h}} p_i \quad +_{\mathbf{h}} = \frac{1}{N(\mathbf{h})} \sum_{i | \mathbf{h}_{ij} = \mathbf{h}} p_j \quad (4.3)$$

Using these statistics for the mean tends to make Equation (4.2) more robust against the effects of a possible underlying drift. Such a drift would be in violation of the assumption of second order stationarity made in Equation (3.7), and could bias the overall results. Writing Equation (3.7) without this assumption,

$$\begin{aligned} \text{cov}(P(\mathbf{x} + \mathbf{h}), P(\mathbf{x})) &= E([P(\mathbf{x} + \mathbf{h}) - \langle P(\mathbf{x} + \mathbf{h}) \rangle][P(\mathbf{x}) - \langle P(\mathbf{x}) \rangle]) \\ &= E(P(\mathbf{x} + \mathbf{h})P(\mathbf{x})) - \langle P(\mathbf{x} + \mathbf{h}) \rangle \langle P(\mathbf{x}) \rangle \end{aligned} \quad (4.4)$$

and it is apparent why the means are calculated in this fashion. Furthermore when applying Equation (4.2), pairs of data-points that are within some tolerance of the vector  $\mathbf{h}$  are also averaged in the summation. This increases the number of pairs in the summation, and thus the confidence in the statistic – certainly it would be impossible otherwise to calculate this statistic if there were few or no data-points separated by the desired lag vector.

Closely related to the stationary covariance function is the correlation function (3.44). Whilst it is just a factored version of the covariance function

$$\mathbf{C}(\mathbf{h}) = \frac{\hat{\mathbf{C}}(\mathbf{h})}{2}, \quad \text{see (3.44)}$$

it is commonly estimated by the statistic [7]

$$\hat{\mathbf{C}}(\mathbf{h}) = \frac{\hat{\mathbf{C}}(\mathbf{h})}{\sqrt{\frac{1}{N(\mathbf{h})} \sum_{i|\mathbf{h}_{ij}=\mathbf{h}} (p_i - \bar{p}_{-\mathbf{h}})^2} \sqrt{\frac{1}{N(\mathbf{h})} \sum_{i|\mathbf{h}_{ij}=\mathbf{h}} (p_j - \bar{p}_{+\mathbf{h}})^2}}, \quad (4.5)$$

wherein the standard deviations in the denominator are calculated over the same points as used for the means in (4.3), namely

$$\bar{p}_{-\mathbf{h}} = \sqrt{\frac{1}{N(\mathbf{h})} \sum_{i|\mathbf{h}_{ij}=\mathbf{h}} (p_i - \bar{p}_{-\mathbf{h}})^2} \quad \bar{p}_{+\mathbf{h}} = \sqrt{\frac{1}{N(\mathbf{h})} \sum_{i|\mathbf{h}_{ij}=\mathbf{h}} (p_j - \bar{p}_{+\mathbf{h}})^2}. \quad (4.6)$$

Similar robustness against non-stationary data results when the correlation function is estimated in this way. As the relative magnitudes of the model covariance functions are more important than their overall scale, the sample correlation function expressed in Equation (4.5) has been employed to estimate the covariance function described throughout “Chapter III - Theoretical Overview”. This function can then be multiplied by a nominal estimate of the total variance of the random phenomenon – an operation which does not affect estimations.

Also used to estimate the auto-covariance function, is the variogram. The statistic is used to calculate it is

$$I(\mathbf{h}) = \frac{1}{2N(\mathbf{h})} \sum_{(i,j)|\mathbf{h}_{ij}=\mathbf{h}} (p_j - p_i)^2. \quad (4.7)$$

The summation and the function  $N(\mathbf{h})$  represent exactly the same processes as were introduced in Equation (4.2). In *krige* auto-covariant statistics are estimated using either the correlation function in Equation (4.5), or as an option the variogram calculated by Equation (4.7) above. When Equation (4.7) is used, it is renormalised and scaled to unity so that the statistics generated by it can be used by the program in exactly the same manner as those statistics generated by Equation (4.5). This is achieved by estimating some total variance  $\sigma_p^2$  and then using the relation in Equation (3.42), viz.

$$\mathbf{C}(\mathbf{h}) = \frac{\sigma_p^2}{2} - I(\mathbf{h}) \quad \text{see (3.42)}$$

to obtain the equivalent covariance function. This is then normalised by  $\frac{2}{p}$ , so that it is comparable to the correlation function.

The estimation of  $\frac{2}{p}$  and the validity of (3.42), which is only strictly correct for second order stationary phenomena, are the weak points in the foregoing scheme. The total variance  $\frac{2}{p}$  was estimated by using the data in the vector of nodal realisations  $\mathbf{p}$ , in the usual way, that is

$$\hat{\frac{2}{p}} = \sqrt{\frac{1}{N(\mathbf{h})} \sum_i (p_i - \bar{p})^2} \quad (4.8)$$

This is not highly satisfactory, as this estimate will include the variance engendered by the drift, should the data be non-stationary – compared with (4.5), this is not a very robust statistic. As a result of this difficulty and the possible non-stationarity of the data, the use of Equation (3.42) may produce apparently negative covariances. These are typically ignored as they always occur at large separations, and the covariance function filters out data at large separations from the estimation, in any case. The same practice is used in relation to the correlation function, which may also be negative, although also typically only over large separations.

The mal-effects of a drift on the estimation of total variance by Equation (4.8) are problematic, as they are also for the estimation of any of the ‘stationary’ statistics presented above. They are mitigated by initially detrending the raw data before structure identification, but this is by no means a cure-all. A more rigorous estimation of the drift and better detrending is possible, but has not been pursued in this thesis for its complexity. Interestingly, the status of the drift in kriging is quite ambiguous anyway. As it is not explicitly estimated but modelled via unbiasedness constraints, the distinction between random and non-random components can be somewhat blurred – and therein lies the flexibility of kriging estimation, according to some practitioners [7, 86].

### IV - 3. Cross-covariant Statistics

The cross-covariant spatial statistics used in this thesis centre on the cross-covariance and the cross-correlation function. As discussed before, they have been chosen as they capture spatial shifts and odd components of the covariance function, and they do not require that the multiple variables in the cokriging are collocated.

The cross-covariance between the random functions  $P^\alpha$  and  $P^\beta$  was defined in Equation (3.61) as

$$C(\mathbf{h}) = E(P(\mathbf{x})P(\mathbf{x}+\mathbf{h})) - \bar{p}_\alpha \bar{p}_\beta \quad \text{see (3.61)}$$

Where as before the lowercase  $p_i$  and  $p_j$  denote realisations of  $P^\alpha$  and  $P^\beta$ , the auto-covariance statistic in Equation (4.2) is thus extended to

$$\hat{C}(\mathbf{h}) = \frac{1}{N(\mathbf{h})} \sum_{(i,j)|\mathbf{h}_i=\mathbf{h}_j} p_i p_j - \bar{p}_\alpha \bar{p}_\beta \quad (4.9)$$

for which the calculation of the mean statistics is also analogous;

$$\bar{p}_\alpha = \frac{1}{N(\mathbf{h})} \sum_{i|\mathbf{h}_i=\mathbf{h}} p_i \quad \bar{p}_\beta = \frac{1}{N(\mathbf{h})} \sum_{j|\mathbf{h}_j=\mathbf{h}} p_j \quad (4.10)$$

Again as it is the overall, relative magnitudes and shapes of the covariance

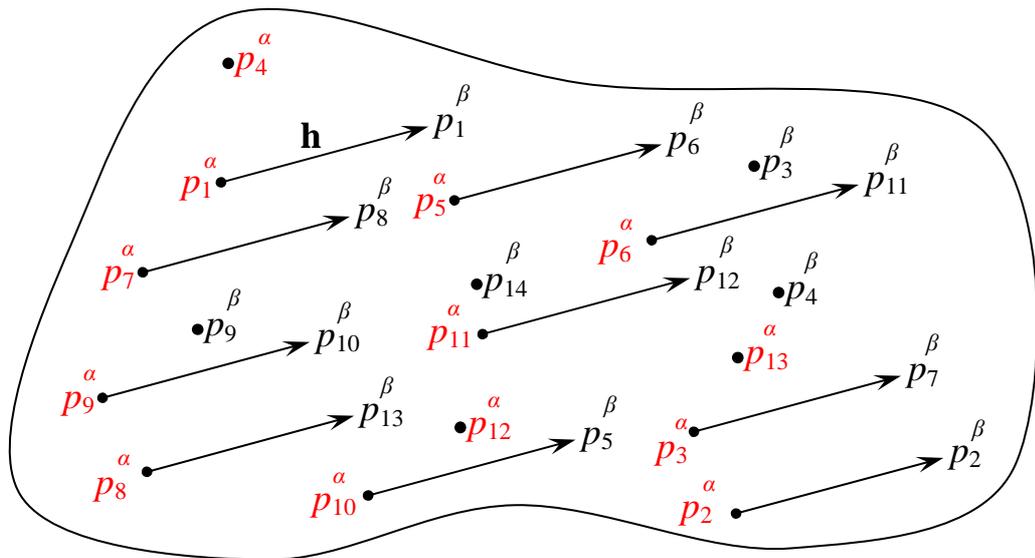


Figure IV—3: Nodal pairs of different variables separated by a given lag vector.

functions that is of interest, the normalised form – the cross-correlation function given by Equation (3.65), is used in preference to Equation (4.9). A robust statistic for this function is formed by

$$\hat{C}_{-h, +h}(\mathbf{h}) = \frac{\hat{C}(\mathbf{h})}{\sigma_{-h} \sigma_{+h}}, \quad (4.11)$$

where the normalising standard deviations are calculated as

$$\sigma_{-h} = \sqrt{\frac{1}{N(\mathbf{h})} \sum_{i|h_j=-\mathbf{h}} (p_i - \bar{p}_{-h})^2} \quad \sigma_{+h} = \sqrt{\frac{1}{N(\mathbf{h})} \sum_{i|h_j=+\mathbf{h}} (p_j - \bar{p}_{+h})^2}. \quad (4.12)$$

Again, the calculation of the statistic in this manner is to render it more robust against less-than-ideal data. Note also that as expected all of the cross-covariant statistics reduce to their auto-covariant counterparts if  $\mathbf{h} \equiv \mathbf{0}$ . Please also note that the notation for total variance estimated by Equation (4.8) needs to be extended to cover multiple datasets. The total variance of the random function  $P^\alpha$  shall be denoted as  $\sigma^2$ , to avoid superfluous subscripts.

## IV - 4. Calculation of Statistics

The calculation of the spatial statistics in the previous sections presents largely the same numerical problem, with relatively slight variations on the final computation. In the covariance functions in Equations (4.2) and (4.9), and in the variogram function in Equation (4.7) – and thus by extension in the correlation functions, there is a summation term over the same pairs of data; those separated by a vector  $\mathbf{h}$ . As previously noted, this is relaxed to include data that are only approximately separated by  $\mathbf{h}$ . Therefore one may think of the above statistics as operating on either one or both of the partners in a list of pairs of nodes. The mathematical operations on the pairs differ depending on which spatial statistic is used, but the actual list on the other hand, stays the same. In this section, the generation of said list is explained.

#### IV - 4.1. Lag generation and splitting scheme

Rather than decide on a lag vector  $\mathbf{h}$ , and then look for all pairs of points within a certain tolerance of it as illustrated in IV—2 and IV—3, the numerical problem has been solved in the ‘opposite’ direction. A list is first compiled of all the separations between points and then the list is sorted into subsets or tiles containing points separated by similar lag vectors. The lists of pairs in each tile become the lists of points required to calculate the statistics presented by Equations (4.2) through (4.12). This method is far simpler and faster than performing a search for each point numerous times, but comes at the disadvantage of a possibly large list size and corresponding sorting problem. Practically, economies may be made to mitigate both potential problems. The procedure is largely identical for both the cross- and auto-covariant statistics. Where  $\mathbf{p}^\alpha$  and  $\mathbf{p}^\beta$  are nodal realisations of random functions  $P^\alpha$

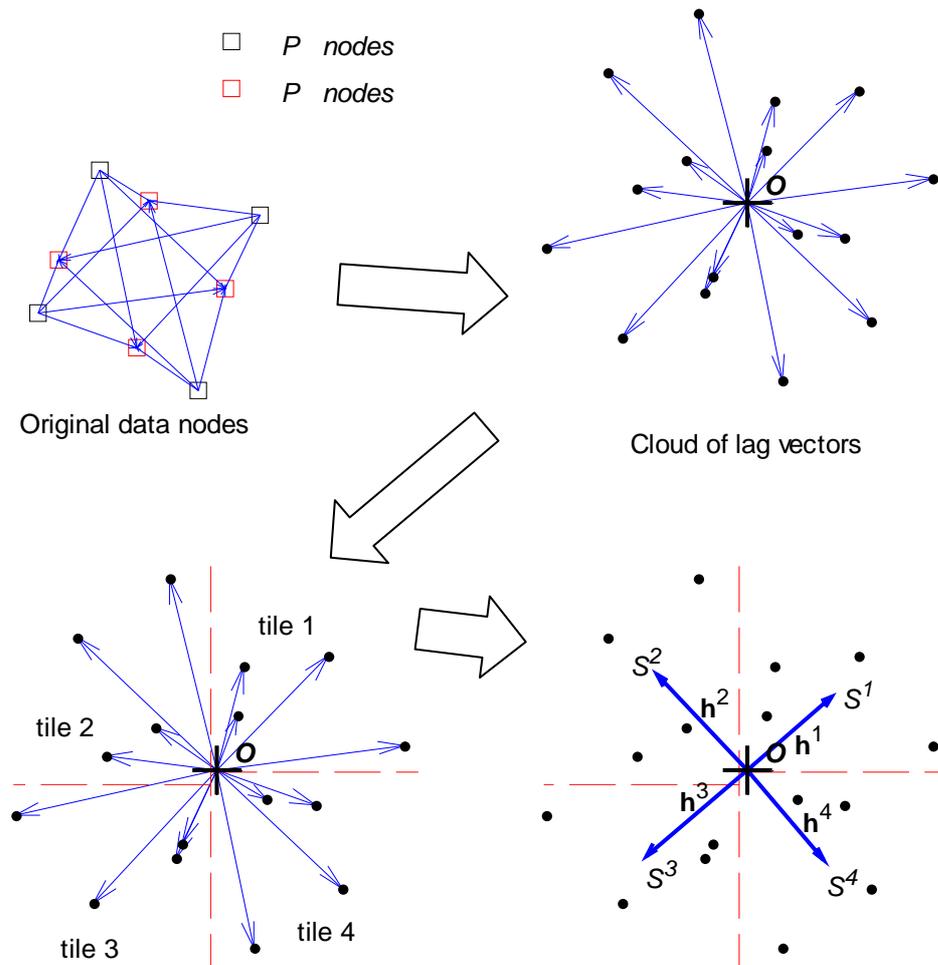


Figure IV—4: Splitting scheme for spatial statistics.

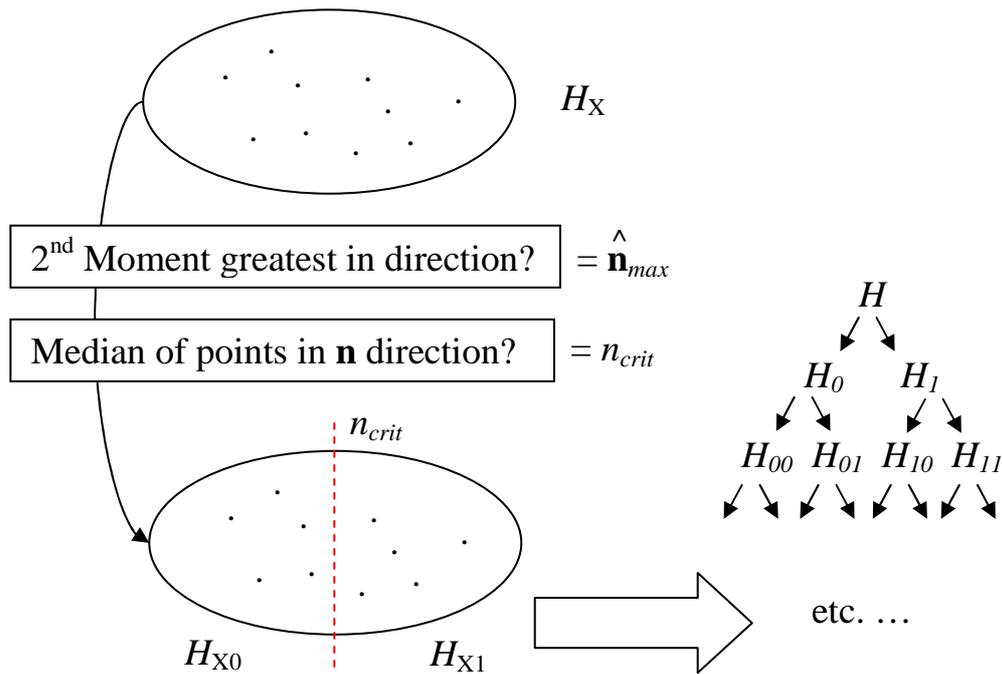


Figure IV—5: Splitting algorithm and criteria.

and  $P^\beta$  respectively, an algorithm that searches for  $P^\beta$  nodes that lie  $\mathbf{h}$  away from  $P^\alpha$  nodes should also work for the auto-covariant sub-case for which  $\mathbf{h} \equiv \mathbf{0}$ . Therefore in the following discussion, the algorithm for the more general cross-covariant case is described only and the auto-covariant case is analogous. The notation

$$\mathbf{p} : p_k \sim P(\mathbf{x}_k) . \quad \text{see (3.45)}$$

introduced in Equation (3.45) is used.

Firstly as illustrated in Figure IV—4, a list  $H$  is compiled of all possible vectors connecting some datum  $\mathbf{x}_i$  to datum  $\mathbf{x}_j$  together with the corresponding values at each end of the vector  $p_i$  and  $p_j$  :

$$H : \{ \mathbf{h}_{ij} = \mathbf{x}_i - \mathbf{x}_j ; p_i , p_j \} \quad (4.13)$$

The set  $H$  of all possible lag vectors between nodes in datasets  $\mathbf{p}^\alpha$  and  $\mathbf{p}^\beta$  is stored on disk as an intermediate file. This is then split up into subsets called *tiles*, containing similar separation vectors  $\mathbf{h}_{ij}$  and the associated pairs of values  $p_i$  and  $p_j$ . To these pairs of values, Equations (4.5), (4.7) or (4.11) are applied to generate spatial statistics whose corresponding lag vector is regarded as the average  $\mathbf{h}_{ij}$  vector

separation over the given tile. The broad algorithm is illustrated schematically in Figure IV—4 which shows how a ‘cloud’ of such vectors  $H$  may be generated and then split.

For simplicity, the splits to generate the tiles are made by a so-called ‘binary tree’ algorithm: sets of vectors are successively split into subsets along an ‘appropriate’ Cartesian coordinate direction. The original set  $H$  is split into two subsets  $H_0$  and  $H_1$ , which in turn are each split into two more, yielding  $H_{00}$ ,  $H_{01}$ ,  $H_{10}$  and  $H_{11}$  – and so on. The direction along which any given set is split is the Cartesian direction (ordinate) possessing the highest spatial spread as measured by its variance, or second moment –  $\hat{\mathbf{n}}_{\max}$  in Figure IV—5. For example, a set that is long and thin in the  $x$ -direction is split in this direction to form two sets that are subsequently less thin. In this way, the algorithm acts to produce as square tiles as possible. The critical ordinate value that splits the data is chosen to be the median value of the ordinate along which the data is split. This value –  $n_{crit}$  in Figure IV—5, divides sets into two subsets of equal size.

As a result, after  $N$  splits there will be  $2^N$  tiles with roughly equal numbers of data pairs and lag vectors in each: the global set  $H$  is broken up into subsets  $T^e$  containing lag vectors between nodal values at either end;

$$T^e : \{ \mathbf{h}_{ij}; p_i, p_j \} \quad (4.14)$$

up to a maximum number of such ‘tiles’, indexed by  $e$ . Then for each tile, an average vector  $\bar{\mathbf{h}}^e$  is calculated from the lag vectors therein;

$$\bar{\mathbf{h}}^e = \frac{1}{N(T^e)} \sum_{(i,j) \in T^e} \mathbf{h}_{ij} \quad (4.15)$$

where  $N(T^e)$  is the number of data pairs in  $T^e$ , and a spatial statistic – one of Equations (4.5), (4.7) or (4.11) as appropriate – is calculated using the nodal values  $\{ p_i, p_j \}$ . The cross or auto-covariant statistic generated over each tile  $T^e$  is denoted as  $S^e$ , regardless of the equation used to generate it – once it has been calculated and normalised, it will be used in the same fashion anyway. Thus finally as seen in Figure IV—4, a set of points  $\bar{\mathbf{h}}^e$  with which spatial statistics  $S^e$  can be associated, is generated.

$$\{\bar{\mathbf{h}}^e; S^e\} \quad (4.16)$$

Because of the splitting procedure outlined above, the number of data pairs and lag vectors in each tile is roughly equal. Therefore, the statistical support for each of the sample points  $S^e$  is also roughly equal, which has the effect of smoothing out irregularities and minimising the effects of outlying data.

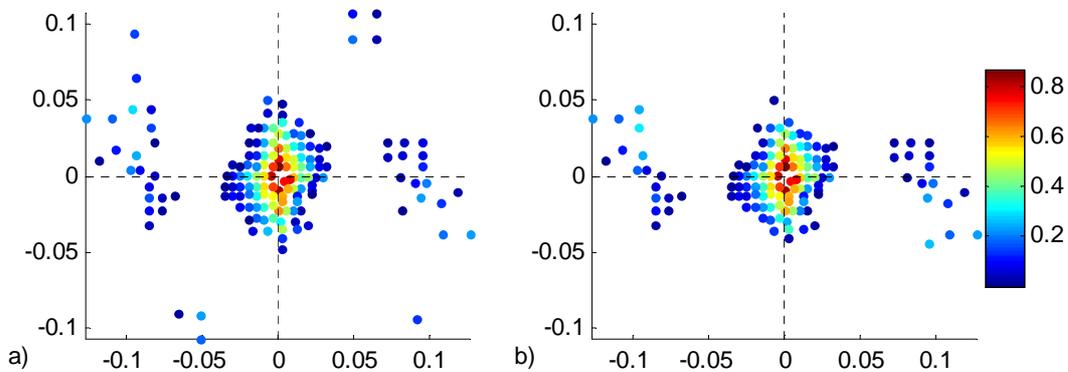
The above algorithm was chosen for its simplicity rather than performance. If one is prepared to admit irregularly sized tiles many simpler alternatives exist. However, it was judged that this would also require a more hands-on approach to tidying up the spatial statistics afterwards. Due to its potential size  $H$  must be stored on disk, which means that whatever splitting algorithm is employed, an effective input/output scheme must be devised, which complicates the programming task. This and possible alternatives to this scheme are discussed in Appendix E - Program Notes.

#### IV - 4.2. Spatial statistics

When  $\alpha \equiv \beta$ , either the auto-correlation in Equation (4.5) or a renormalised version of the variogram in Equation (4.7) can be used, as initially specified by the user in the program input arguments (see Appendix B - 1). Thus  $S^e$  is calculated as

$$S^e = 1 - \frac{\hat{I}(\bar{\mathbf{h}}^e)}{\hat{\sigma}^2} \quad \text{or} \quad S^e = \hat{\gamma}(\bar{\mathbf{h}}^e) \quad (4.17)$$

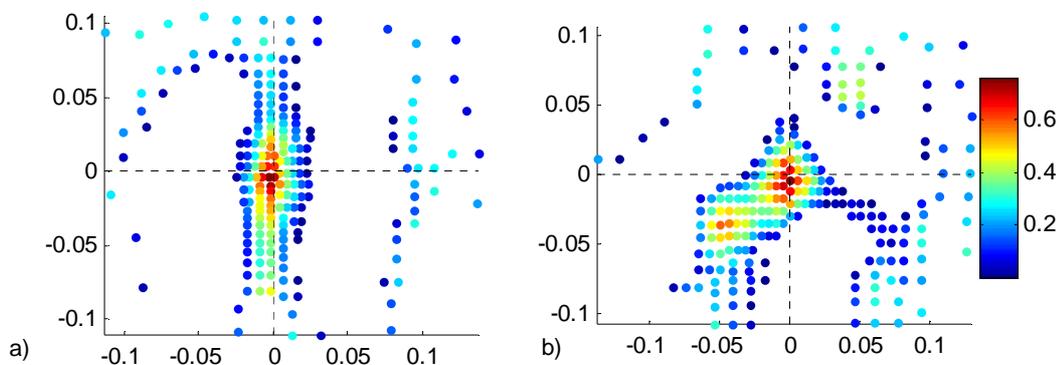
as desired, where  $\hat{I}$  is calculated on a realisation of the stationary random function



**Figure IV—6: Examples of sample auto-covariant structures; (a) correlation function, (b) normalised variogram.**

$P^\alpha$ , whose variance is estimated by Equation (4.8). The pairs of points over which the summation terms in Equations (4.7) and (4.2) – and thus by extension Equation (4.11), are evaluated, are supplied by the data pairs in the tile  $T^e$ . When the correlation statistic is used, the terms  $\mu_{-h}$  and  $\sigma_{-h}$  are calculated on the points  $p_i$  at the  $-h$  end of the pairings in  $T^e$ , and  $\mu_{+h}$  and  $\sigma_{+h}$  are calculated on the points  $p_j$  at the  $+h$  end. Note also that Equation (4.17) will always produce an auto-covariant statistic between zero and one, hence it is called a normalised covariance. When  $\alpha \neq \beta$ , exclusively the cross-correlation in Equation (4.11) has been used to provide the statistic  $S^e$ , which will consequently range between +1 and –1. In exactly the same fashion as for the auto-covariant statistics, the summation in this equation encompasses the members of the given tile  $T^e$ , and the terms  $\mu_{-h}$ ,  $\sigma_{-h}$  and  $\mu_{+h}$ ,  $\sigma_{+h}$  are calculated over the  $P^\alpha$  and  $P^\beta$  data respectively.

A sample auto-correlation function (Equation (4.5)) generated in the above manner over a two-dimensional domain, is shown in IV—6(a). This function has a clear peak at the origin, which decreases away from the origin – proximal points display high covariance, whereas distant points have little or no covariance. It is possible to add another point to this graph, exactly at the origin and equal to the total variance  $\sigma^2$ , because of the relation in Equation (3.10). However, whether this point is added or not depends on how the nugget effect is to be modelled and interpreted, and also the uncertainty (as previously noted) in estimating  $\sigma^2$ . Figure IV—6(b) is an example of a variogram as normalised by Equation (4.17). In this case it is largely similar, but situations can arise when the variogram provides the better spatial



**Figure IV—7: Examples of sample cross-covariant structures; (a) ‘good’ correlation, (b) distorted correlation.**

description – which is why it may be used optionally instead of the correlation function in program `krige`.

Cross-covariant structures are similar to the auto-covariant structures, but generally display some level of distortion depending on the nature of the relationship between the datasets. Two such structures are shown in IV—7: note that they are not symmetric, although the cross-correlation in part (a) comes close as the datasets from which it was generated are very much alike. Naturally, as the  $P^\alpha$  and  $P^\beta$  data approach parity, the cross-covariance function should approach the auto-covariance function of either data, which will be symmetric. A common structure that is spatially shifted, between the  $P^\alpha$  and  $P^\beta$  data will manifest itself in the cross-covariance function as a peak that is shifted from the origin by a similar amount, as might be the case for the smaller peak in the third quadrant of IV—7(b). As the two datasets become more ‘scrambled’ in various directions, the peak will become smudged out and blunted to a greater extent, also as seen in Figure IV—7(b). Consequently, this usually makes it more difficult to fit a theoretical covariance model, which may develop only poor correlations between the data.

It was mentioned that unexpected or possibly outlying spatial statistics are pruned from the raw set of statistics produced by the splitting procedure. These points almost always occur away from the origin of the covariance functions. In this region, typically beyond the sill should it exist, the statistics are particularly susceptible to biases in the sampling and the data. This is primarily because relatively fewer large separations exist amongst a set of nodal data, thus the statistics calculated from the tiles  $T^e$  are less localised. In Figure IV—6 there are no negative points as all the negative auto-covariant statistics are removed from consideration. Furthermore, the statistics are sorted by their distance to the origin  $\|\bar{\mathbf{h}}^e\|$  and their magnitude  $S^e$ . If any of the largest 33<sup>rd</sup> percentile of values  $S^e$  appear in the largest 33<sup>rd</sup> percentile of distance to the origin  $\|\bar{\mathbf{h}}^e\|$ , they are also removed, as one would not usually expect to find them there. This is obviously in some senses “leading the witness” – one finds what one is looking for; correlation – and it would be preferable to examine the raw statistics visually, but this runs contrary to the aim of creating a closed-form tool.

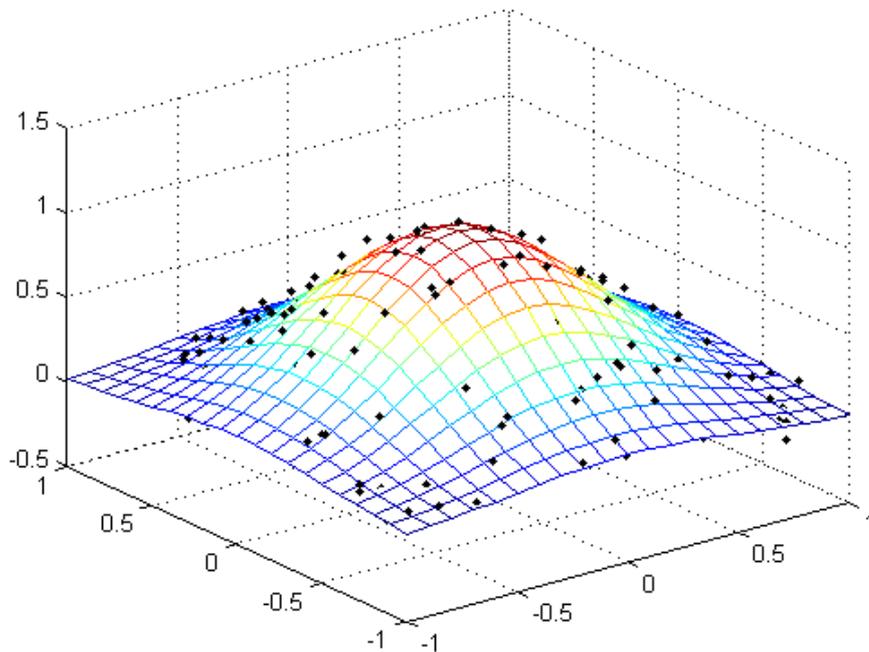
In summary, the above algorithm is applied to all possible variable combinations  $\alpha$  and  $\beta$  in the cokriging to produce sets of spatial statistics pertaining to the normalised auto- and cross-covariance functions. To distinguish between the statistics generated between different pairs of cokriging variables, the normalised covariance function between variables  $P^\alpha$  and  $P^\beta$  will henceforth be notated thus:

$$\{ \bar{\mathbf{h}}^e ; S^e \} \quad (4.18)$$

where again,  $e$  indexes the points. If just a kriging with one variable is desired then there will simply be one such set of statistics, but if a cokriging between  $N$  variables is desired then there will be  $N(N+1)/2$  such sets. Because of the symmetry relation between the cross-covariance functions expressed in Equation (3.63), the function  $C_{\beta\alpha}(\mathbf{h})$  may be determined from  $C_{\alpha\beta}(\mathbf{h})$ : therefore, statistics of both functions need not be calculated.

## IV - 5. Fitting a Covariance Model

There are two parts to fitting a model for covariance, expressed by Equation (3.69): firstly, base models are generated by considering the  $N(N+1)/2$  sample



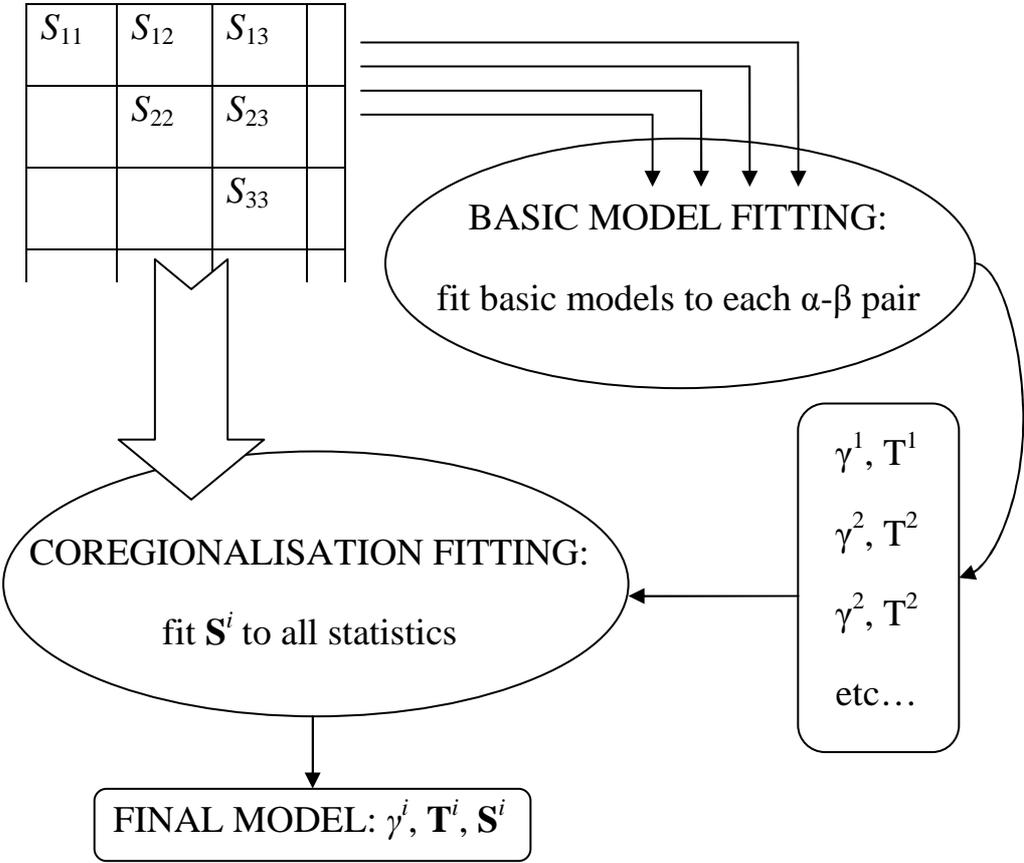
**Figure IV—8: Fitting a Gaussian function to sample points.**

covariance functions separately, and then the spatial statistics are all considered concurrently to generate the complete covariance model by fitting them to all of the statistics, respecting the linear model of coregionalisation. In both procedures a non-linear, least-squares fit of the model functions has been pursued, largely for simplicity. The present author has used the Levenberg-Marquardt algorithm [65, 66] in either case to achieve this, and a brief description of it is offered below. A more thorough explanation of the algorithm and the choice of its performance parameters can be found in Appendix C - 2.

The Levenberg-Marquardt algorithm is a non-linear least squares algorithm that finds the set of parameters that reduces the squared difference between a function (response surface) and a set of ‘known’ points. Thus there is a function

$$g(\mathbf{x}, \mathbf{r}) \quad \mathbf{x} \in \mathbb{R}^{2,3}, \quad \mathbf{r} \in \mathbb{R}^N \tag{4.19}$$

**SPATIAL STATISTICS**



**Figure IV—9: Generating the final covariance model from the spatial statistics.**

in which there are some parameters  $\mathbf{r}$  that are to be adjusted so that the least squares difference to a set of points  $\{\mathbf{x}_i; G_i\}$  is minimised. Where  $\mathbf{r}'$  is the optimal set of values, the optimisation problem

$$\mathbf{r}' = \min_{\mathbf{r} \in \mathbb{R}^N} \left\{ \sum_k [G_k - g(\mathbf{x}_k; \mathbf{r})]^2 \right\} \quad (4.20)$$

is solved. Because the function  $g(\cdot)$  is possibly non-linear in  $\mathbf{r}$ , the algorithm drives down the sum-of-squares iteratively. Essentially the function  $g(\cdot)$  is linearised around the current value of  $\mathbf{r}_i$ , and the *linear* solution to the least squares problem  $\mathbf{r}_{i+1}$ , provides the next point at which the function is linearised and approximately solved. Thus the sequence  $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 \dots$  will eventually converge upon the solution, should one exist.

The function  $g(\mathbf{x}, \mathbf{r})$  is linearised using the Jacobian matrix  $\mathbf{J}$ , given by

$$\mathbf{J} = [J_{ij}] = \frac{\partial g(\mathbf{x}_i, r_j)}{\partial r_j} \quad (4.21)$$

which evaluated at a particular set of parameters  $\mathbf{r}_i$ , is  $\mathbf{J}_i$ . The residuals  $\mathbf{R}$  (differences) in Equation (4.20) are written as

$$[G_k - g(\mathbf{x}_k, \mathbf{r})] \equiv [\mathbf{G} - \mathbf{g}] \equiv \mathbf{R} \quad (4.22)$$

and the residuals for some particular parameters  $\mathbf{r}_i$ , are  $[\mathbf{G} - \mathbf{g}_i]$  or  $\mathbf{R}_i$ . So for an iteration where the parameters are equal to  $\mathbf{r}_i$

$$[\mathbf{J}_i^T \mathbf{J}_i - \mathbf{I}] \Delta \mathbf{r}_i = \mathbf{J}_i^T [\mathbf{G} - \mathbf{g}_i] \quad (4.23)$$

may be solved for  $\delta \mathbf{r}_i$ , the change in the parameter values that minimises the linearised problem. Therefore the next iterate is:

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \Delta \mathbf{r}_i \quad (4.24)$$

which yields  $\mathbf{J}_{i+1}$  and  $\mathbf{r}_{i+1}$ , which may be used with Equation (4.23) to solve for the next step, and so forth.

The coefficient  $\kappa$  in Equation (4.23) is what allows the Levenberg-Marquardt algorithm to interpolate between a Gauss-Newton step and a steepest descent step. When it is close to zero, the algorithm approaches a second order Gauss-Newton

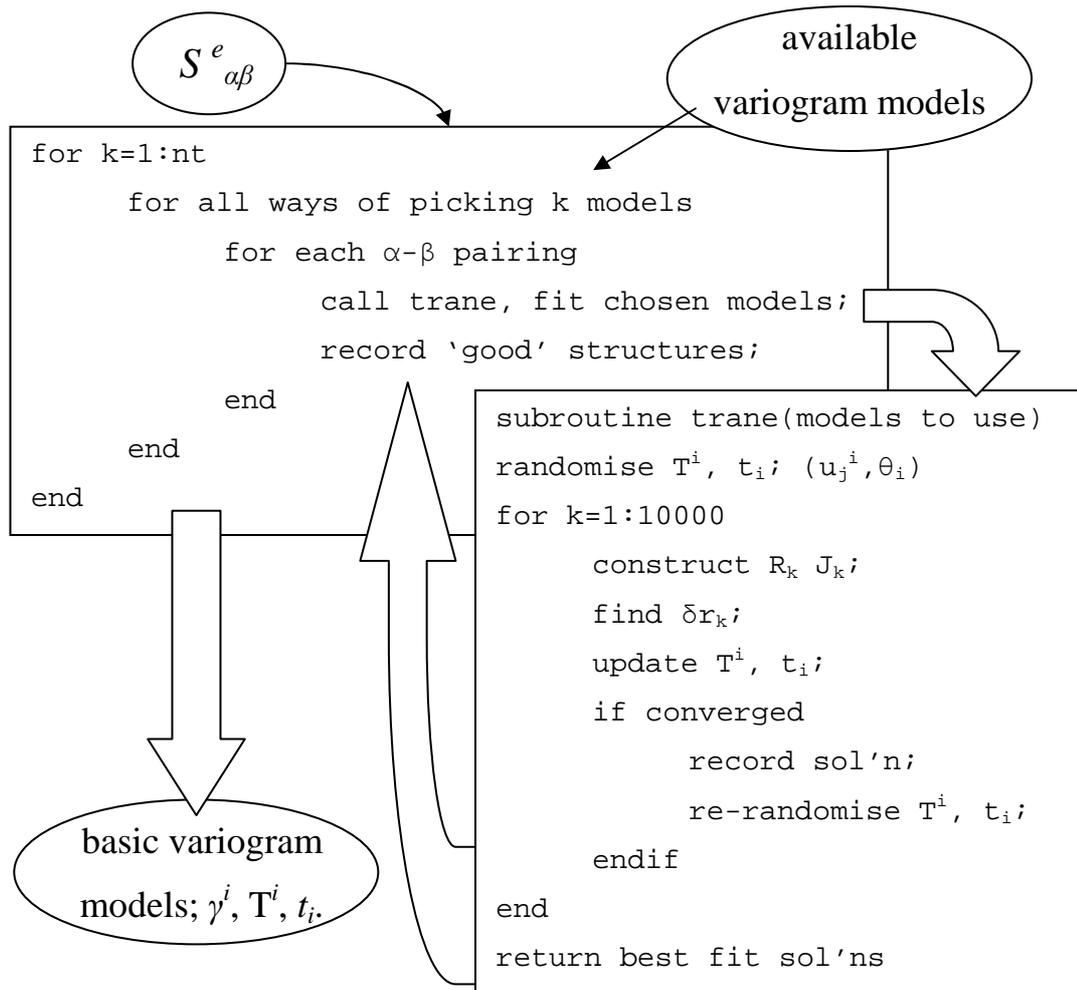
minimisation algorithm but when it is large the algorithm approaches a steepest-descent minimisation scheme. This offers the possibility both of speed and robustness, depending on how  $\kappa$  is tuned throughout the iterative solution. When the optimising function is well behaved or approaches the solution, the more aggressive Gauss-Newton method is employed for speed, but the algorithm is detuned where it encounters badly conditioned problems – providing better robustness than Gauss-Newton alone.

The Levenberg-Marquardt algorithm is kept in a module called `optimisation`, and it is used in two parts of the program; both routines in module `krige`, called `trane` and `koreg`. A complete explanation of the algorithm and the way in which  $\kappa$  is tuned is offered in Appendix C - 2. The particular objective functions, residual vectors, Jacobians and relevant constraints used in its application are described in the following sections. A flow chart describing how these sections are linked is provided in Figure IV—9.

#### IV - 5.1. Basic variogram model fitting

The subroutine `trane` uses the optimisation routines to fit a particular configuration of variogram models to a set of sample points  $\{\bar{\mathbf{h}}^e; S^e\}$  on the auto or cross-correlation function, or correlogram. These spatial statistics are normalised to unit magnitude at the maximum values of the function. However at this stage of structure identification, their scaling is not important.

To create a complete picture of the covariance function represented by Equation (3.69), one must first decide which basic variogram models  $\gamma^i$  to use and their respective anisotropies  $\mathbf{T}^i$ . To discover these a univariate covariance model, presented in Equation (3.28), is fitted to the points in  $\{\bar{\mathbf{h}}^e; S^e\}$  for each  $\alpha$ - $\beta$  pairing separately, for various combinations of the variogram models in Equations (3.23)a-f). Significant models, for which  $|t_i|$  is considered large are retained, along with their anisotropies. Therefore, this stage of structure identification is only a means of obtaining structures  $\gamma^i$  and  $\mathbf{T}^i$ . This process is illustrated by the flow diagram and pseudo code in Figure IV—10, where a preliminary piece of code generates



**Figure IV—10: Pseudo code and flow chart for basic model identification.**

combinations of basic variogram models, which a second piece of code uses to perform a fitting using the Levenberg-Marquardt algorithm.

The optimisation algorithm represented in the second block of code (bottom right) in Figure IV—10 is the more computationally and mathematically complex procedure. In short, the aim is to fit the function (3.28), namely

$$C(\mathbf{h}) = \sum_i t_i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right] \quad , \quad t_i > 0 \quad \text{see (3.28)}$$

to the points  $\{\bar{\mathbf{h}}^e; S^e\}$  – the  $\alpha$ - $\beta$  pairing that generates them is largely unimportant at this stage. Consider that the models  $\gamma^i$  have already been selected, and what remains to be determined are the matrices  $\mathbf{T}^i$  and the coefficients  $t^i$ . Note also that as this may also be fitted to cross-covariant statistics, it is not required (yet) that  $t_i > 0$ . The squared error function is calculated as

$$E_{rr} = \sum_i [S^e - C(\mathbf{h}^e)]^2 \quad (4.25)$$

As the total variance is scaled to unity, there is also the sum

$$\sum_i t_i = 1 \quad (4.26)$$

to be observed as a constraint. However, for the purposes of fitting a response surface to generate basic models, it is convenient to neglect the nugget effect model – effectively modelling it implicitly. Therefore instead of the above relation, the relation

$$\sum_i t_i \leq 1 \quad (4.27)$$

is used instead. After the model variograms are fitted and the coefficients  $t_i$  and  $\mathbf{T}^i$  in Equation (3.28) are determined, the contribution of the nugget effect model is determined by the difference

$$t_{nugget} = 1 - \sum_i t_i \quad (4.28)$$

Additionally, to avoid massive cancellations, which are unrealistic, it is also required that

$$|t_i| \leq 1 . \quad (4.29)$$

The existence of negative structures in auto-correlation functions is also unrealistic, so as a further constraint on the  $t_i$  parameters, it is required *additionally* for auto-covariant structures that

$$t_i > 0 . \quad (4.30)$$

The above requirements change slightly for the cross-covariant statistics for whom  $\alpha \neq \beta$ , as these may exhibit negative correlation. In spite of this, the same basic model, presented in Equation (3.28) is used, except that now the relation

$$-1 \leq \sum_i t_i \leq 1 \quad (4.31)$$

is used instead of the relation in Equation (4.27). Instead of allowing the coefficients to become negative in Equation (3.28), special *negative* models are introduced;  $(I^{\bar{r}} - 1)$  for every  $(1 - I^{\bar{r}})$ , to model possibly negative correlation.

Finally, the spatial statistics  $\{\bar{\mathbf{h}}^e; S^e\}$  are to be fitted to the model in Equation (3.28) which is the parametric representation of the actual function these statistics approximate. In doing this, it is required that the matrices  $\mathbf{T}^i$  are positive definite, that the constraints in Equations (4.28) through (4.30) are observed, and that the appropriate constraint in Equation (4.27) or (4.31) is applied, for auto or cross-covariant statistics respectively.

All of these constraints are achieved by the following sleights. Firstly the  $N \times N$  matrices  $\mathbf{T}^i$  are re-parameterised as the sum of  $N = 2$  or  $N = 3$  outer products of vectors in  $\mathbb{R}^N$ ;

$$\mathbf{T}^i = \sum_k \mathbf{v}_k^i \mathbf{v}_k^{i\text{T}} \quad (4.32)(a)$$

or

$$T_{mn}^i = \sum_k v_{mk}^i v_{nk}^i \quad (4.32)$$

By using the coefficients in  $\mathbf{v}_j^i$  as the unknowns in the optimisation problem instead of the raw coefficients of  $\mathbf{T}^i$ , an unconstrained minimisation may be essayed. This removes the need to impose explicit constraints on the coefficients of  $\mathbf{T}^i$  – a complex and messy problem to code. Furthermore, the coefficients  $t_i$  may be represented thus;

$$t_i = \frac{\exp(\theta_i)}{1 + \sum_j \exp(\theta_j)} \quad (4.33)$$

This parameterisation automatically satisfies all the constraints in Equations (4.27) through (4.30). Again one may solve the unconstrained minimisation problem over  $\theta_i$ , in place of a tricky constrained minimisation over  $t_i$ .

It was mentioned that to model negative behaviours present in cross-correlograms, special variogram models that are “upside-down” are introduced. If there is a model  $(1 - I^{\bar{r}})$  in the linear combination in Equation (3.28), its negative counterpart  $(I^{\bar{r}} - 1)$  is also used. One may then use two sets of coefficients  $t_i^+$  and  $t_i^-$ , and two anisotropy matrices  $\mathbf{T}_i^+$  and  $\mathbf{T}_i^-$ , pertaining to  $(1 - I^{\bar{r}})$  and  $(I^{\bar{r}} - 1)$

respectively, and solve the unconstrained minimisation problem over  $\theta_i^+$  and  $\theta_i^-$  where;

$$t_i^+ = \frac{\exp(\theta_i^+)}{1 + \sum_j \exp(\theta_j^+)} \quad , \quad t_i^- = \frac{\exp(\theta_i^-)}{1 + \sum_j \exp(\theta_j^-)} \quad (4.34)$$

Note that  $\theta_i^+$  and  $\theta_i^-$  are independent sets of variables: their derivatives do not interact.

At this point the original constrained minimisation has been converted into an unconstrained minimisation over the domain of the pseudo-vector formed by the various  $\mathbf{v}_j^i$  and  $\theta_i$ . Equation (3.28) may be rewritten over this new domain as

$$\mathbf{C}(\mathbf{h}) = \sum_i \left\{ \frac{\exp(\theta_i)}{1 + \sum_j \exp(\theta_j)} \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}} \right) \right] \right\} \quad (4.35)$$

For the purposes of the Levenberg-Marquardt algorithm, the Jacobian and the residual vector are required in terms of the variables in the minimisation. These variables are arranged into a pseudo-vector of parameters  $\mathbf{r}$ . Where  $K$  is the number of models,  $N$  is the dimensionality of the kriging and  $\mathbf{v}_j^i \in \mathbb{R}^N$

$$\mathbf{r}^T \equiv \left[ \mathbf{v}_1^{1T} \quad \dots \quad \mathbf{v}_N^{1T}, \quad \mathbf{v}_1^{2T} \quad \dots \quad \mathbf{v}_N^{2T}, \quad \dots \quad \mathbf{v}_N^{KT}, \quad 1 \quad \dots \quad N \right] \quad (4.36)$$

The vector of residuals, indexed by  $e$  is simply

$$\mathbf{R} = [\mathbf{F} - \mathbf{f}] \equiv [S^e - \mathbf{C}(\mathbf{h}^e)] \quad (4.37)$$

and the Jacobian  $\mathbf{J}$  may be calculated with respect to the variable  $v_{ab}^c$  as

$$\begin{aligned} \frac{\partial \mathbf{C}(\mathbf{h}^e)}{\partial v_{ab}^c} &= \frac{\partial}{\partial v_{ab}^c} \left\{ \sum_i \left\{ t_i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}^e} \right) \right] \right\} \right\} \\ &= \frac{-t_c I^{*c} \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \right) h_a^e v_b^c \mathbf{h}^e}{\sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e}} \end{aligned} \quad (4.38)$$

the full derivation of which is presented in Appendix D - 6. Note that in Equation (4.38) the function  $\gamma'(h)$  is just the derivative of  $\gamma(h)$  with respect to  $h$ . The

derivatives with respect to the coefficients  $\theta_i$  are also part of the Jacobian, and are calculated by

$$\begin{aligned}
\frac{\partial C(\mathbf{h}^e)}{\partial \theta_d} &= \frac{\partial}{\partial \theta_d} \sum_i \left\{ \frac{\exp(\theta_i)}{1 + \sum_j \exp(\theta_j)} \left[ 1 - I^i \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}^e} \right) \right] \right\} \\
&= \frac{\exp(\theta_d)}{1 + \sum_j \exp(\theta_j)} \left[ 1 - I^d \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^d \mathbf{v}_k^{dT}) \mathbf{h}^e} \right) \right] + \dots \\
&\quad - \sum_i \frac{\exp(\theta_i) \exp(\theta_d)}{\left( 1 + \sum_j \exp(\theta_j) \right)^2} \left[ 1 - I^i \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}^e} \right) \right]
\end{aligned} \tag{4.39}$$

Note that in the foregoing expressions, there are many terms and expressions that are common. In the code these are generally calculated once only, and re-used as they are required – the only exceptions being those quantities whose storage would entail such heavy memory usage as to render this approach impractical. Further information is included in Appendix E - Program Notes.

To perform the above minimisation, one must first decide on the set of models in Equation (4.35) to use – represented schematically as the first (top left) block of pseudo code in Figure IV—10. The number and type of variogram models to use in the fitting is a relatively arbitrary decision, which in general will depend on the sorts of behaviours one might expect to see in the data. In order to cover a range of behaviours that are typical, various combinations of the spherical, exponential and Gaussian models in Equation (3.23) have been used. However, adding models to the base program is designed to be quite simple. The model combinations are chosen in the following manner. Firstly, there exists a parameter  $n_t$  that sets an upper limit on the number of variogram models to be fitted at any one time. Then, basic variogram model fitting is performed on all possible ways of choosing *with replacement* up to  $n_t$  models. Thus should  $n_t = 2$ , the fit would be made with;  $1 \times \gamma_S$ ,  $1 \times \gamma_E$ ,  $1 \times \gamma_G$ ,  $2 \times \gamma_S$ ,  $2 \times \gamma_E$ ,  $2 \times \gamma_G$ ,  $1 \times \gamma_S$  and  $1 \times \gamma_E$ ,  $1 \times \gamma_G$  and  $1 \times \gamma_S$ , and  $1 \times \gamma_E$  and  $1 \times \gamma_G$ . For absolute simplicity it is possible to use  $n_t = 1$ , and simply fit each variogram model in turn – indeed, this is most usually what happens. Generally, adding ever more combinations of

models to the mix only has the effect of increasing run-time, and producing unrealistically ‘over-fit’ models: simplicity is therefore the best policy.

Finally, the dominant models are noted for use in the final covariance model. When only one model  $\gamma^i$  is used, it and its anisotropy matrix are retained by default. However, when a combination of models (as described above) is used just the models with the largest coefficients  $|t_i|$  are retained.

#### IV - 5.2. Fitting the linear model of coregionalisation

The variogram fitting above is used to generate the anisotropies  $\mathbf{T}^i$ , which are associated with particular variogram models  $\gamma_i$ . The final variogram model for the cross and auto-covariances is then constructed from a sum of positive definite linear combinations of these models. This final model must observe the linear model of coregionalisation which is described in Section III - 7.2, and implemented in the routine `koreg`.

The coregionalisation model is described by the matrices of sill values  $[s^i]$  or  $\mathbf{S}^i$ , where  $i$  indexes the set of variogram models used – at this point the basic models  $\mathbf{T}^i$  and  $\gamma_i$  have been generated, but the matrices  $\mathbf{S}^i$  still need to be identified; the basic model’s contributions to the auto and cross-covariance functions. Again this is a function fitting problem wherein the sets of sample statistics estimating the auto and cross-covariant structures  $\{\mathbf{h}^e ; S^e\}$ , must be approximated by response surfaces  $C_{\alpha\beta}(\mathbf{h})$ , now of the form presented in Equation (3.69). Again the Levenberg-Marquardt algorithm in the module `optimisation` is used to perform the fitting, except that the models and matrices  $\gamma^i$  and  $\mathbf{T}^i$  are now known – from here on, these are no longer modified.

As remarked previously, a multivariate kriging in  $N$  variables produces  $N(N+1)/2$  sets of statistics  $\{\mathbf{h}^e ; S^e\}$  because of the symmetry relation in Equation (3.63). Therefore,  $N(N+1)/2$  response surfaces are sought to fit these statistics concurrently, by ascertaining the matrices  $\mathbf{S}^i \in \mathbb{R}^{N \times N}$  in Equation (3.69)

$$\mathbf{C}(\mathbf{h}) = \sum_i s^i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right]. \quad \text{see (3.69)}$$

The sum-of-squares error function is formulated so that the error contributions of the various  $\alpha$ - $\beta$  pairs are normalised by the total *possible* (co-)variance  $\sigma_\alpha\sigma_\beta$  ; i.e. to unity for the auto-covariant parts.

$$E_{rr} = \sum_{e, \dots} \left\{ S^e - \frac{1}{\sum_i s^i} \left[ 1 - I^* \left( \sqrt{\mathbf{h}^e \text{ }^T \mathbf{T}^i \mathbf{h}^e} \right) \right] \right\}^2 \quad (4.40)$$

In this way, the consideration of any particular variable in the cokriging will not suffer for the relative magnitude of its estimated total variance,  $\hat{\sigma}^2$ . The squared error in Equation (4.40) is minimised subject to the requirement expressed in Equation (3.70) that the matrices  $\mathbf{S}^i$  are symmetric and positive definite. This requirement is met in the same way as the positive definite constraints on the transform matrices  $\mathbf{T}^i$  in Equation (3.28) were met earlier. These matrices are reformulated as the following sum of  $N$  outer products of vectors  $\mathbf{u}_k^i \in \mathbb{R}^N$ .

$$\mathbf{S}^i = \sum_k \mathbf{u}_k^i \mathbf{u}_k^{i \text{ } T} \quad (4.41)(a)$$

$$s^i = \sum_k u_k^i u_k^i \quad (4.41)$$

Using this formulation, an unconstrained optimisation over the domains of  $\mathbf{u}_k^i \in \mathbb{R}^N$  may be performed.

The concatenated pseudo-vector of parameters is simply;

$$\mathbf{r}^T = \left[ \mathbf{u}_1^{1 \text{ } T} \quad \dots \quad \mathbf{u}_N^{1 \text{ } T}, \quad \mathbf{u}_1^{2 \text{ } T} \quad \dots \quad \mathbf{u}_N^{2 \text{ } T}, \quad \dots \quad \mathbf{u}_N^{K \text{ } T} \right] \quad (4.42)$$

For the Levenberg-Marquardt algorithm, this amounts to the following residuals vector:

$$\mathbf{R}^{obj} \equiv \left[ S^e - \frac{C(\mathbf{h}^e)}{\sum_i s^i} \right] \quad (4.43)$$

where the subscripts in the pseudo-vector range over all extant combinations in  $\{\mathbf{h}^e ; S^e\}$  of  $\alpha$ - $\beta$  pairs and corresponding spatial statistic, indexed by  $e$ .

It is shown in Appendix D - 7 that the Jacobian  $\mathbf{J}$  with respect to the parameters  $u_{ab}^c$  in Equation (4.41) is ;

$$\frac{\partial C(\mathbf{h}^e)}{\partial u_{ab}^c} = \begin{cases} \frac{2 \left[ 1 - I^{\sigma} \left( \sqrt{\mathbf{h}_{aa}^{eT} \mathbf{T}^c \mathbf{h}_{aa}^e} \right) \right] u_{ab}^c}{\sigma_a \sigma_b}, & a = b \\ \frac{\left[ 1 - I^{\sigma} \left( \sqrt{\mathbf{h}_a^{eT} \mathbf{T}^c \mathbf{h}_a^e} \right) \right] u_b^c}{\sigma_a}, & a \neq b \\ \frac{\left[ 1 - I^{\sigma} \left( \sqrt{\mathbf{h}_a^{eT} \mathbf{T}^c \mathbf{h}_a^e} \right) \right] u_b^c}{\sigma_a}, & a \neq b \\ 0, & \text{otherwise} \end{cases} \quad (4.44)$$

Note that in the above, the denominators on the right hand side are actually all equivalent and equal to  $\sigma_a \sigma_b$ . In the preceding derivations, this factor is carried through all calculations without really effecting the basic residual minimisation problem. Indeed as the spatial statistics are all normalised between [-1, 1], it is possible to neglect this factor entirely in Equation (4.40), solve the minimisation problem for  $\mathbf{S}^i$  and then multiply the result retrospectively by the  $\sigma_a \sigma_b$  terms. This is simpler, but to permit greater flexibility in the future and for historical reasons, this is not how the code is structured.

The basic method for generating the coregionalisation model hence the final covariance model, has been presented. However, there are a number of constraints that the optimisation in this section should observe, and the exact nature of these depends on how the user chooses to model the behaviour of the covariance functions at the origin. These options and their implementation is presented in the next subsection.

### IV - 5.3. Coregionalisation modelling options

There are a number of constraints that may act on the minimisation in Section IV - 5.2. These are introduced to improve the physicality of the model, and to set optional arguments that control the modelling of the nugget effect. These aims are all achieved by largely the same constraint mechanism, which is described here.

In order to be realistic, some limit on the total variance is imposed – as suggested by Equations (3.71) and (3.72). As the covariances are normalised to unity (correlations), this amounts to a requirement that the entire normalised covariance

model lies within plus and minus one. However, this requirement is complicated by the possible need to set the total structured variance for any modelled auto or cross-covariance at a given level. There are two options for the coregionalisation in `krige` which concern the modelling of the nugget effect:

1. Unknown nugget effect: the nugget effect is left to ‘float’ to whatever value the spatial statistics suggest.
2. Known nugget effect: the nugget effect is set to a pre-conceived input value, on desired kriging variables.

Both of these options are facilitated by the manner in which the constraint on total variance is applied.

The total variance  $\sigma^2$  for each  $\alpha$ - $\beta$  pair comprises a structured and an unstructured component. The unstructured component is the nugget effect, and its treatment is different to the other variogram models. As it presents a non-differentiable discontinuity at the origin, it is modelled implicitly in much the same manner as in Section IV - 5.1. Spatial statistics located exactly at the origin can only arise from coincident data nodes – which do not occur in the data encountered thus far, so this should be unimportant anyway. It is also assumed that the nugget effect’s contribution to the cross-covariant functions is zero, as it would be unusual for such short-scale variability or ‘noise’, to be cross-correlated. This of course has the added benefit that the positive definite requirements in Equation (3.70) are automatically satisfied.

Whilst the total auto-covariance may be estimated by Equation (4.8), no similar estimation can be made of total cross-covariance, unless the nodal realisations of the different kriging variables are spatially coincident. Therefore, it is difficult to use Equation (3.71)

$$\sum_i s^i = \sigma^2 \quad \text{see (3.71)}$$

to impose a limit on total covariance for the cross-covariance functions. There is however some estimate of the total auto-covariances, so the limit in Equation (3.72)

$$\sum_i s^i \leq \sigma^2, \quad \forall \alpha \neq \beta \quad \text{see (3.72)}$$

is used instead. The limit on total variance, whether auto or cross-covariant, can be written generally as

$$\sum_i s^i \leq V \quad (4.45)$$

where usually

$$V = \quad (4.46)$$

Certainly, if some estimate of  $V$  is available, this may be used instead.

When the nugget effect is unknown, it is modelled implicitly. After the least-squares algorithm has run its course subject to the constraints on total variance in Equation (4.45), it is set as zero for the cross-covariance functions and the difference

$$s^{nugget} = V - \sum_i s^i \quad (4.47)$$

for the auto-covariant functions. When an estimate is supplied of the nugget effect's contribution for a particular kriging variable, say  $P^\alpha$  – the constraint in Equation (4.45) is simply modified to:

$$\sum_i s^i = V \quad (4.48)$$

If the nugget effect is known and the covariance may be estimated, then so may the total structured variance be estimated. Therefore, this quantity on the left of Equation (4.48) is fixed at a particular value  $V_{\alpha\beta}$  throughout the minimisation, and the known nugget effect contribution is set after the solution  $\mathbf{S}^i$  is found.

The above discussion amounts to a set of constraints indexed by  $\alpha$  and  $\beta$  limiting somehow the total variances to  $V_{\alpha\beta}$  in the manner described by Equation (4.45) or Equation (4.48). Given that this constraint can be either one, or two-sided (equal to or less than, or just equal to), the present author has employed a *penalty function* [66] (p. 277) to enforce it. This allows flexibility and most importantly simplicity, of code. The penalty function  $f_1^{pen}$  is of the form

$$f_1^{pen} = f_{con}^2 \sum_i \left\{ \frac{1}{\left| \sum_i s^i - V \right|} \right\}^4 \quad (4.49)$$

for the constraint in Equation (4.45), where  $f_{con}$  is a factor to scale the penalty function and  $\langle \cdot \rangle$  signifies the Macaulay function, which sends negative arguments to zero. Similarly, it is

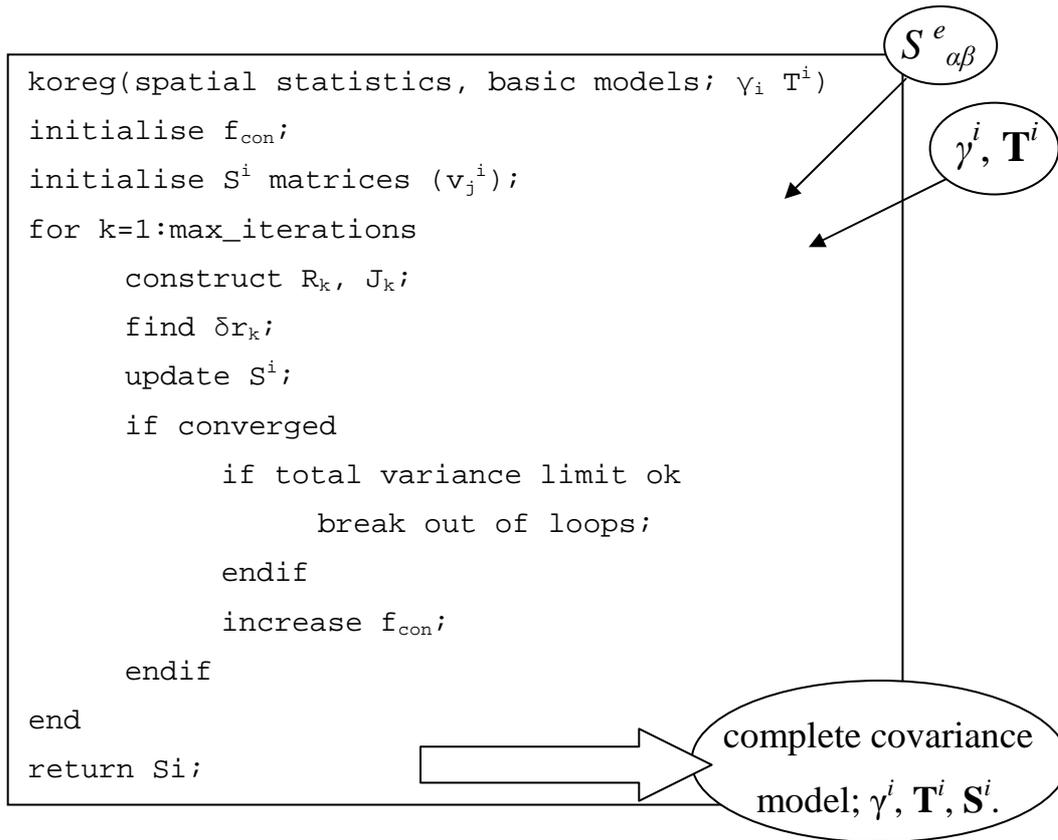
$$\frac{pen}{2} = f_{con}^2 \sum_i \left\{ \frac{1}{s^i} - V \right\}^4 \quad (4.50)$$

for the equality constraint in Equation (4.48). As before the difference is normalised by the maximum possible value  $\sigma_\alpha \sigma_\beta$ . Again, due to the symmetry of the cross-covariance functions, it is only necessary to consider those  $\alpha$ - $\beta$  combinations that are present in the sets of spatial statistics  $\{\mathbf{h}^e ; S^e\}$ .

The Macaulay function in Equation (4.49) immediately assumes non-zero values once past a boundary in the solution space of the problem, thus representing a discontinuity along this boundary. The order of this discontinuity is important. The present author has used penalty functions in Equations (4.49) and (4.50) that utilise the fourth power to ensure that the discontinuity is at least  $C^3$  continuous. The third derivative along this boundary is a continuous function. This means that the second order Hessian matrix (formed by the Jacobian inner product  $\mathbf{J}^T \mathbf{J}$ ) does not change abruptly over the constraint boundary. The penalty function is therefore gradual and ‘invisible’ to the Levenberg-Marquardt algorithm – which aids smooth convergence. It was experienced that discontinuous behaviour here has the potential to stall the algorithm as it proceeds along a constraint to a minimum: recurring ‘bouncing’ behaviours into and out of the constrained region may be initiated.

The fourth power is also relatively easily integrated into the least squares Levenberg-Marquardt algorithm as the penalty functions (4.49) and (4.50) can be written as extra squared residuals in the objective function. A vector of additional residuals results;

$$\mathbf{R}^{pen} = \begin{bmatrix} pen \\ \vdots \end{bmatrix} = \begin{bmatrix} f_{con} \left\{ \frac{1}{\left\| \sum_{i,n} u_n^i u_n^i - V \right\|} \right\}^2 \\ \vdots \end{bmatrix} \quad (4.51)$$



**Figure IV—11: Pseudo code for fitting a coregionalisation model, hence determining  $S^i$  and the complete covariance model.**

which is concatenated with those already present in Equation (4.43)

$$[\mathbf{F} - \mathbf{f}] \equiv \begin{bmatrix} \mathbf{R}^{obj} \\ \mathbf{R}^{pen} \end{bmatrix} \quad (4.52)$$

Note that in Equation (4.51), the  $[\cdot]$  bracketing conveys that the Macaulay function in Equation (4.49) or normal brackets in Equation (4.50) are to be inferred as are needed to describe the one-sided or two-sided constraints, respectively. The factor  $f_{con}$  is arbitrary, and it is adjusted from an initial ‘small’ value to a value sufficiently large as to satisfy the constraints in Equations (4.45) or (4.48) to some degree of accuracy – the present author has chosen one part in one hundred. Its adjustment is outlined schematically in the pseudo code in Figure IV—11. The extra terms  $\mathbf{R}^{pen}$  in the residual vector mean that there will be extra corresponding terms in the Jacobian, calculated in Appendix D - 8 as;

$$\frac{\partial \text{pen}}{\partial u_{ab}^c} = \begin{cases} \frac{2f_{con}}{a^2} \left[ \sum_i s_{aa}^i - V_{aa} \right] 2u_{ab}^c, & = a \\ \frac{2f_{con}}{a^2} \left[ \sum_i s_a^i - V_a \right] u_b^c, & = a, \neq a \\ \frac{2f_{con}}{a^2} \left[ \sum_i s_a^i - V_a \right] u_b^c, & \neq a, = a \\ 0, & \text{otherwise} \end{cases} \quad (4.53)$$

which completes the Jacobian matrix.

The decision to handle constraints with penalty functions was largely inspired by the desire to simplify the programming problem – the approach does have its pitfalls. As with all penalty function methods, the constraint is only ever satisfied to within some degree of accuracy. Furthermore as quite an aggressive penalty function using the fourth power was chosen, the solution (minimum) itself can be quite ill-conditioned, posing difficulties at convergence for the optimisation algorithm. This problem is exacerbated if a very tight tolerance on the constraint is required, and it is particularly notable for the two-sided constraint. To alleviate these difficulties, a rather coarse tolerance on constraint convergence was specified – an error of one part in one hundred, on any given constraint. There are a number of flexible options in the optimisation that are also modified to yield more robust performance, as are outlined in the Appendices C - 2 and E - 6.

## IV - 6. Estimation

Having determined a complete model for covariance by considering all the data, estimates are produced by considering local subsets of the data. How these subsets are chosen, and more particularly how this is managed when there are many estimation points and local subsets – is outlined in brief here. A more detailed explanation of estimation and especially data management, is found in the programming notes.

Often and especially when nodal data is to be interpolated, estimates are required at a multitude of locations. It was noted before that this can be achieved by solving the kriging equations in Equations (3.37), (3.55) or (3.58) with different right hand side vectors  $\mathbf{c}_0$ , back-substituting the same left hand side  $\mathbf{LU}$  (or equivalent) decomposition. However, on the grand scale of entire datasets this is both impractical and possibly theoretically inconsistent. When say a regular grid or raster of estimation points is required that covers all or a significant portion of the local data, a set of nodal points local to each estimation point must be chosen. The kriging equations are then written and solved for this subset. This is done for two reasons. Firstly, if the entire set of nodal data were used, an unfeasibly large set of equations might result, and even performing one  $\mathbf{LU}$  decomposition on them could be very time-consuming. Secondly, the kriging equations assume second order stationarity and to be realistic, the data used in the estimation ought to be of the same scale so as to make this assumption tenable. However, it is worthwhile to bear in mind that the covariance functions will still screen out data that is outside of this scale – data that is far away from the estimation point does not usually play a great role in the final weighting and estimation scheme.

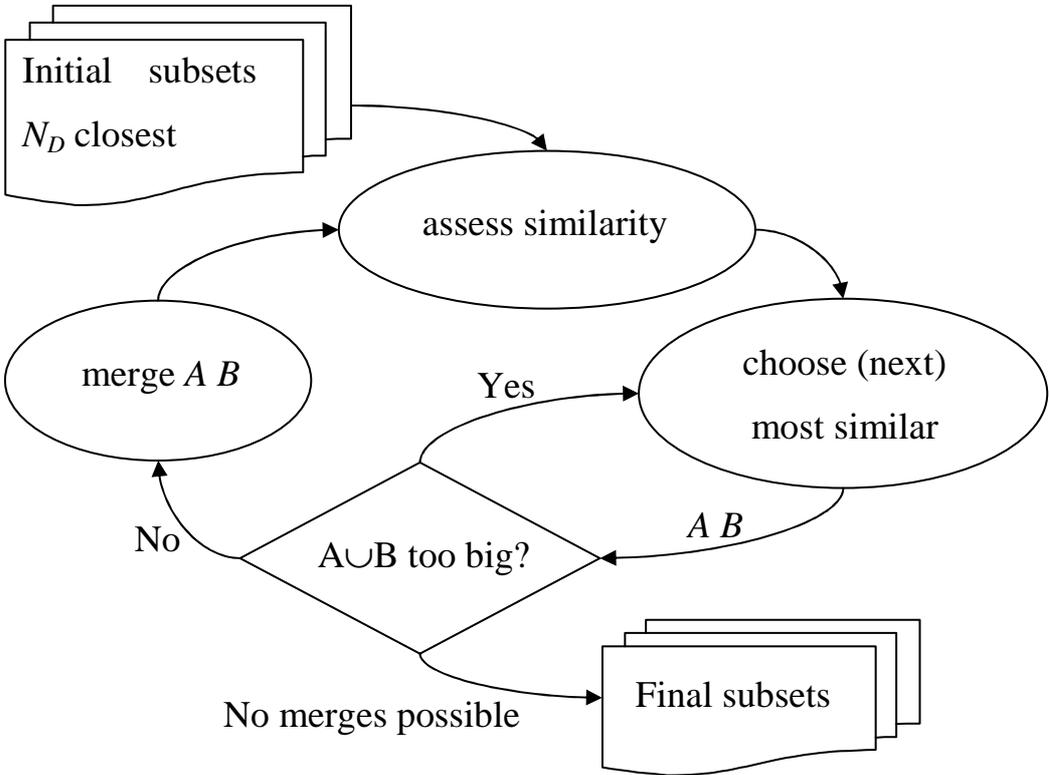


Figure IV—12: Generating local spatial subsets from initial  $N_D$  closest nodes.

Therefore when estimating multiple points, the kriging equations are written for subsets of the global data that are local to each estimation point. However, choosing separate subsets for each point may also be computationally inefficient. If two local subsets are very similar (or the same) it is worthwhile to merge them, as the numerical cost of a slightly enlarged system of equations compares favourably with performing two, largely similar **LU** decompositions. There are then, conflicting aims in that maintaining small subsets diminishes the systems of equations, whereas merging subsets enlarges the systems of equations. A balance must be struck between solving a small number of large systems, or a large number of small systems.

For simplicity, local subsets consisting of the nearest  $N_D$  data to each estimation point are chosen. Note of course, that these subsets are not mutually exclusive – their members may overlap. These subsets are then merged on the basis of similarity, up to a maximum size  $N_{max}$ . Also, for storage reasons a limit  $P_{max}$  is imposed on the number of estimation points. Which sets to merge is determined on a competitive basis: the most ‘similar’ sets are merged, provided they are not too big. Similarity is measured in terms of fractions  $C$  of member overlap of the smaller member, thus where  $A$  and  $B$  are the sets of nodes to be merged and  $N(\cdot)$  is the number of nodes (members) in each;

$$C = \frac{N(A \cap B)}{N(A)} \quad \text{where } N(A) < N(B) \quad (4.54)$$

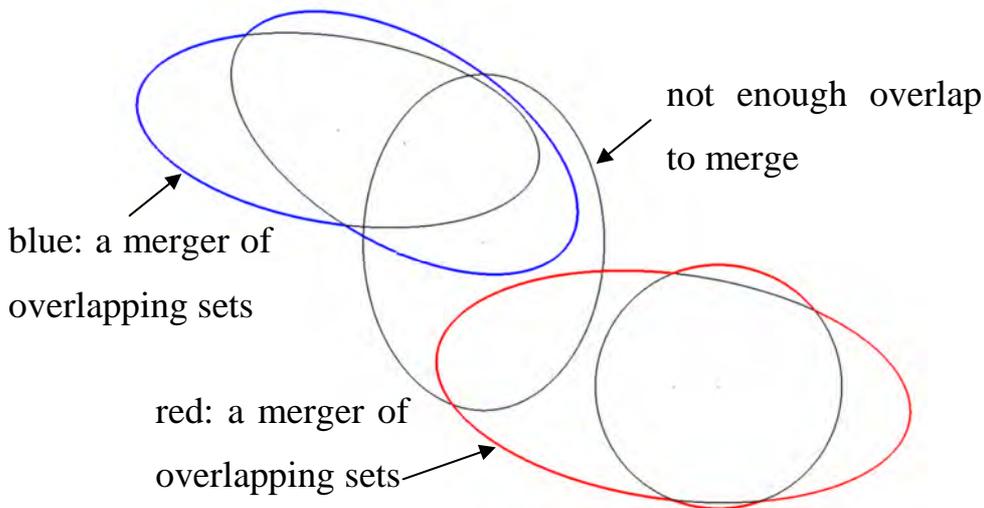
When two sets are merged, the similarity of the resulting set is re-calculated with respect to the remaining sets, and the next, most similar pair is chosen for possible merging – set size permitting.

This basic procedure is unfortunately complicated by the way in which the raw data is stored. As previously mentioned, the raw nodal data is stored on disk in spatially local clusters, to limit the size of searches and expand maximum problem (or database) size beyond what is achievable solely in RAM. Thus the searches outlined above have first to pass through a preliminary search for the data clusters. To find the nearest node to an estimation point, one must first look for the nearest clusters, retrieve them, and then look amongst their contents for the nearest nodes. This of course is a ridiculous excursion when performed for individual points, but

becomes practical when there are groups of such points to be fetched. In this case, essentially the same algorithm is used as illustrated in Figures IV—12 and IV—13, but the parameters  $N_D$ ,  $N_{max}$  and  $P_{max}$  are changed to meet different storage, latency and performance requirements. Thus for each estimation point, a search is performed for the nearest  $N_D$  clusters – as measured to the cluster’s spatial centroid, usually  $N_D = 2^N$  when the datasets are in  $\mathbb{R}^N$ . These sets are then merged up to  $N_{max}$  and  $P_{max}$ , where now  $N_{max} = N_D$  and  $P_{max}$  is set by storage requirements. At the end of this, there are sets of estimation points that are relevant to sets of clusters. These clusters are in turn read from disk, and searches on the actual nodal data are performed on the data therein.

Once the local subsets have been generated, estimation is relatively straightforward. Covariance matrices and their corresponding multiple left hand side vectors are constructed as appropriate using the covariance model determined in the previous sections. These are then solved for the estimation weights which are used to produce an estimate using Equation (3.46) and calculate the kriging variance. Equations (3.15) and (3.48) are not used to calculate the kriging variance; instead an algebraic shortcut is employed. Note that all the kriging equations have a basic form corresponding to

$$\begin{bmatrix} \mathbf{C} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ \mathbf{b} \end{pmatrix} \quad (4.55)$$



**Figure IV—13: Merging five local subsets on the basis of common members.**

Using this form, Equation (3.48) or any kriging variance may be re-written in the form

$$\begin{aligned} \sigma_{CK}^2 &= \sigma^2 + \sum_{i,j} w_i w_j C(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_i C(\mathbf{x}_i, \mathbf{x}_0) \\ &= \sigma^2 + \mathbf{w}^T \mathbf{C} \mathbf{w} - 2 \mathbf{c}^T \mathbf{w} \end{aligned} \quad (4.56)(a)$$

The  $\mathbf{C}$  matrix term may be expanded to include the  $\mathbf{B}$  matrices in Equation (4.55), as long as the extra terms are subtracted from the extended inner product;

$$= \sigma^2 + (\mathbf{w}^T \quad \mathbf{a}^T) \begin{bmatrix} \mathbf{C} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{a} \end{pmatrix} - 2 \mathbf{a}^T \mathbf{B} \mathbf{w} - 2 \mathbf{c}^T \mathbf{w} \quad (4.56)$$

Note that as an adjunct to Equation (4.55), there is the relation

$$\mathbf{B} \mathbf{w} = \mathbf{b} \quad (4.57)$$

because the zero matrix in the corner of Equation (4.55) effectively decouples the relationship between the  $\mathbf{B}$  and  $\mathbf{b}$  matrices. Equations (4.55) and (4.56) may then substitute for the matrix inner products in Equation (4.56) to arrive at

$$\begin{aligned} \sigma_{CK}^2 &= \sigma^2 + (\mathbf{w}^T \quad \mathbf{a}^T) \begin{pmatrix} \mathbf{c} \\ \mathbf{b} \end{pmatrix} - 2 \mathbf{a}^T \mathbf{b} - 2 \mathbf{c}^T \mathbf{w} \\ &= \sigma^2 - \mathbf{a}^T \mathbf{b} - \mathbf{c}^T \mathbf{w} \end{aligned} \quad (4.58)$$

which presents less operations than the original Equation (3.48), at the expense of the storage of the right hand side of the kriging equations in Equation (4.55) – which might be smaller than the left hand side, anyway.

Also available, are some options regarding the actual estimated quantity. The present author has adopted the convention that the ‘first’ variable,  $P^1$  amongst  $P$ , is the estimated quantity. Of course, there is no particular reason to structure the covariance matrices this way and indeed, any of the variables in a cokriging could be the primary variable in Equations (3.55) or (3.58), with the simple re-arrangement of the right hand side. To avoid complexity this convention has been maintained in the code, but similarly changes may be made to just the right hand side of the generalised Equation (4.55) which change the estimated quantity in some way. Options for three such changes are available:

1. Normal estimation; right hand side covariances as calculated by the covariance model.
2. Estimation filtered for noise; right hand side covariances calculated without the nugget effect.
3. Estimation of underlying drift; right hand side covariances are zero.

All of these options change the estimated quantity, but use the same model and basic equations, which are modified only on the right hand side.

The first option is straightforward, and estimation proceeds as described in Chapter IV. It was noted in Section III - 3 that the nugget effect model has the effect of performing a spatial averaging to remove ‘noise’ from the estimation. In actual fact, the interpolation is still ‘exact’ and still passes through all the nodal data points in this case. However because of the discontinuous nugget effect, the interpolation effectively jumps from the spatially averaged trend-line to the known value at each node. Apart from these point discontinuities, the interpolation is continuous. The second option listed above simply removes the discontinuities that are introduced by the nugget effect. To do this the right hand side vectors in Equation (4.55) are calculated normally, but the nugget effect model’s contribution is neglected so that

$$\begin{bmatrix} \mathbf{C} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} \mathbf{c}^* \\ \mathbf{b} \end{pmatrix} \quad (4.59)$$

where  $\mathbf{c}^*$  is calculated using

$$\mathbf{C}^* (\mathbf{h}) = \sum_{i, i \neq \text{nugget}} s^i \left[ 1 - I^* \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right] \quad (4.60)$$

The left hand side matrices remain exactly the same – they include any nugget effect contributions, as usual. Kriging variance is calculated analogously using Equation (4.58). Note that whereas it should be zero at the ‘known’ points, filtering the results for noise by using Equation (4.60) means that it is now non-zero at these points – there is uncertainty in the raw data.

The third option estimates the non-random drift expressed in Equations (3.29) and (3.52) that is present in the primary variable. To estimate the drift component the right hand side is modified so that [84]:

$$\begin{bmatrix} \mathbf{C} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix} \quad (4.61)$$

The resulting weights are then applied in the usual way using Equation (3.46) to produce an estimate. What Equation (4.61) actually expresses is that the estimated quantity – the drift, has zero covariance with the known nodal quantities around it, hence the vector  $\mathbf{c}$  is a zero vector. This is because ideally, the drift has the property that it is uncorrelated with the random fluctuations about it. Note however, that this is still just an estimate of the actual drift and there is statistical uncertainty in the estimate. Indeed, there is a modelled kriging variance associated with such an estimate, which is again calculated using Equation (4.58).

One might be tempted to estimate the drift at all of the nodal data-points, with a view to detrending them *before* generating a better covariance model and estimations. Certainly, the drift pollutes the spatial statistics and may lead to misspecification of the covariance model as a result. However, adopting a simplistic approach like this does not work, again for the reason that there is uncertainty in the drift estimation itself. Generally this uncertainty is of the same magnitude as the estimation uncertainty – detrending the data in this way was attempted, and it was found that although the biasing effects of the drift were removed, so was any spatial correlation. As previously noted, much of the value of the kriging estimator lies in the ambiguity of the drift.

# Chapter V - Case Study 1: Numerical investigations

In this thesis, the random models that support kriging are adjuncts to the broader aims of data assimilation and estimation. The kriging estimator is considered to be a tool for estimation, and the deterministic origins of the numerical spatial data are overlooked for the purpose of this exercise. Of course the numerical models are not random in the usual aleatoric sense, because any repeat of the numerical experiment will yield exactly the same results, under identical initialisation and convergence criteria. However, they certainly comprise epistemic uncertainties, and the scale and complexity of any non-trivial numerical solution may permit some kind of statistical interpretation. How these epistemic uncertainties may contribute to a greater statistical interpretation of the stationary covariance model on a numerical result is investigated in this chapter.

The present authors consider stationary statistics calculated on the numerical data as broadly reflecting the inter-relation of the data at scales larger than the numerical grid. If the actual solution to the equations were to be found, free of iterative and truncation errors, it would arguably consist entirely of a drift component with no uncertain or random component. However for non-exact numerical solutions,

there is epistemic uncertainty associated with an unknown higher order error function and possibly even chaotic iterative uncertainty due to non-linearity. In this context, frequentist statistical interpretation cedes to a more Bayesian conception of probability. Some means of interpreting stationary spatial statistics calculated on deterministic numerical results is sought.

A way in which numerical models certainly *can* behave in an seemingly random manner is in the manifestation of chaotic behaviours [4]. Because of the non-linearity of the Navier-Stokes equations, non-trivial numerical solutions of them are always iterative at some level. An acceptable maximum residual on the approximate solution is often defined for the purpose of limiting convergence error, but even if the solution is to iterate until the limit of machine precision is reached, it will continue to change in the last couple of significant figures – sometimes periodically, but most often quasi-randomly or *chaotically*. When the solution is terminated, it is merely a snapshot of a non-linear dynamic system which is evolving in a quasi-random manner to its next state. This form of iterative randomness shall not be examined directly here, yet is worth bearing in mind as part of the argument for statistical appraisals of numerical models.

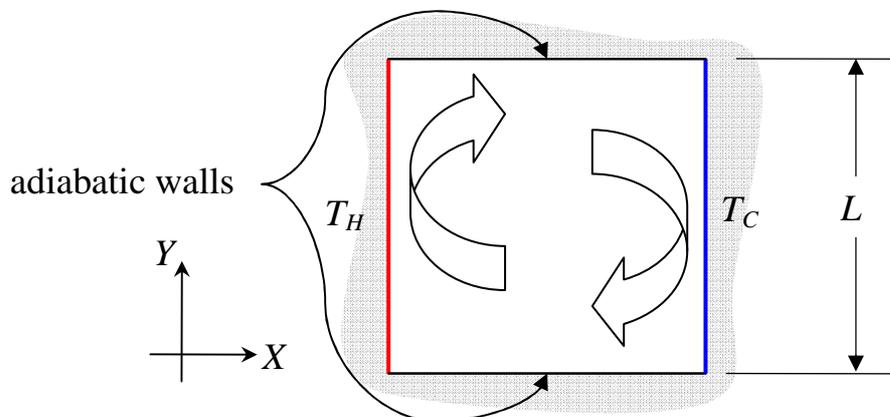
The FV method does not assume a functional form between nodes for reasons both of operational and theoretical necessity [8, 9]. This begs the question; what is happening in between the nodes? The solution is assumed to be smooth for the calculation of derivatives and fluxes at the FV cell boundaries, yet unlike the Finite Element (FE) method continuity is not explicitly modelled over each cell. The local truncation error committed over each cell consists of higher order terms in the local Taylor series expansions – effectively an epistemic uncertainty which the present author idealises as a random behaviour. Indeed a contour map of a mesh quality indicator such as “skew ratio” or “squish index” on an unstructured grid, often looks to all intents and purposes like a stationary random function of sorts. Therefore it is proposed here that the smoothness of the FV solution provides some indication of error, and it may be examined at least roughly using the stationary (and possibly *generalised*) covariance functions.

The effect of grid size, hence truncation error on stationary covariance functions is examined in this chapter. In contrast with real-world numerical solutions

which effectively display randomness through their high complexity, very simple, contrived situations are examined for which one cannot argue that the simulated physics is sufficiently complex for the flow-field to take on quasi-random properties. The assumption of underlying aleatoric variation is attenuated in this way for two reasons; to check that useful statistics can still be generated even on relatively simple problems, and in order to see if any useful behaviours emerge as the numerical data approach convergence. The latter reason is of great theoretical interest, as a means of judging grid convergence on the basis of spatial statistics could be a very useful verification tool.

## V - 1. Convection in a Square Cavity

One of the first test cases for the prototypical kriging algorithm was its application to results concerning the well-known numerical benchmark established by De Vahl Davis [112]: the natural convection of air inside of a two-dimensional square cavity with opposing hot and cold isothermal walls, and adiabatic walls at its top and bottom surfaces. This is represented schematically in Figure V—1. Given a square one-to-one aspect ratio, the fluid velocity and temperature distribution are fully described by a Rayleigh number ( $Ra$ ) and a Peclet ( $Pe$ ) number. The steady-state Navier-Stokes equations are solved with a buoyancy term over a series of grids at different resolutions for which spatial statistics and covariance functions are generated and fitted. The progression of these functions towards typically smoother



**Figure V—1: Schematic of convection in a square cavity.**

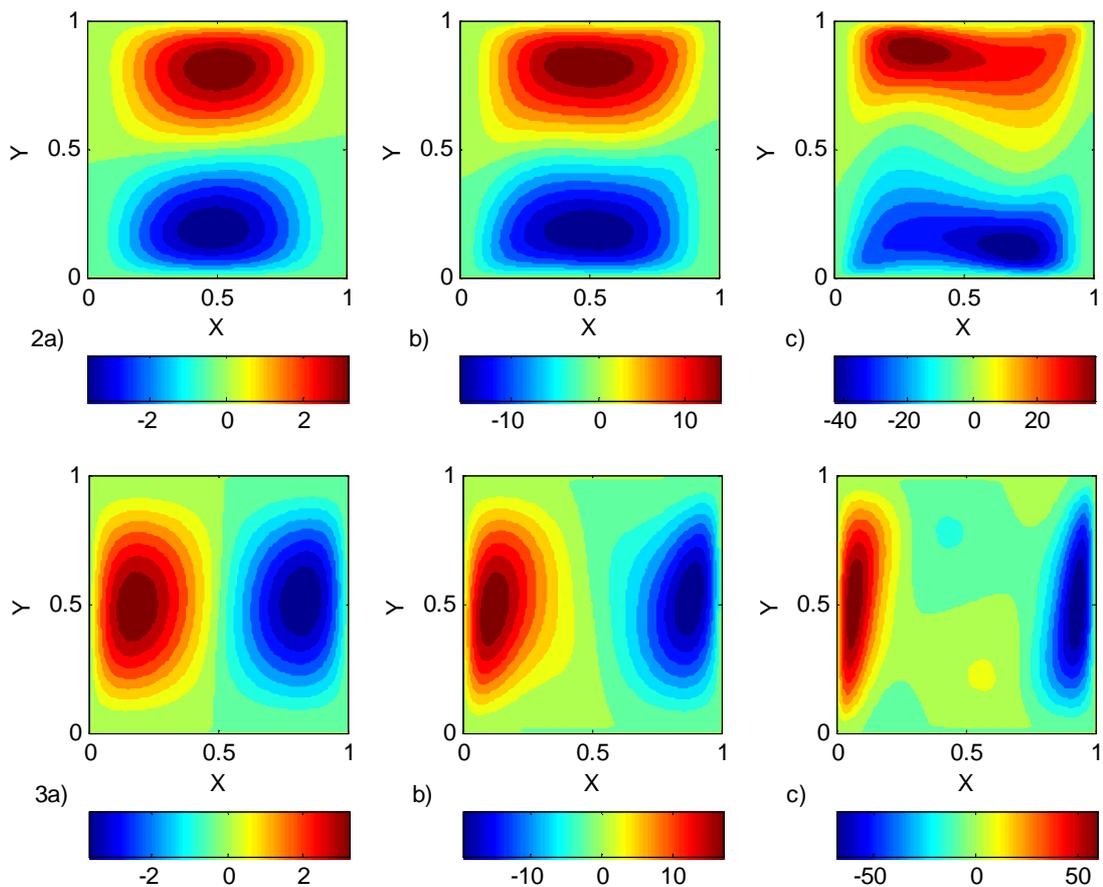
structures is examined.

### V - 1.1. Description of flow

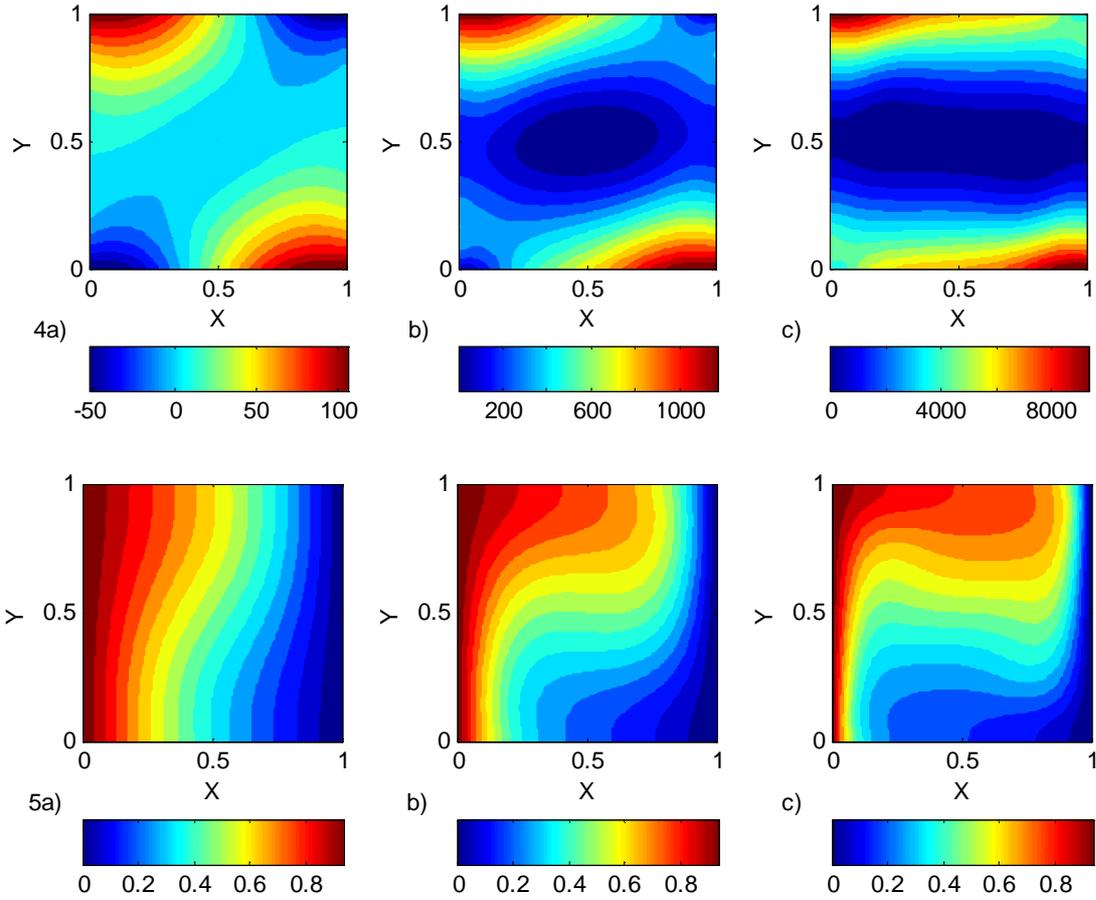
This two-dimensional flow is characterised by the boundary conditions, material properties and the side length of the square domain. The fluid is assumed to be an incompressible, ideal, Newtonian fluid and a Boussinesq approximation is used to calculate a buoyancy force per unit volume due to variations in temperature – ostensibly modelling the effect of changes in density. The reference temperature  $T_R$  for linearisation is the average temperature over the hot and cold walls;

$$T_R = \frac{T_H + T_C}{2} . \quad (5.1)$$

The equations of continuity, momentum conservation and energy transport



Figures V—2: Non-dimensionalised  $x$ -velocity, and V—3:  $y$ -velocity; (a)  $Ra = 10^3$ , (b)  $Ra = 10^4$ , c)  $Ra = 10^5$ .



**Figures V—4: Non-dimensionalised pressure, and V—5: temperature; (a)  $Ra = 10^3$ , (b)  $Ra = 10^4$ , (c)  $Ra = 10^5$ .**

respectively are

$$\nabla \cdot \mathbf{u} = 0, \quad (5.2)(a)$$

$$\mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nabla^2 \mathbf{u} + \begin{bmatrix} 0 \\ g (T - T_R) \end{bmatrix} \text{ and} \quad (5.2)(b)$$

$$C_p \mathbf{u} \cdot \nabla T = k \nabla^2 T \quad (5.2)(c)$$

for this problem. Symbols have their customary meanings;  $\mathbf{u}$  is velocity ( $\text{ms}^{-1}$ ),  $p$  is pressure (Pa),  $\mu$  is dynamic viscosity ( $\text{Ns m}^{-2}$ ),  $\rho$  is density ( $\text{kg m}^{-3}$ ),  $T$  is temperature (K),  $k$  is thermal conductivity ( $\text{W m}^{-1}\text{K}^{-1}$ ) and  $C_p$  is specific heat capacity ( $\text{J kg}^{-1}\text{K}^{-1}$ ). The variable  $\beta$  is the coefficient of thermal expansion ( $\text{K}^{-1}$ ) used in the Boussinesq approximation, which is

$$= \frac{1}{T_R} = \frac{2}{T_H + T_C} \quad (5.3)$$

for an ideal gas at the reference temperature  $T_R$ . Note that the gravitational acceleration  $g$  ( $\text{ms}^{-2}$ ), is a positive number but is acts in the negative  $y$ -direction indicated in Figure V—1 when  $T < T_R$ . The flow is thus characterised by a Rayleigh Number and a Prandtl Number:

$$\text{Pr} = \frac{C_p}{k} \quad (5.4)$$

$$\text{Ra} = \frac{2^2 g C_p (T_H - T_C) L^3}{k (T_H + T_C)} \quad (5.5)$$

where  $L$  is the side length of the square enclosure,  $T_h$  is the temperature of the hot wall and  $T_c$  is the temperature of the cold wall. The spatial dimensions, differential operators and unknowns in Equation (5.2) may be expressed non-dimensionally as

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{L}, \quad \tilde{\nabla} \equiv L \frac{\partial}{\partial x_i}, \quad \tilde{\nabla}^2 \equiv L^2 \sum_i \frac{\partial^2}{\partial x_i^2} \quad (5.6)(\text{a,b,c})$$

$$\tilde{\mathbf{u}} = \frac{C_p L \mathbf{u}}{k}, \quad \tilde{p} = \frac{L^2 C_p^2 p}{k^2}, \quad \tilde{T} = \frac{T - T_C}{T_H - T_C}. \quad (5.6)(\text{d,e,f})$$

With some manipulation Equations (5.2) can be re-written non-dimensionally as:

$$\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0 \quad (5.7)(\text{a})$$

$$\tilde{\mathbf{u}} \cdot \tilde{\nabla} \tilde{\mathbf{u}} = -\tilde{\nabla} \tilde{p} + \text{Pr} \tilde{\nabla}^2 \tilde{\mathbf{u}} - \text{Ra} \text{Pr} \tilde{T} \quad (5.7)(\text{b})$$

$$\tilde{\mathbf{u}} \cdot \tilde{\nabla} \tilde{T} = \tilde{\nabla}^2 \tilde{T} \quad (5.7)(\text{c})$$

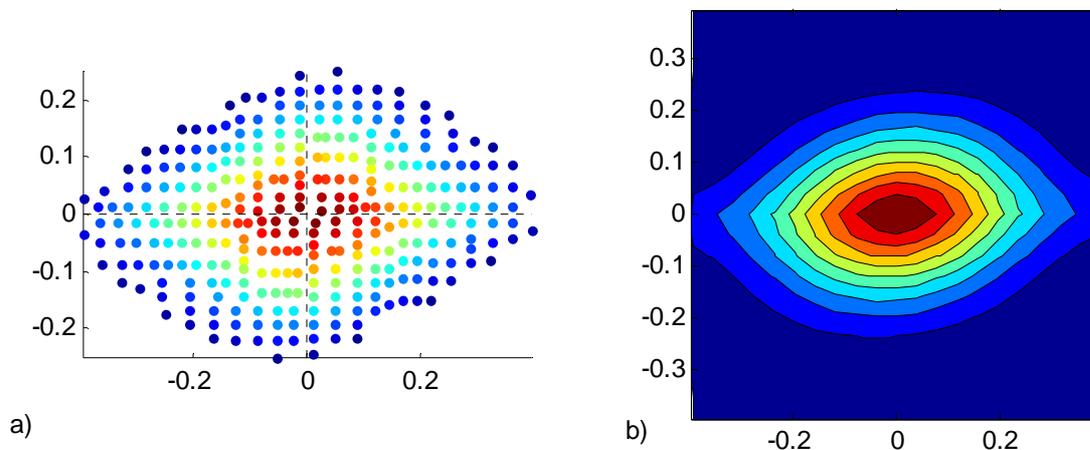
All results and further discussion are tacitly in terms of the non-dimensionalised terms introduced above.

The Prandtl number of air for all simulations was assumed to be  $\text{Pr} = 0.71$ , and the Rayleigh number took on values of  $\text{Ra} = 10^3$ ,  $\text{Ra} = 10^4$  and  $\text{Ra} = 10^5$ . This simulation was performed with a regular FV discretisation in FLUENT on a series of different sized grids;  $24 \times 24$ ,  $32 \times 32$ ,  $40 \times 40$ ,  $48 \times 48$ ,  $56 \times 56$ ,  $64 \times 64$ ,  $72 \times 72$  and  $80 \times 80$  control volumes. A first order upwinding scheme with SIMPLE pressure coupling was adopted. This scheme was not selected for outright accuracy but for the purposes of examining the convergence of the solutions. Iterative convergence limits were set reasonably low; at a FLUENT energy norm of  $2.0 \times 10^{-4}$  for the momentum and

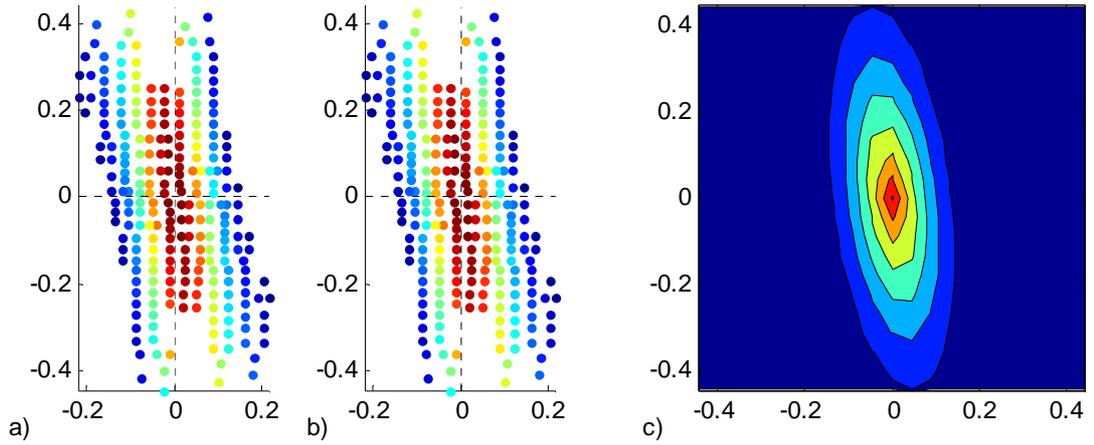
continuity equations, and  $2.0 \times 10^{-7}$  for the energy equations. It is therefore expected that the main source of error is truncation error. Results consisting of contour plots of normalised  $x$  and  $y$ -velocity, pressure and temperature are presented in Figures V—2, V—3, V—4, and V—5 respectively, for the finest  $80 \times 80$  grid.

### V - 1.2. Covariance function behaviour

To examine the behaviour of the stationary covariance structures, correlograms and variograms were generated for the spatial variables at each level of discretisation. This was performed using the algorithms outlined in Chapter IV - Implementation, first using the nodal results to generate sample points on the correlogram/variogram surfaces, and then fitting the canonical structures in Equation (3.23) to these points. For Equation (3.28) three canonical structures only were used – the Gaussian, the exponential and the spherical models (the nugget effect is fitted implicitly). More complicated combinations of models were not investigated – there are, after all, limited sample points from which inferences may be drawn. The use of too many models can lead to unrealistic covariance modelling, as each model can specifically cater to each small deviation from the general behaviour. By choosing just three models, the important behaviour at the origin of the covariance function is well resolved: smooth behaviour at the origin is represented via the Gaussian model, linear behaviour via the exponential and spherical models, and noise via the nugget effect.



**Figure V—6: Auto-correlation function for  $x$ -velocity,  $Ra = 10^4$ ; (a) Sample points, (b) corresponding covariance function model.**



**Figure V—7: Auto-covariant structures for  $T$ ,  $Ra = 10^5$ ; (a) Correlation statistics (b) variogram statistics (c) corresponding covariance function model.**

As an example, the sample correlogram for the  $x$ -component of velocity;  $Ra = 10^5$ ,  $80 \times 80$  grid, and the covariance function  $C(\mathbf{h})$  that has been fitted to this sample correlogram, is presented in Figure V—6. Whilst slightly anisotropic, the spatial statistics in Figure V—6(a) do not seem to be adversely affected by the top-to-bottom variation in  $x$ -velocity. The stationarity assumption is generally badly effected by drift in the data, which can be filtered in estimation but is difficult to filter in structure identification. The mal-effects of drift manifest themselves as very pronounced anisotropies in some of the other variogram models. An example of this is given in Figure V—7, which shows a sample correlogram and corresponding covariance function for normalised temperature, again at  $Ra = 10^5$ ,  $80 \times 80$  discretisation. For comparison, a variogram of the same quantity is produced in Figure V—7(b), but even the assumption of stationarity at a higher order is not enough to remove the strong directional dependency. There is an even higher order drift in these results which is not filtered by preliminary linear detrending of the data. The source of the problem is apparent when comparing in Figures V—2 and V—5, the normalised  $u_x$  and temperature contours. Detrending the temperature profiles in a simple linear manner removes their top-to-bottom variation, but leaves their left-to-right variation, because linear detrending cannot accomplish both. By contrast, the top-to-bottom drift in velocity is better removed by a linear correction.

As it is the behaviour near the origin of the covariance model  $C(\mathbf{h})$  that characterises the smoothness of the data, the total contribution of each variogram model to the function  $C(\mathbf{h})$  through their coefficients  $t_i$  in Equation (3.28) is

examined. Figure V—8 is a plot of percentage Gaussian (smooth) behaviour for pressure, temperature and the velocity components. This is the percentage of the total variance  $\Sigma t_i$  that has been formed by the value  $t_{GAUSS}$  in Equation (3.28) once the covariance model has been fitted in the manner described in Section IV - 5.1. Of particular interest is how the total covariance model changes as the grid is refined. Given the relatively simple numerical problem, any such change is due to the extra confidence with which one may infer the continuity of the covariance function at the origin, which as noted before can only be resolved to the scale of the minimum grid spacing, at best.

In Figure V—8 it is seen that the Gaussian model predominates as finer discretisations are used, implying that the nodal data becomes functionally smoother as the grid is refined. The transition is not absolutely monotonic, perhaps because of the imperfect nature of both the generation of the sample points and the subsequent covariance modelling, which were formulated with greater consideration of estimation problems in mind. In particular, the Gaussian model plays no role in describing the covariance model for pressure at  $Ra = 10^5$  for fine discretisations, perhaps because of the difficulty of fitting a covariance model to the marked anisotropies observed previously. In spite of this, there is certainly a trend towards a greater Gaussian coefficient as grid size increases and furthermore, this seems to happen earlier for lower Rayleigh numbers, which may reflect the earlier

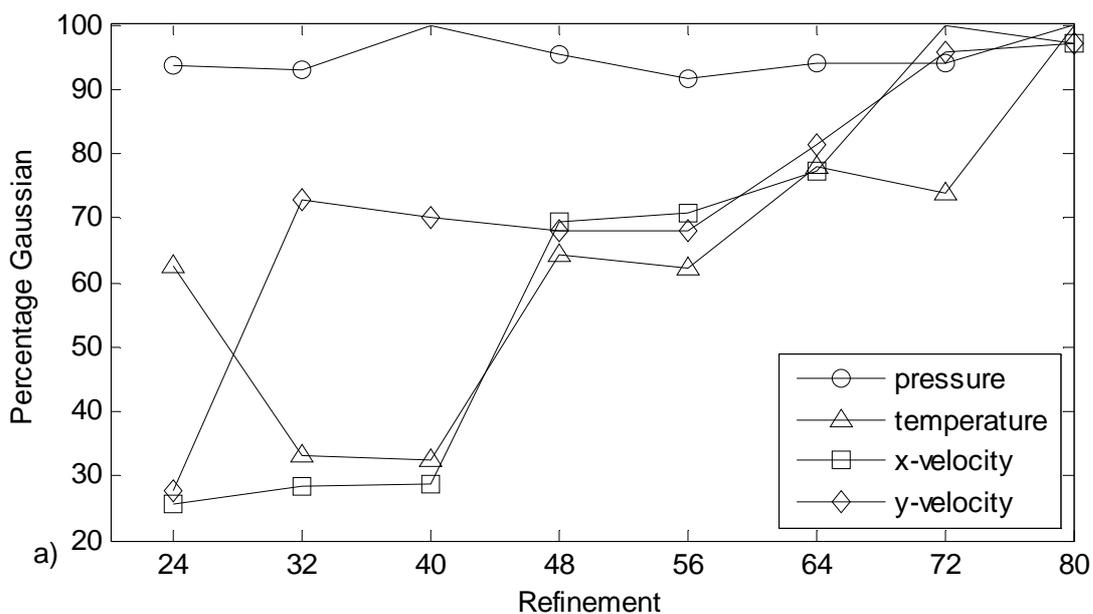
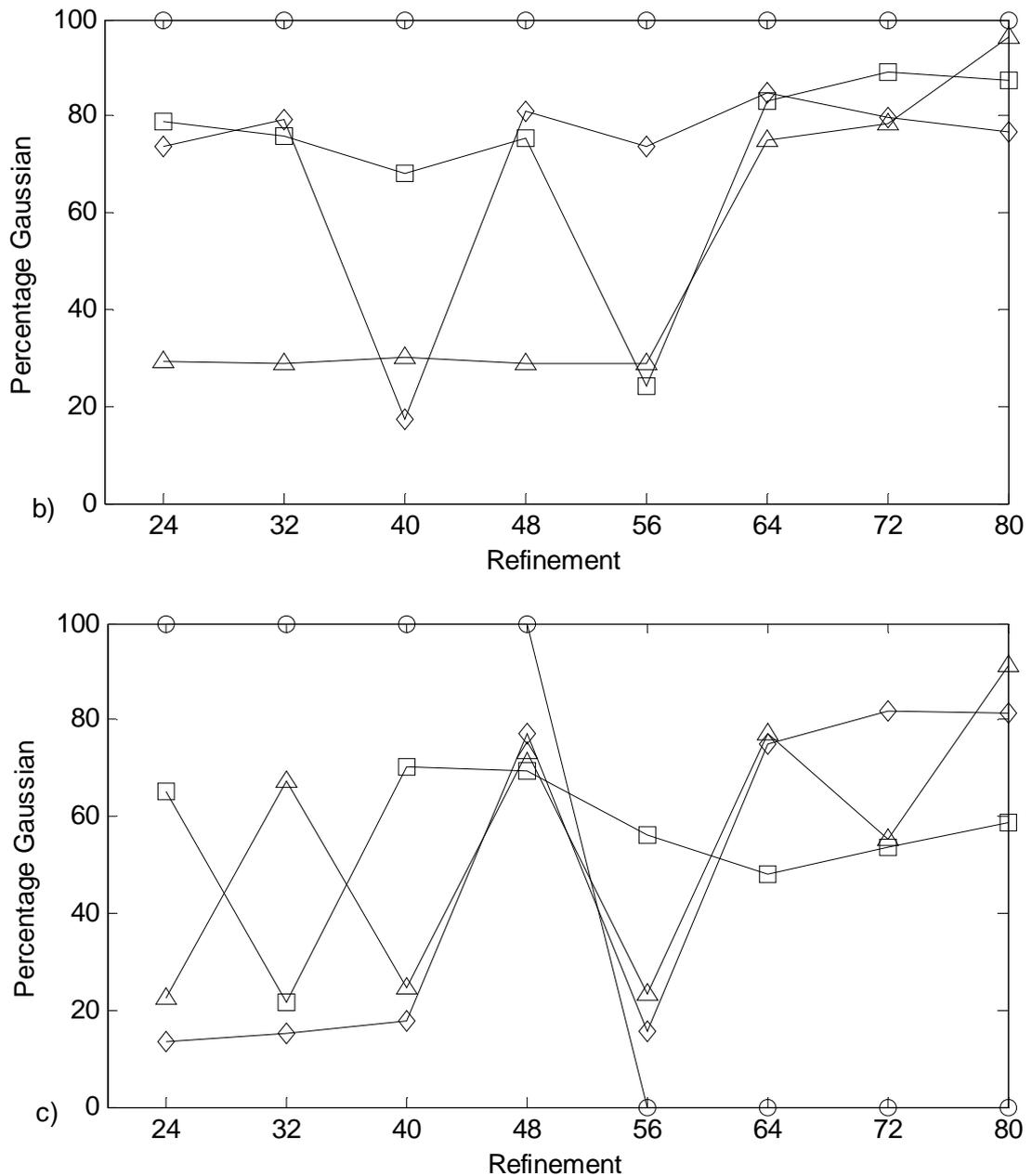


Figure V—8(a) Percentage Gaussian component for  $Ra = 10^3$ .



**Figure V—8(b,c) Percentage Gaussian component for (b)  $Ra = 10^4$  and (c)  $Ra = 10^5$ .**

convergence of these models.

Intuitively, one should expect a gradual shift towards smoother behaviours as the mesh is refined simply because more spatial statistics (lag vectors) are generated close to the origin of the covariance functions. This behaviour is not undesirable, but it is of far greater interest should this phenomenon actually be a by-product of the reduction of higher-order truncation errors, as this would imply that such errors can be quantified using spatial statistics. Unfortunately, the relatively agricultural implementation proposed in Chapter V is arguably not sufficiently sensitive to these

changes for conclusive results here. This is why in the next section, a more complex *generalised* covariance model is prepared with reference to a yet simpler numerical problem.

## V - 2. Generalised Covariance Theory

It is seen in the previous section that the presence of a large-scale drift that is not removed by preliminary detrending distorts the generation of spatial statistics. Such a drift may bias the spatial statistics in a particular direction, resulting in pronounced anisotropies in the covariance function (Figure V—7). The relatively crude preliminary linear detrending is used for simplicity only. It is possible to remove the drift more rigorously at least over small scales, by using a higher order description of the random function, briefly introduced earlier as an *intrinsically random function of order k* (IRF- $k$ ) in Section III - 5.

Higher order descriptions of stationary covariance are built on linearly weighted combinations of the value of the random function over a spatial template. These however are not just any linear combinations, but *Allowable Linear Combinations of order k*, or ALC- $k$  for short. An ALC- $k$  over some function  $Z$ , abbreviated as  $Z(\lambda)$ , takes the form [84]

$$Z(\lambda) \equiv \sum_i \lambda_i Z(\mathbf{h}_i + \mathbf{x}) \quad (5.8)$$

This is best visualised as a template applied at some point  $\mathbf{x}$  in the domain of the function  $Z$ , where the function value at the template points  $\mathbf{x} + \mathbf{h}_i$  are multiplied by coefficients  $\lambda_i$  and summated. Importantly, these coefficients are chosen specifically so that the template annihilates polynomials of order  $k$  or less, thus where  $\text{Poly}^k$  is any polynomial of order  $k$  in  $\mathbb{R}^N$ ,

$$\sum_i \lambda_i \text{Poly}^k(\mathbf{h}_i + \mathbf{x}) = 0 \quad (5.9)$$

Thus if there is a non-stationary polynomial drift component to a random function  $Z$ , the allowable linear combination will annihilate or *filter* it. Intrinsic Random Functions of order  $k$  (IRF- $k$ ) are constructed using the above definition. An

intrinsic random function of order  $k$  is defined as a random process  $Z$  for which the ALC- $k$  measure is second order stationary. Thus the expected value of the ALC- $k$  is:

$$E\left[\sum_i Z(\mathbf{h}_i + \mathbf{x})\right] = 0 \quad (5.10)$$

and its variance is

$$\text{var}[Z(\mathbf{x})] = E[Z(\mathbf{x})^2] = \sum_{i,j} K(\mathbf{h}_j - \mathbf{h}_i) \quad (5.11)$$

where  $K(\mathbf{h})$  is the so-called *generalised covariance function*. Its form may be determined by examining the variance of some allowable linear combination – called the *generalised variogram*. What is intriguing about this theory, is that a common and useful class of ALC- $k$  is formed by the finite difference operators, which are used to estimate the partial derivatives in a grid. Indeed, a commonly used generalised variogram essentially consists of the variance of a difference operator, applied over a function's domain.

Equation (5.8) was illustrated schematically in Figure III—12, which is reproduced in Figure V—9 but now for different scales  $h$  of the Laplacian operator. The Laplacian template, described by

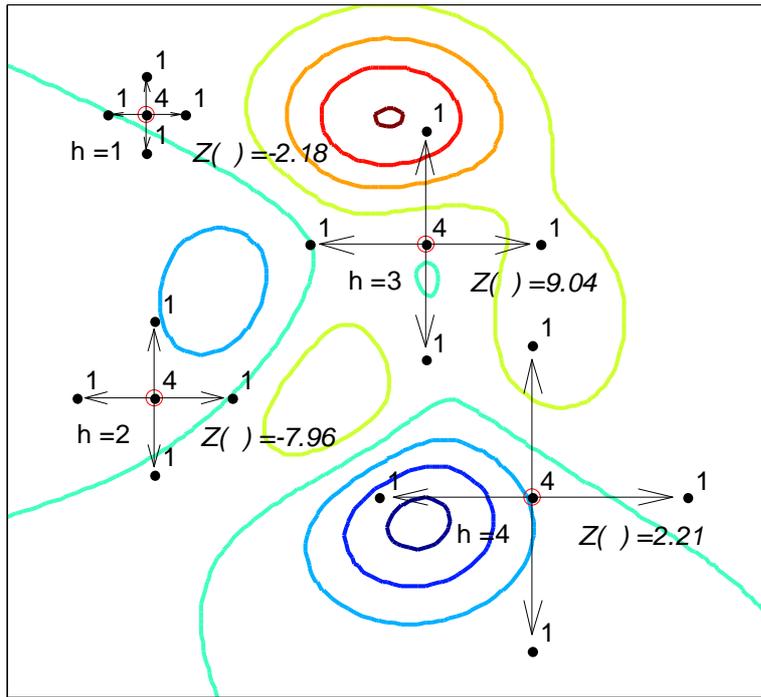
$$Z(\mathbf{x}) = Z(\mathbf{x} + h\hat{\mathbf{i}}) + Z(\mathbf{x} - h\hat{\mathbf{i}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) + Z(\mathbf{x} - h\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \quad (5.12)$$

is an ALC-1, which corresponds to the coefficients  $c_i = (1, 1, 1, 1, -4)$  and lag vectors  $\mathbf{h}_i = (h\hat{\mathbf{i}}, h\hat{\mathbf{j}}, -h\hat{\mathbf{i}}, -h\hat{\mathbf{j}}, \mathbf{0})$  in Equation (5.8). It is a relatively easy matter to prove that this metric filters linear, or first order, drifts in two dimensions. Note that the template  $\mathbf{h}_i$  on which it operates may be scaled by a factor  $h$  without affecting its ALC-1 properties.

Just as there are canonical forms for  $C(\mathbf{h})$  (Equation (3.23)), there are also canonical forms for  $K(\mathbf{h})$  (Equation (5.11)). For an isotropic IRF-1 a typical form is [84]

$$K(h) = C_0\Delta(h) - b_1h + b_2h^2 \log h + b_3h^3 \quad (5.13)$$

where  $\delta(h)$  is the nugget effect discontinuity at the origin (Equation (3.23)f) and where in  $\mathbb{R}^2$  the coefficients are subject to;



**Figure V—9: Laplacian ALC  $Z(\lambda^h)$  applied over different scales  $h$  on some function  $Z$ .**

$$C_0 \geq 0, \quad b_1 \geq 0, \quad b_3 \geq 0, \quad b_2 \geq -\frac{3}{2}\sqrt{b_1 b_3}. \quad (5.14)$$

If the variance of a particular ALC-1 is considered over different scales  $h$ , a generalised variogram can be calculated. Its form with respect to  $h$  is related to Equation (5.13) via Equation (5.11). Just as smooth behaviours were represented by the cubic, and in the limit Gaussian, models in Equation (3.23), smooth behaviours here manifest themselves in the cubic and logarithmic terms of Equation (5.13). In fact, the logarithmic term corresponds to a biharmonic spline model. The leading term in Equation (5.13) corresponds to a nugget effect, which characterises short-scale variability, and engenders spatial averaging in estimation.

Given some spatial discretisation of a boundary value problem, what is of interest is whether or how the covariance function reaches a steady form as the grid-spacing tends to zero. Whilst this has not been implemented on a large scale, it is still interesting even at a demonstration level as it may offer the numerical analyst another means of assessing the quality, or at least smoothness of the raw numerical result.

## V - 3. The Poisson Equation

To apply the above theory with respect to grid refinement, an even simpler problem was chosen – the conduction of heat over a square one-to-one domain modelled by the Poisson equation. This is an extremely basic elliptic PDE, but it is useful as one may expect that if any quasi-random behaviours can be imputed to the Poisson equation, such behaviour can only be more complicated and ostensibly aleatoric for the more complex transport equations. A similar approach as was pursued in Section V - 1 shall be pursued in this section. The FD solution of the Poisson equation over a differentially heated square domain under two different Neumann boundary conditions is examined under successive refinement, but this time using the more general and higher order description of the covariance function  $\mathbf{K}(\mathbf{h})$  introduced in Section V - 2.

### V - 3.1. Problem description

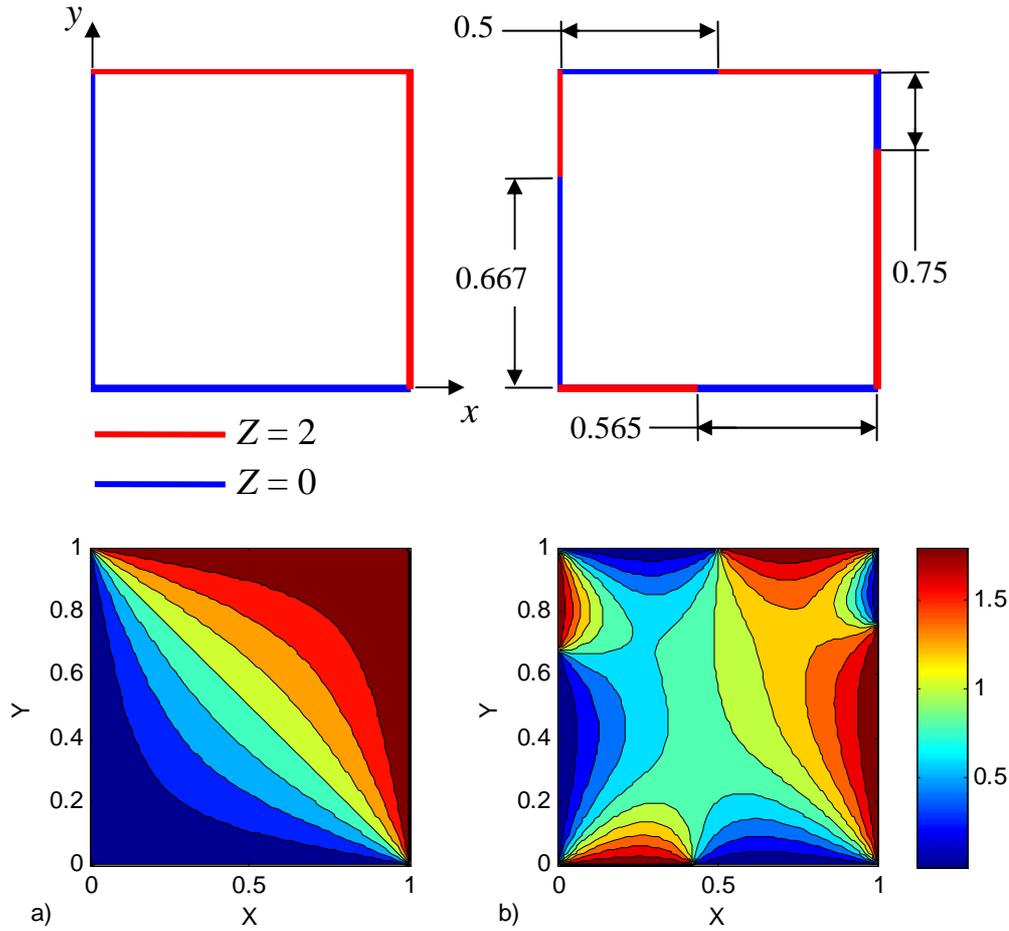
Where  $Z(\mathbf{x})$  is taken to be temperature and  $Y(\mathbf{x})$  is a non-dimensionalised source term, the steady state diffusion of heat over a two-dimensional domain without convection is often approximated by:

$$\frac{\partial^2 Z}{\partial x^2} + \frac{\partial^2 Z}{\partial y^2} = Y \quad (5.15)$$

This equation was discretised using the central difference formula

$$Z(\mathbf{x} + g\hat{\mathbf{i}}) + Z(\mathbf{x} - g\hat{\mathbf{i}}) + Z(\mathbf{x} + g\hat{\mathbf{j}}) + Z(\mathbf{x} - g\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \Big|_{\mathbf{x}_k} = Y(\mathbf{x}) \Big|_{\mathbf{x}_k} \quad (5.16)$$

on a homogenous, equally-spaced  $N \times N$  grid of nodes spaced at intervals of  $g$ . The side lengths of the one-to-one square domain were scaled non-dimensionally to unity. Neumann boundary conditions then specified the normalised temperature at  $4(N-2)$  nodes at the edges so that Equation (5.16) could be written at the remaining  $(N-2)^2$  nodes in the grid, yielding  $(N-2)^2$  equations and as many unknowns. The source term  $Y(\mathbf{x})$  was set to zero and two sets of boundary conditions were considered, resulting in the two cases described in Figure V—10 in which the resulting solutions for  $Z$  are also presented.



**Figure V—10: Boundary conditions and corresponding contour plots of  $Z$  for; (a) Case 1, (b) Case 2.**

Clearly, the left hand side of Equation (5.16) also forms a Laplacian template which, as has been noted previously, is an ALC-1. The variance of this ALC at any given scale  $h$

$$\Gamma(h) = \text{var} \left[ Z(\mathbf{x} + h\hat{\mathbf{i}}) + Z(\mathbf{x} - h\hat{\mathbf{i}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) + Z(\mathbf{x} - h\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \right] \quad (5.17)$$

is called the generalised variogram, and its behaviour for different  $h$  can now be examined over the domain of the numerical solution articulated by Equation (5.16). Note that in particular, when  $h = g$ ;

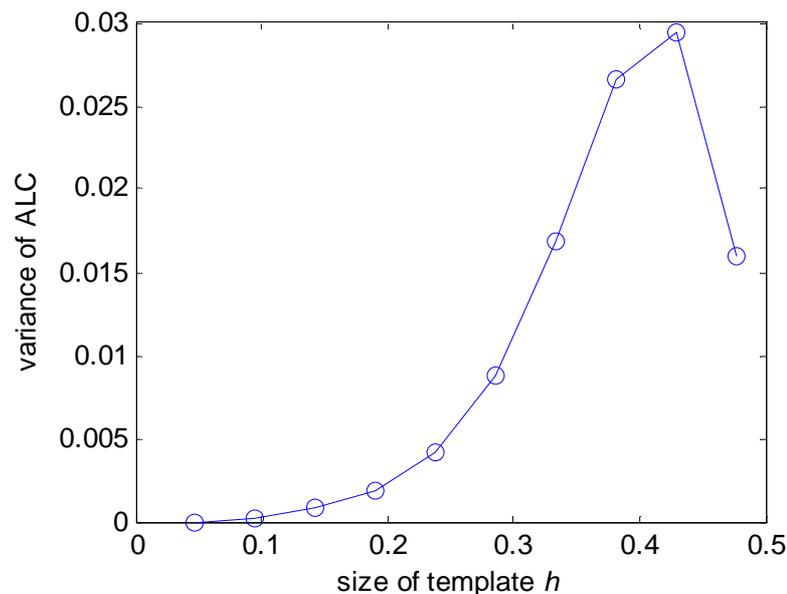
$$\begin{aligned} \Gamma(g) &= \text{var} \left[ Z(\mathbf{x} + g\hat{\mathbf{i}}) + Z(\mathbf{x} - g\hat{\mathbf{i}}) + Z(\mathbf{x} + g\hat{\mathbf{j}}) + Z(\mathbf{x} - g\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \right] \\ &= \text{var} [Y(\mathbf{x})] \end{aligned} \quad (5.18)$$

because of the relation in Equation (5.16). Thus because  $Y$  is zero over the domain, the variance of the Laplacian at  $h = g$  (one grid-spacing) is always zero, thus

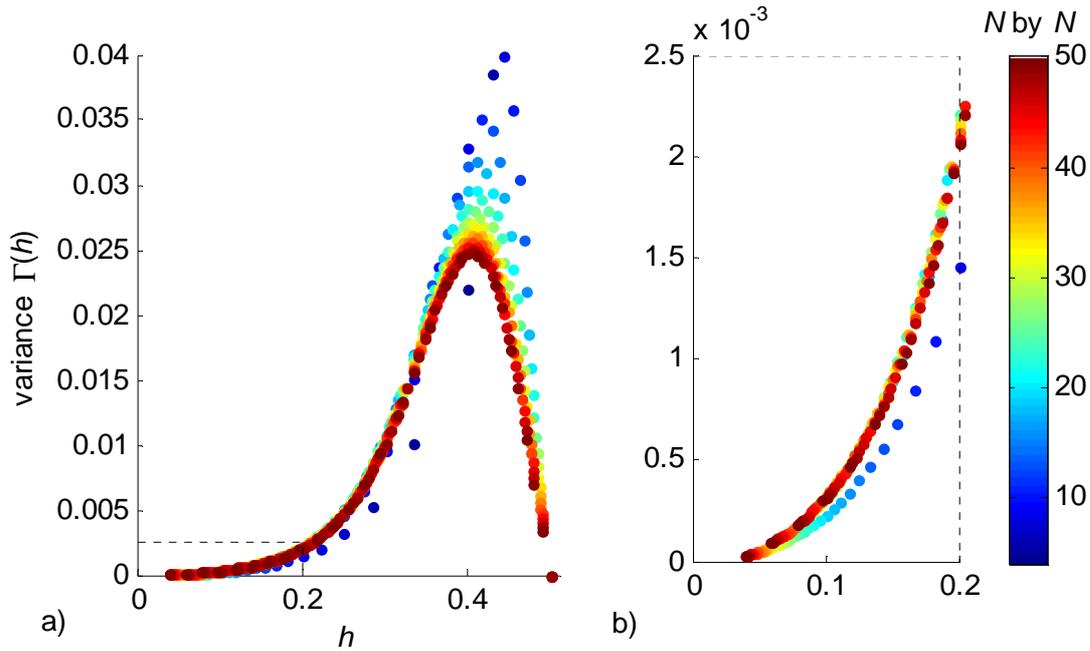
$\Gamma(g)=0$ . However, were this statistic to be calculated over the same lag  $g$  on the *exact* solution to Equation (5.15), which is free of truncation errors, obviously one would find that  $\Gamma(g)\neq 0$  because the exact solution must still vary by a finite amount over finite distances  $g$ . Indeed, the exact solution displays variation even at microscopically small scales  $h < g$ . What must be examined then is the variation of Equation (5.17) at larger scales  $h > g$ , which are not constrained to be zero by the discretisation. For sufficiently small scales  $h$ , the variance in Equation (5.17) always goes to zero – of particular interest is how it does this at slightly larger scales than  $g$ , especially as the grid is refined.

### V - 3.2. Generalised covariance behaviour

Discrete solutions of Equation (5.15) with  $Y = 0$  over domains with the boundary conditions exemplified in Figure V—10 were produced, using the discretisation in Equation (5.16). A range of grid-sizes was used – a  $4\times 4$  grid, a  $5\times 5$  grid, and so on up to a  $50\times 50$  grid – and the generalised variogram over different extents  $h$  was calculated for each to examine the behaviour of higher order spatial statistics as the numerical scheme converged.



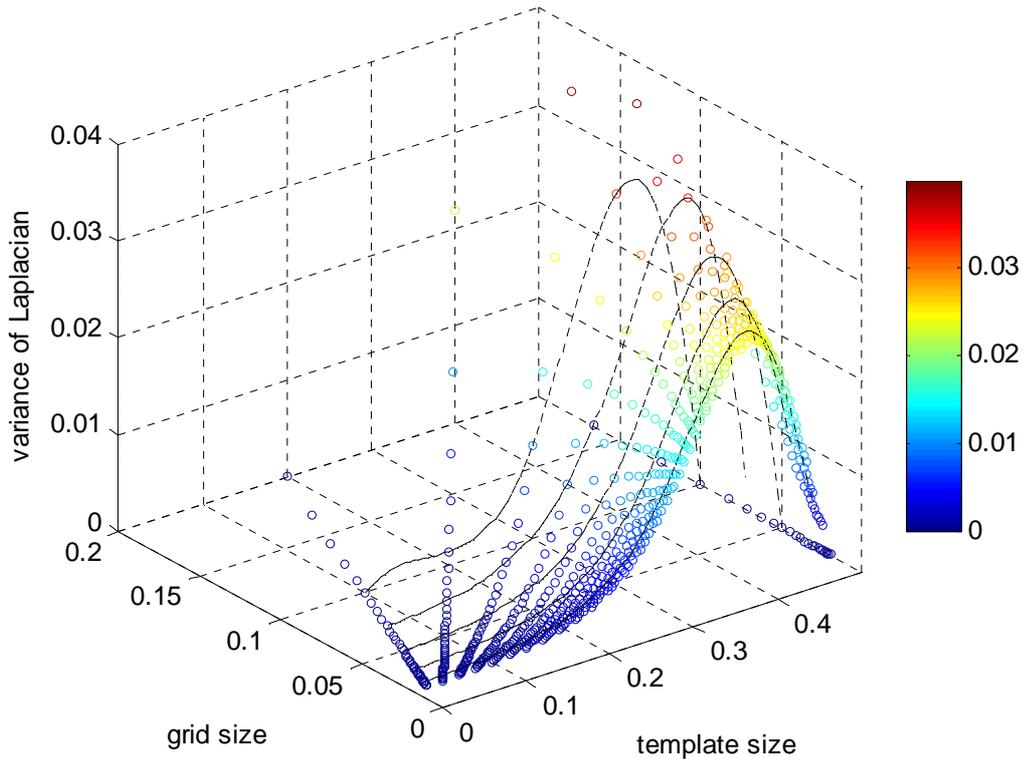
**Figure V—11: Variance of Laplacian template  $\Gamma(h)$  for the boundary conditions in case one, with  $20\times 20$  grid nodes.**



**Figure V—12(a) Generalised variogram on a Laplacian template at  $N \times N$  grid-size indicated by the spectrum (case one), (b) inset of part (a).**

The variance of the Laplacian  $\Gamma(h)$  is presented in Figure V—11 over different scales  $h$  calculated from the discrete solution employing a  $20 \times 20$  grid over the simpler ‘case one’ boundary conditions. At a given scale  $h$ , this variance statistic is calculated over all the nodes on the grid where a template of size  $h$  may fit – note that the scale  $h$  must be a multiple of grid spacing  $g$ . As already noted, when  $h = g$  the variance is zero. It increases for  $h > g$  but then falls off as the scale of the template approaches that of the boundary conditions,  $h \sim 1$ . Thus, the variance is constrained microscopically by the physical equations, and macroscopically by the far-field boundary conditions. Figure V—12 presents the same basic plot as Figure V—11, but for multiple grid-sizes, where each plot has been overlaid and coloured according to grid-size as indicated by the spectrum on the right. This figure is reproduced three-dimensionally in Figure V—13 where instead the variance magnitude is the contoured variable.

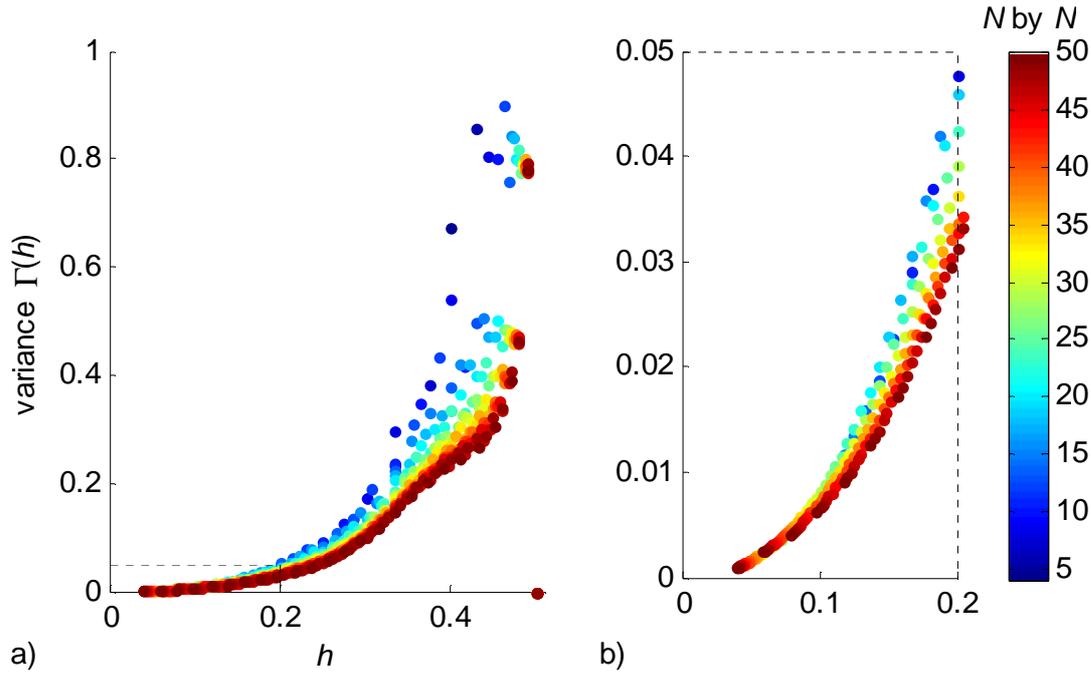
A first observation of Figures V—12 and V—13 is that the points on the generalised variogram seem to be approaching a steady structure as the grid is refined. A second observation, is that away from the origin the confidence with which these points are calculated must be reduced, because as template size increases there are fewer grid nodes at which the ALC can be calculated – there are fewer ‘samples’. Generally one should be more confident of the statistics generated over



**Figure V—13: Generalised variogram on a Laplacian template in 3D where variance is the contour variable (case one).**

finer grids for the same reason. Regarding the first observation more specifically, in Figure V—12 the movement towards the ‘true’ generalised variogram is monotone in the vicinity of the origin,  $h=0$ . With the exception of statistics generated from the very coarse grids at the scale of the boundary conditions, the final structure in red is approached from one side. Note that the variance over the smallest possible template  $\Gamma(h=g)$  must always underestimate the asymptotic structure as it is zero. Should there be a ‘true’ structure that the statistics on  $\Gamma(h)$  are tending towards under grid refinement, it is of immediate interest what it is, and how closely the generalised variograms calculated from discrete approximations of the solution of Equation (5.15) approach it.

However, imputing any sort of confidence *interval* on the statistics generated above depends entirely on the distribution of the ALC from which the variance is calculated. This is presented in Figure V—15 for the finest grid ( $50 \times 50$ ) at  $h=2g$  and  $h=3g$ . It is immediately apparent that these are not normal distributions, and neither really should one expect them to be. However, at least the distributions are centred around zero, which is expected given the problem’s symmetry, and for

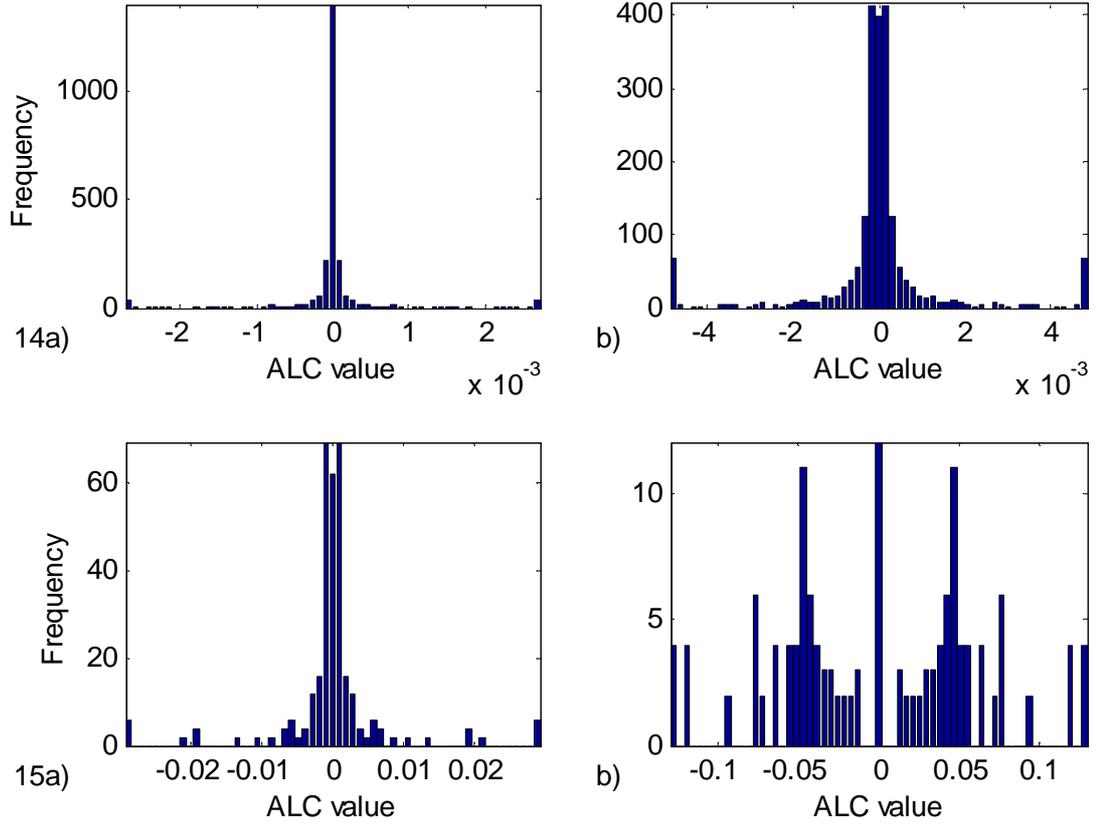


**Figure V—14(a) Generalised variogram on a Laplacian template at  $N \times N$  grid-size indicated by the spectrum (case two), (b) inset of part (a).**

$h = 2g$  there is a single strong peak – the variance therefore provides a meaningful measure of spread for histograms generated at small scales of  $h$ . The gradual emergence in Figure V—15(b) and Figure V—16(a) (where the  $20 \times 20$  grid was used) of two peaks is almost certainly again due to the strong symmetry of what is, an artificially simple numerical example. The departure from a central peak is of course more pronounced at larger scales, as seen in Figure V—16(b).

To study more precisely how the functions  $\Gamma(h)$  approach zero, especially with respect to grid refinement, a canonical form is assumed for the underlying generalised covariance function  $\mathbf{K}(\mathbf{h})$ . Where particularly,  $\mathbf{e}_i = (1, 1, 1, 1, -4)$  and  $\mathbf{h}_i = (h\hat{\mathbf{i}}, h\hat{\mathbf{j}}, -h\hat{\mathbf{i}}, -h\hat{\mathbf{j}}, \mathbf{0})$  (the Laplacian template), Equation (5.11) can be used to relate the function  $\Gamma(h)$  and  $\mathbf{K}(\mathbf{h})$ :

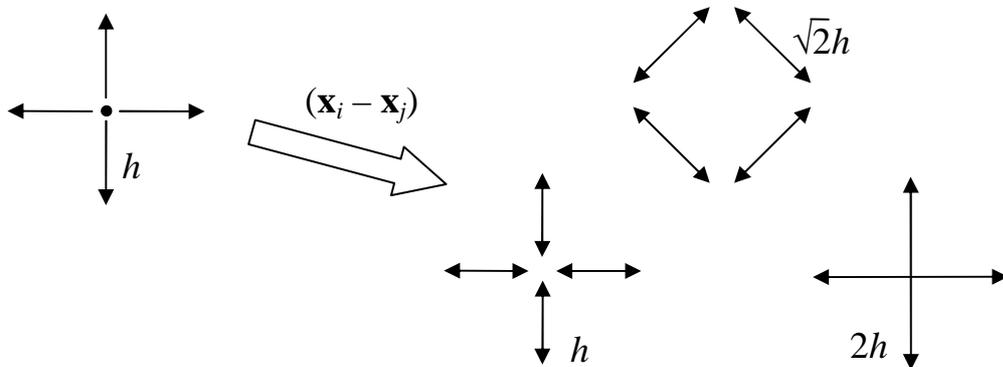
$$\begin{aligned} & \text{var} \left[ Z(\mathbf{x} + h\hat{\mathbf{i}}) + Z(\mathbf{x} - h\hat{\mathbf{i}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) + Z(\mathbf{x} - h\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \right] \dots \\ & \equiv \text{var} \left[ \sum_i \mathbf{e}_i Z(\mathbf{x} + \mathbf{h}_i) \right] = \sum_{i,j} \mathbf{e}_i \mathbf{e}_j \mathbf{K}(\mathbf{h}_j - \mathbf{h}_i) \end{aligned} \quad (5.19)$$



**Figures V—15 and V—16: ALC frequency distributions for 50×50 and 20×20 respectively; (a) at  $h = 2g$ , (b) at  $h = 3g$  for 50×50 and  $h = 5g$  for 20×20.**

Note that this expression applied to the Laplacian template will produce lag vectors of  $\pm g\hat{\mathbf{i}}, \pm g\hat{\mathbf{j}}, \pm 2g\hat{\mathbf{i}}, \pm 2g\hat{\mathbf{j}}$  and  $[(\pm g\hat{\mathbf{i}}) + (\pm g\hat{\mathbf{j}})]$  in the final line, as illustrated in Figure V—17. Following Chilès and Delfiner [84] (p. 276-279), it is assumed that the generalised covariance  $K(\mathbf{h})$  is isotropic, thus;

$$K(\mathbf{h}) = K(\|\mathbf{h}\|) = K(h) \tag{5.20}$$



**Figure V—17: Lag vector generation on the Laplacian.**

$N$	$B_1$	$B_3$	$N$	$B_1$	$B_3$	$N$	$B_1$	$B_3$
6	-0.03909	0.70298	21	-3.51e-4	0.28912	36	3.123e-5	0.35469
7	-0.03048	0.68626	22	-2.82e-4	0.29145	37	3.690e-5	0.36018
8	-0.01983	0.58806	23	-2.25e-4	0.29433	38	4.183e-5	0.36572
9	-0.01258	0.49621	24	-1.80e-4	0.29766	39	4.612e-5	0.37132
10	-0.00814	0.42741	25	-1.42e-4	0.30134	40	4.986e-5	0.37697
11	-0.00544	0.37922	26	-1.10e-4	0.30534	41	5.311e-5	0.38266
12	-0.00376	0.34621	27	-8.39e-5	0.30958	42	5.595e-5	0.38840
13	-0.00267	0.32380	28	-6.17e-5	0.31404	43	5.841e-5	0.39417
14	-0.00197	0.30872	29	-4.30e-5	0.31869	44	6.055e-5	0.39997
15	-0.00148	0.29877	30	-2.70e-5	0.32349	45	6.240e-5	0.40581
16	-0.00113	0.29245	31	-1.33e-5	0.32844	46	6.400e-5	0.41167
17	-8.81e-4	0.28875	32	-1.62e-6	0.33350	47	6.538e-5	0.41756
18	-6.93e-4	0.28698	33	8.461e-6	0.33867	48	6.657e-5	0.42348
19	-5.50e-4	0.28667	34	1.716e-5	0.34393	49	6.758e-5	0.42942
20	-4.39e-4	0.28746	35	2.470e-5	0.34927	50	6.843e-5	0.43538

**Table V-1: Fit coefficients  $B_1$  and  $B_3$  for case one,  $N \times N$  grid.**

so there are effectively only three lag magnitudes;  $g$ ,  $2g$  and  $\sqrt{2}g$ . The same procedure as that used by Chilès and Delfiner [84] (pp. 276-279) for the one-dimensional case is adopted to arrive at the following expression for the left hand side of Equation (5.19):

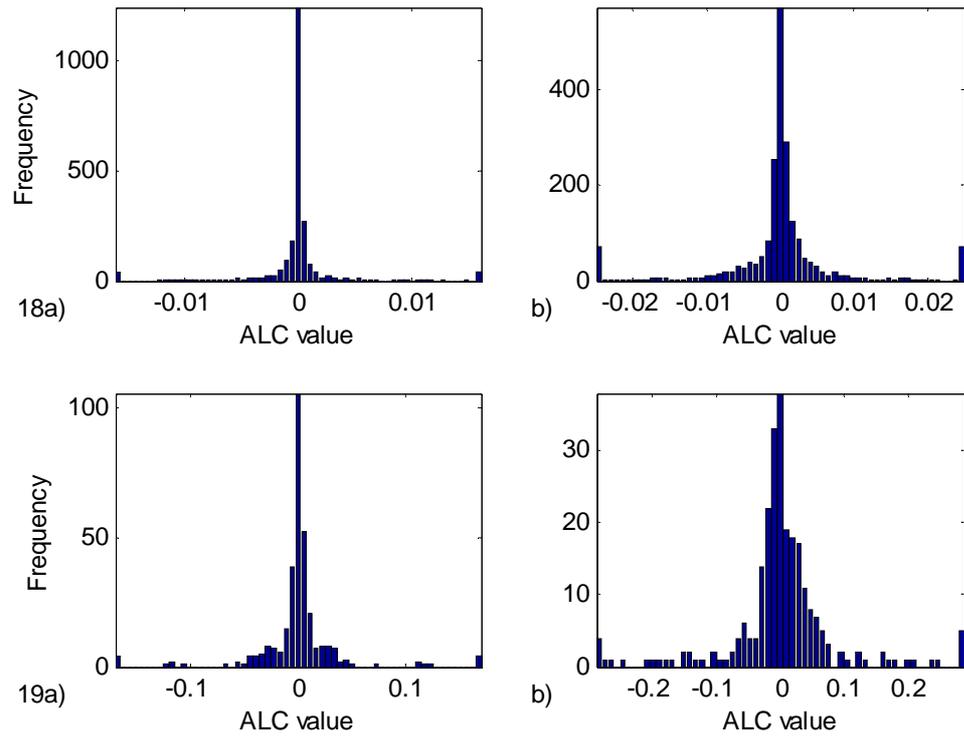
$$\begin{aligned} & \text{var} \left[ Z(\mathbf{x} + h\hat{\mathbf{i}}) + Z(\mathbf{x} - h\hat{\mathbf{i}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) + Z(\mathbf{x} - h\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \right] \\ &= 20K(0) - 32K(h) + 4K(2h) + 8K(\sqrt{2}h) \end{aligned} \quad (5.21)(a)$$

Thus by the definition in Equation (5.17);

$$\Gamma(h) = 20K(0) - 32K(h) + 4K(2h) + 8K(\sqrt{2}h) \quad (5.21)$$

which relates the generalised variogram to the generalised covariance function  $K(h)$ . Combining this relation with the canonical form in Equation (5.13), one obtains the final relation;

$$\frac{\Gamma(h)}{20} = C_0(1 - \Delta(h)) + \frac{6 - 2\sqrt{2}}{5} b_1 h + \frac{4}{5} \log(2\sqrt{2}h) b_2 h^2 + \frac{4\sqrt{2}}{5} b_3 h^3. \quad (5.22)$$



**Figures V—18 and V—19: ALC frequency distributions for 50×50 and 20×20 respectively; (a) at  $h = 2g$ , (b) at  $h = 3g$ .**

for which the complete derivation is presented in Appendix D - 9. Equation (5.22) is a canonical form that can be fitted to the statistics generated in Figures V—12, V—13 and V—14 by tuning the parameters  $C_0$ ,  $b_1$ ,  $b_2$  and  $b_3$ .

To make clearer inferences, a simplified version of Equation (5.22) has been fitted to the statistics generated in Figures V—12 and V—14. It was mentioned before that both the FD and FV methods make assumptions about how the actual solution varies, but unlike the FE method they do not propose an explicit form for the solution between the nodes. Therefore, how the calculated nodal values ought to be interpolated is in some senses an open question. It is reasonable to expect that such an interpolation should at least pass through the nodal values – flawed as they are by truncation error. For this reason, the nugget effect term  $C_0$  in Equations (5.13) and (5.17) is considered to be zero. Furthermore, as the logarithmic and cubic terms double up inasmuch as they both model smoothly varying behaviours, only the cubic term has been considered for simplicity. Both of these models display zero-slope approaching the origin, but the logarithmic term is undefined at the origin. The model that is fitted to the statistics is then, simply;

$$\Gamma(h) = B_1 h + B_3 h^3 \quad (5.23)$$

for which the coefficients have been amalgamated in  $B_1$  and  $B_3$ . The results in Equation (5.22) of the full derivation in Appendix D - 9 were required as the result in Chilès and Delfiner [84] is for the one dimensional case only. The form in Equation (5.18) is fitted to the  $\Gamma(h)$  statistics generated at successive levels of grid refinement on the ‘case one’ boundary conditions – presented in Figure V—12, and the coefficients  $B_1$  and  $B_3$  are reported in Table V-1. Only two points are required to fit Equation (5.23), so for simplicity and to reflect the greater confidence in statistics that are close to the origin, the statistics occurring at  $\Gamma(2g)$  and  $\Gamma(3g)$  have been used, where  $g$  is the grid spacing. Naturally,  $\Gamma(g)$  is *not* used as it is always zero because of the constraint on variation imposed by the solution of Equation (5.16) at that scale.

As expected, in Table V-1 it is apparent that the cubic term eventually dominates and the linear term becomes vanishingly small as the grid is refined – the numerical solution becomes ‘smoother’. Furthermore, the constraint  $B_1 > 0$  (Equation (5.14)) is not satisfied for the coarser grids which would seem to indicate that the fit of the generalised covariance function for them is rather bad. However, because the variance over the shorter lags reduces so quickly for the simpler case one boundary conditions, it is difficult to draw many conclusions regarding the solution

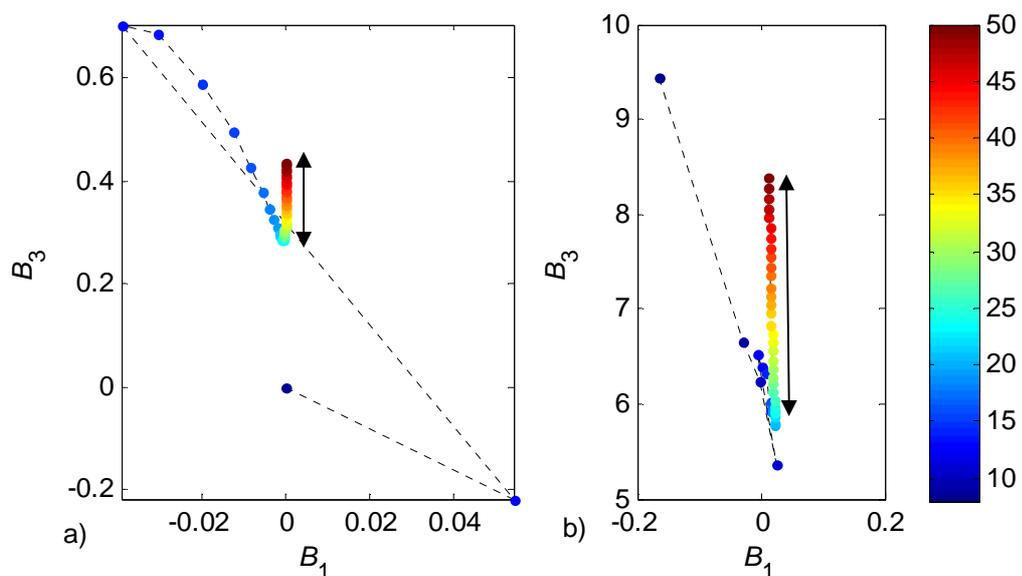
$N$	$B_1$	$B_3$	$N$	$B_1$	$B_3$	$N$	$B_1$	$B_3$
6	-0.74858	14.9469	21	0.021832	5.81140	36	0.016435	6.95370
7	-0.26622	9.16130	22	0.021516	5.86996	37	0.016101	7.04544
8	-0.16285	9.43564	23	0.021215	5.92388	38	0.015775	7.13935
9	-0.02756	6.65721	24	0.021451	5.91626	39	0.015454	7.23611
10	-0.00335	6.23174	25	0.021088	5.97248	40	0.015101	7.35192
11	0.025803	5.36064	26	0.020622	6.04796	41	0.014807	7.44900
12	-0.00632	6.52265	27	0.020144	6.12987	42	0.014520	7.54811
13	0.001267	6.43030	28	0.019760	6.20658	43	0.014245	7.64702
14	0.006880	6.32063	29	0.019359	6.27916	44	0.013952	7.75935
15	0.015678	5.93857	30	0.018877	6.37436	45	0.013694	7.86025
16	0.015713	6.01229	31	0.018488	6.45240	46	0.013445	7.96172
17	0.017415	5.98410	32	0.018015	6.56150	47	0.013204	8.06384
18	0.019261	5.91936	33	0.017603	6.65549	48	0.012957	8.17359
19	0.019745	5.93546	34	0.017233	6.74312	49	0.012731	8.27679
20	0.021795	5.78336	35	0.016876	6.83113	50	0.012512	8.38092

**Table V-2: Fit coefficients  $B_1$  and  $B_3$  for case two,  $N \times N$  grid.**

convergence; the coefficient  $B_1$  quickly assumes an order of  $10^{-5}$ .

To make more meaningful assessments, a second scenario was examined, for which boundary conditions were presented in Figure V—10(b) – referred to as case two. These conditions were deliberately asymmetrical and present a slightly more complicated problem for which one would expect slower convergence. This is reflected in higher values for the corresponding variance plot in Figure V—14, and also for the coefficients describing it in Table V-2. Notably for this second case, perhaps because there is greater spatial variation and at smaller scales, it is clear that the proposed model (5.23) does not fit the generalised variogram well at all until grid size has progressed beyond  $15 \times 15$ . Up to this point, the conditions in Equation (5.14) are not well satisfied and unlike for Table V-1 their infraction is not relatively small. Therefore if the positivity conditions were to be enforced, the model could not exactly interpolate the statistics at  $\Gamma(2g)$  and  $\Gamma(3g)$ . Nevertheless as the solution converges, a tendency towards smoother cubic behaviour is again demonstrated. Frequency histograms of the actual ALC variation over the case two solution domain are presented in Figures V—18 and V—19, again for the  $50 \times 50$  and  $20 \times 20$  grids. These are now only symmetrical in the limit as  $g \rightarrow 0$ , because of the fundamentally asymmetrical boundary conditions. However the tendency at small separations is still towards zero centred distributions with heavy tails.

The behaviour of the coefficients  $B_1$  and  $B_3$  as the numerical solution



**Figure V—20: Convergence plot for coefficients  $B_1$  and  $B_3$ ; (a) case one and (b) case two boundary conditions.**

converges upon the exact solution is presented graphically in Figure V—20. As previously noted the linear coefficient approaches zero at higher resolutions. The present author surmises that when it starts to do this in an orderly, monotonous manner, the discrete solution is within the asymptotic range of the numerical method – as indicated by the arrows in Figure V—20. It appears that the cubic coefficient  $B_3$ , is not approaching a fixed value in the same way as its counterpart  $B_1$ . This is probably because the actual covariance structure towards which these statistics are tending is not *quite* a simple cubic form. If this is the case, the coefficient  $B_3$  will scale in some way with  $g$  as only two successive points at  $h = 2g$  and  $h = 3g$  are considered for the fit and  $g$  itself is approaching zero as the grids become finer. An obvious way around this problem is simply to consider a parametric function of best fit of points considered to be within some static neighbourhood of the origin. Alternatively a more complex canonical form perhaps for which  $b_2 \neq 0$  may also remedy this scaling effect. However as a preliminary investigation, the behaviour of  $B_1$  and  $B_3$  certainly suggests that some ‘true’ covariance structure is being converged upon as grid-size tends to zero.

## V - 4. Afterword

The foregoing discussion is intriguing, but lends itself to a number of interpretations. Certainly, at a basic level what is being observed is an artefact of the smoothness of the FD solution and the original PDEs. However, it would seem that there is a more interesting possibility of a general link between stationarity as expressed by Equation (5.11), the observed behaviour, and the convergence of the numerical solution.

Whilst it is not really surprising that the variance of the Laplacian ALC-1 approaches zero as the grid size is reduced – it must because of the PDEs, the precise manner in which it does this is of interest. Firstly, this variance is constrained to zero at one grid spacing which cannot be true of the actual solution. Thus from the outset, stationary covariance structures of a discrete solution will always differ from those of the true solution, for which there must be some variation over a finite sized template. Also, it is apparent from the plots and tables of coefficients  $B_1$  and  $B_3$ , that after an

initial settling phase the tendency of the generalised variogram is then to approach a ‘smoother’ behaviour, which comprises progressively less of the linear model. During the initial settling phase, a simple model is generally unreliable, as an exact fit cannot be obtained without the violation of the positive definite constraints in Equation (5.14). Considering these behaviours, it is reasonable to wonder if a more general measure of grid convergence can be formed by examining the goodness-of-fit of a simple covariance model, and the behaviours represented therein. Note that, should a simple model be unsatisfactory – it was remarked that the underlying ‘true’ generalised variogram did not seem to be an entirely cubic model, there are more exotic, and completely general spectral representations of the generalised covariance function  $K(\mathbf{h})$  [84].

The term ‘covariance’ is used very loosely in the foregoing discussion. The distributions in Figures V—15, V—16, V—18 and V—19 are not generated by a statistical process, but by deterministic processes. Rather, they may be viewed as the distributions of residual error over different scales. The variance of the Laplacian template at a given separation  $h$  is indicative of how much better the grid performs because of the extra levels of discretisation between  $h$  and  $g$  node-spacing. Without these levels of discretisation the variance of this particular ALC-1 at  $h$  would be zero. This last observation is serendipitous insofar as the Laplacian template aligned with the grid coordinates produces this result by virtue of Equation (5.16). However, the Laplacian is by no means the only ALC-1. If the numerical result is loosely regarded to behave as an IRF-1, the theory states that an IRF-1 is stationary for *all* ALC-1, and so the next step would be to examine the variation of other ALC-1, besides the Laplacian template. A simple variation might be to rotate the Laplacian, or to consider completely different ALC-1, such as the square arrangement in Figure V—21. All of these would be enough to render the variation at the smallest separation – set by grid-size, non-zero. However, one would expect that because of the larger modelled PDEs, the same result should still arise. Certainly the rotated Laplacian merely represents the equations in a rotated coordinate frame, so in the limit as the solution is converged upon, one should expect the same behaviour.

It is interesting to consider what would happen, if the situation were changed ever so slightly and the source term in Equation (5.15) were non-zero; say, a non-random distribution  $Y(\mathbf{x})$  over the domain. Obviously, the distributions in Figures V—15(a), V—16(a), V—18(a) and V—19(a) for  $h = 2g$  will no longer necessarily be centred around zero. At this scale, these functions approximate a single peak at zero which is effectively the distribution of residuals at  $h = g$ . Were there a non-zero source term, the distribution for  $h = g$  would then exactly mirror that of the function  $Y(\mathbf{x})$  over the domain, and the distributions at small lags  $h \sim g$  would resemble the distribution of  $Y(\mathbf{x})$  convolved with some residual error distribution. The plots of variance then in Figures V—12 and V—14 would still converge but now to an offset value: the variance of  $Y(\mathbf{x})$  over the domain. The behaviour at the origin of the generalised variogram would also be similar, but again offset positively in the vertical direction by the variance of  $Y(\mathbf{x})$  over the domain.

This may lead one to the conclusion that it is only the underlying PDEs in the particular case of the heat conduction equation, that cause a movement towards smoother behaviours – as articulated by the generalised variogram, at higher grid resolutions. Any departures from these equations require modification of the interpretation of the generalised variogram. However, theoretically at least the tendency towards a central distribution may be restored by either using a higher order ALC or correcting or detrending the ALC-1 with the known source term. The latter would amount to the consideration of the residuals;

$$\text{var} \left[ Z(\mathbf{x} + h\hat{\mathbf{i}}) + Z(\mathbf{x} - h\hat{\mathbf{i}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) - 4Z(\mathbf{x}) - Y(\mathbf{x}) \right]$$

and the exact nature of the former would depend on the form of  $Y(\mathbf{x})$ . If this source

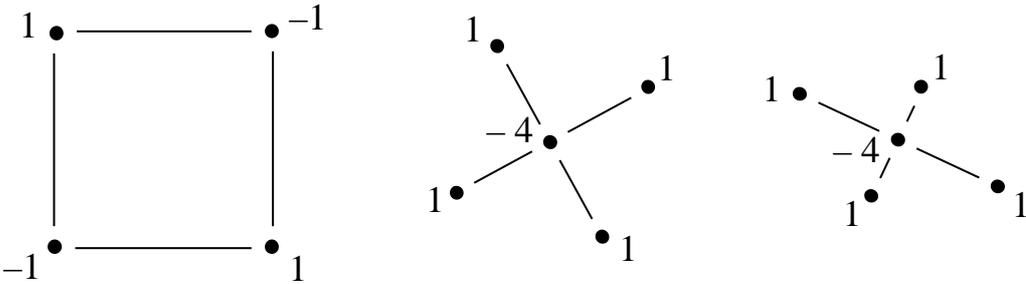


Figure V—21: Alternative ALC-1 templates.

term were to be represented by a polynomial of order  $k$  over the domain, it is expected that an ALC- $(k+1)$  will render  $Z(\mathbf{x})$  stationary. For example, the above investigation has been performed using the Laplacian, an ALC-1, to render conduction on a constant (order zero, ALC-0) source term, stationary.

The reason for using ALCs in the first place is to filter the existence of polynomial drifts: they provide a means of looking at the spatial data that renders them stationary. Curiously, this is exactly what the physical equations and their discretisation also attempt to do. Examining the residuals in a mesh with respect to the actual or a 'better' solution ought – if the mesh is well constructed, to reveal a zero-centred distribution of sorts. If there are peaks away from the origin it can only mean that the mesh is performing badly, as there are then more underperforming cells than cells that are closely approximating the actual solution. Given that the discretised equations always provide a way of rendering the numerical data stationary – at least at small scales, it may be possible to use modified or extended versions of the ALC to do the same thing. Examining the variation of ALCs on either  $Y$  or  $Z$  in isolation does not make much sense, but a stationary increment of sorts can be formed by considering the two of them together. There do exist cross-covariant equivalents of the generalised covariance, and these may be of interest for this reason.

There is no aleatoric uncertainty in the numerical result, but there is certainly epistemic uncertainty: it is impossible to know what the higher order error terms are without reference to a finer discretisation. Furthermore, because error must be zero at the boundaries of the boundary value problem, the error function must be in some respects or at some order, stationary. In this light, the statistical treatment of numerical field variables assumes an almost Bayesian interpretation wherein probability is a degree of confidence, based on prior experiences – or in this case, locations.

# Chapter VI - Case Study 2: Inverted wing and rolling wheel

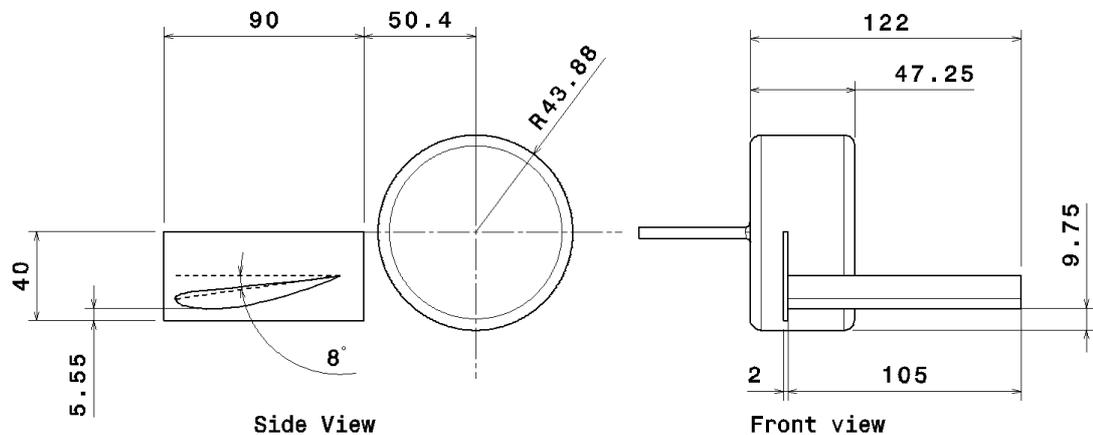
The primary case study that has been addressed in this work was provided by Sam Diasinos. Diasinos has conducted an extensive study on the aerodynamic characteristics of a wheel rolling in the wake of an inverted wing [113-116] – a configuration typical to many open wheel formulas in motor racing. This study concentrated particularly on the characterisation of the aerodynamic interactions between the wheel and the wing, and the development of a practical numerical model for the flow physics. To these ends, Diasinos constructed an experimental scale model around which the three-dimensional flow-field could be surveyed using Laser Doppler Anemometry. These results were compared with steady-state numerical simulations of the scale model employing a variety of turbulence models, with a view to choosing an appropriate model.

This is a classic example of an ‘arbitrary’ comparison of spatial data: the selection of a turbulence model relies heavily on the intuition and understanding of the practitioner, as unfortunately the flow physics is *not* really well approximated by

the model. The present author aims to use the covariance modelling schemes outlined in “Chapter IV - Implementation” to provide some indication of spatial correlation – thus providing a quantitative means of comparison. The development of meaningful relationships between the datasets consequently influences the interpretation of the spatial data themselves. Especially in engineering applications, each dataset informs a broader picture of the flow physics – the ‘truth’ is imagined to be situated somewhere amongst them, and the analyst’s conception of it will change as new datasets challenge previous hypotheses. With this in mind, one might ask further how the results may be blended and the datasets interpolated, to arrive at a best estimate of the whole-field behaviour. This is of course easy to do on an ad-hoc basis, but cokriging offers a means of blending results that also bears theoretical justification.

## VI - 1. A Validation Scenario

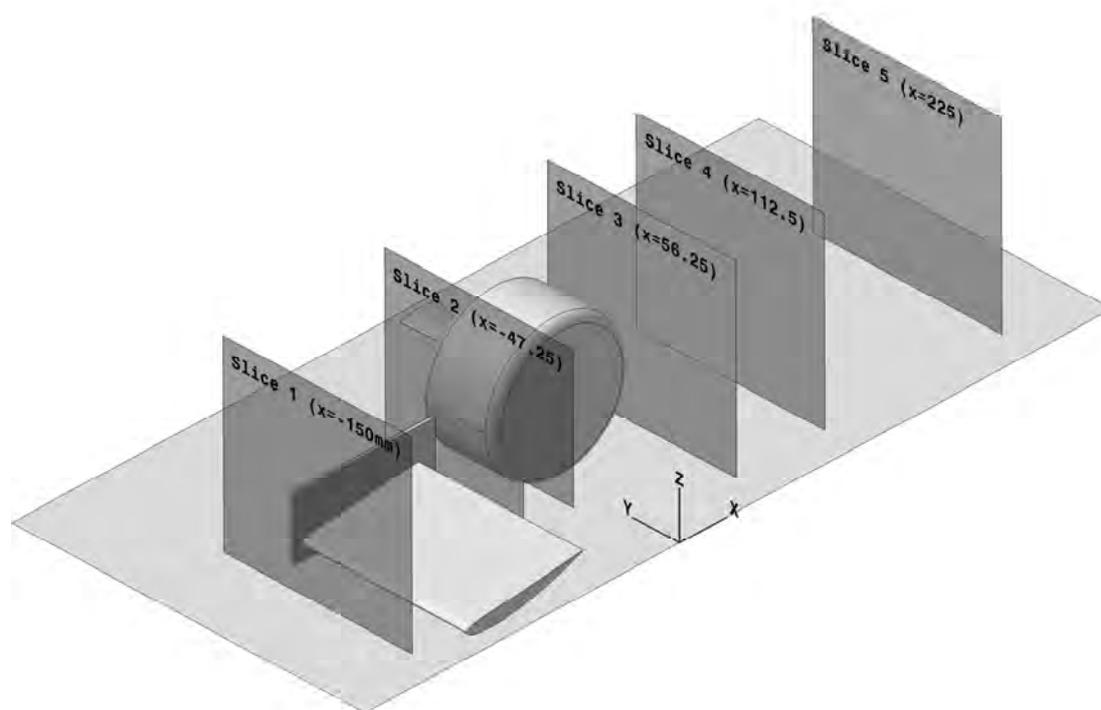
Diasinos has built and tested an experimental scale-model of a wheel and inverted wing configuration in a moving ground wind-tunnel. The model consists of a rolling wheel driven by the moving ground and held in place by a slender, streamlined sting running outboard of the wheel-wing configuration (‘outboard’ is considered to lie in the positive  $y$ -direction denoted in Figure VI—2). The wheel rolls in the wake of a single element, constant chord, NACA 4412 aerofoil section with no sweep or taper. This wing is cantilevered from the inboard wall of the wind tunnel and has a chord of 75mm, an angle of attack of  $8^\circ$  and a span of 105mm. The lowest point on the wing rides 9.75mm above the moving ground, and the wheel axle



**Figure VI—1: Schematic drawing of wheel-wing geometry, all dimensions in mm.**

is positioned 50.4mm leeward of the trailing edge of the endplate. The wheel itself is 47.25mm wide with a diameter of 87.75mm and its outboard face (to which the sting is connected) tracks 15mm further outboard into the wind tunnel than the endplate of the wing. For full dimensional details the reader is referred to Diasinos [113-115], a sketch of the model is provided in Figure VI—1.

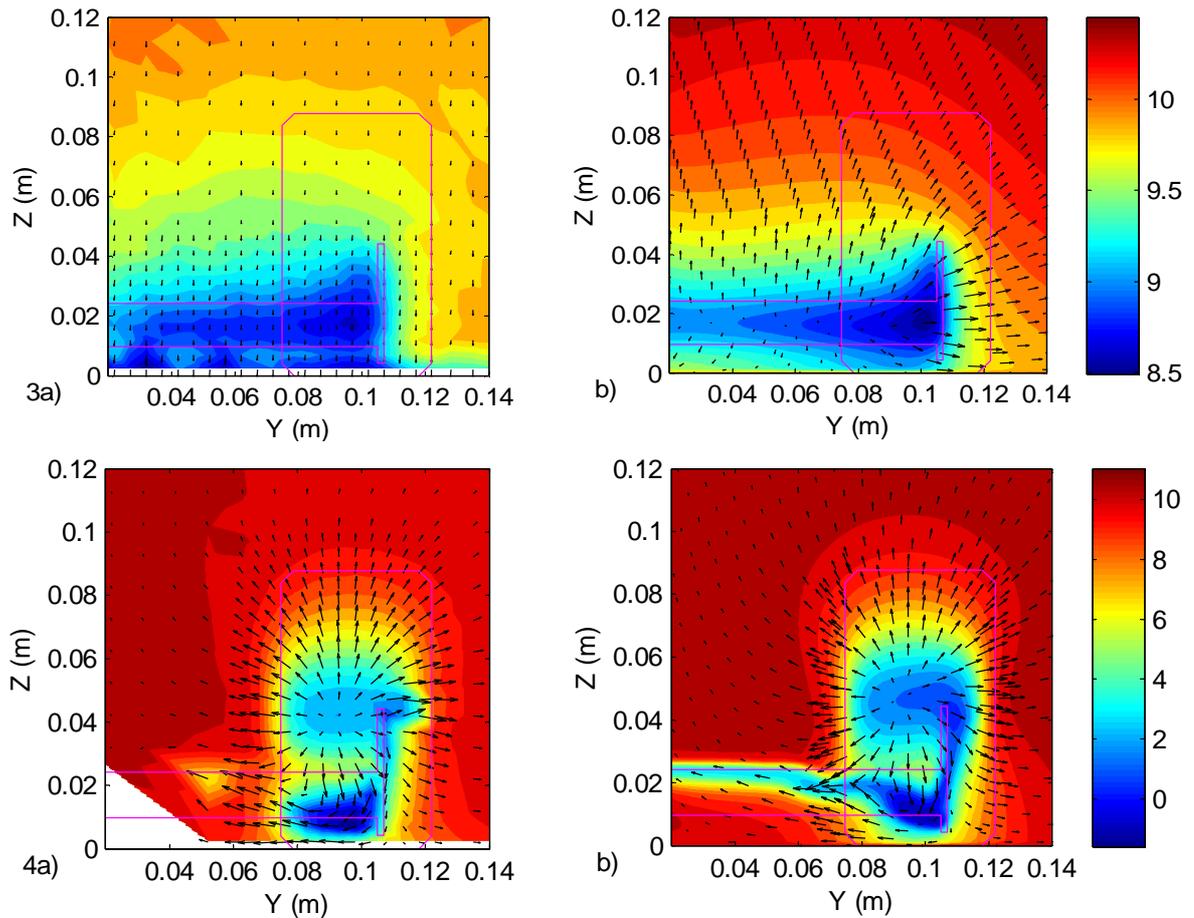
The wind tunnel is an open circuit, closed test section tunnel of cross-section 225mm by 340mm, with a free-stream turbulence intensity of 0.15%, a figure used in the numerical models. The moving ground is 990mm long and precautions have been



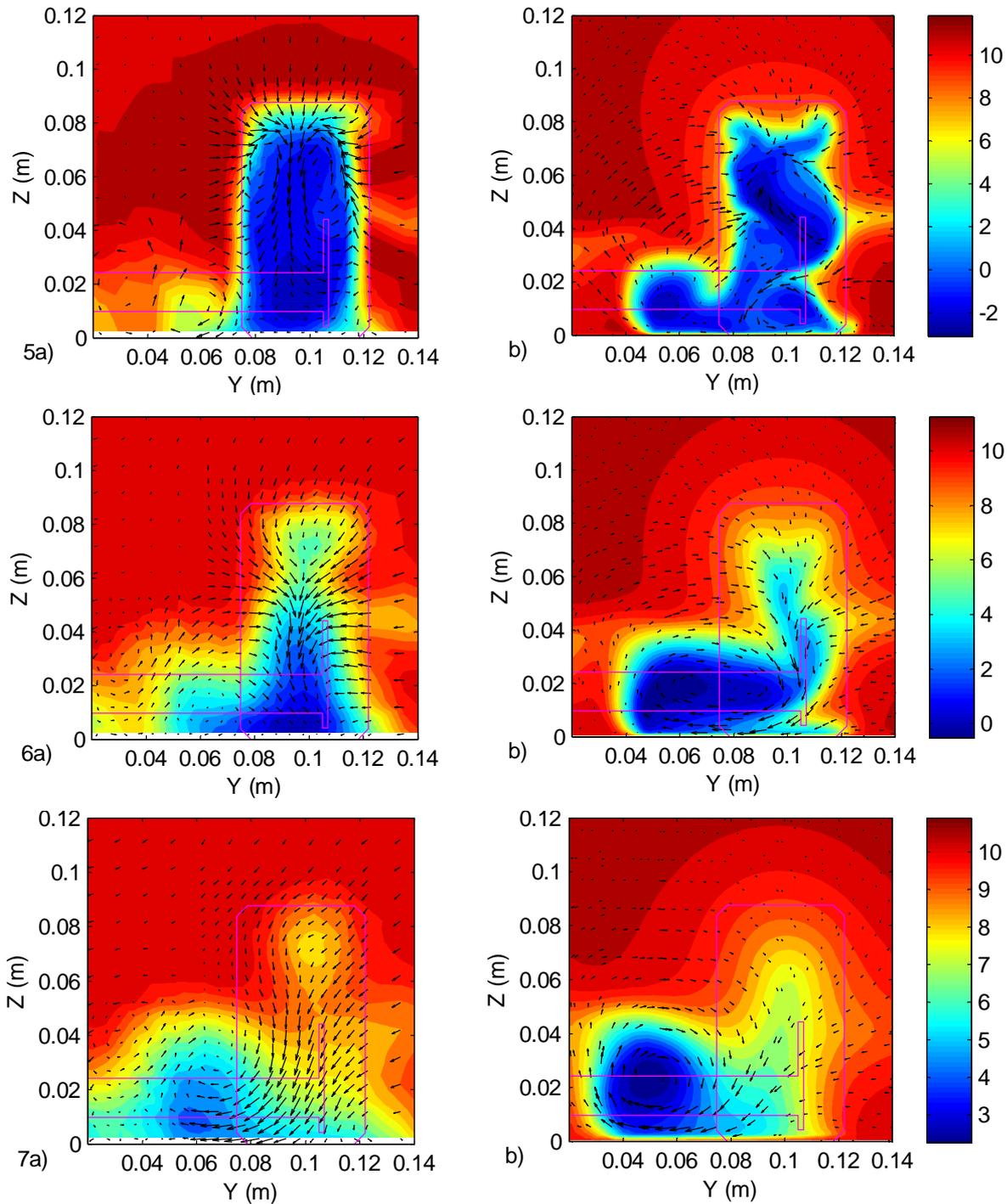
**Figure VI—2: Three-dimensional schematic of wing-wheel configuration in the moving-ground wind tunnel with LDA measurement planes.**

taken to prevent the development of a boundary layer over its length when no experimental model is present. Computed simulations treat static boundaries as smooth walls, with moving boundaries for the wheel and moving ground surfaces, and a constant velocity inlet and outflow condition at the far ends of the tunnel. A second order upwind differencing scheme was adopted to solve the steady-state Reynolds Averaged Navier-Stokes (RANS) equations over this domain. The free-stream velocity was set at 10m/s for all results shown in this work. This, when used with the chord length of 75mm, gives rise to a Reynolds number of  $5.11 \times 10^4$ .

Experimental results have been recorded by Diasinos at five planes perpendicular to the free stream velocity to track the evolution of structures in the flow. The planes were located at 0.63 and 2 chord-lengths forward of the axle, and at 0.75, 1.5 and 3 chord-lengths aft of the axle. A schematic diagram of the wing and wheel showing the locations of the five planes at which the LDA was used is shown in Figure VI—2 and results shall be referred to in the same order as they are denoted in this diagram. The LDA records the three spatial components of velocity simultaneously over a statistically significant interval of time, and given a timeseries of data, the Dantec™ proprietary Burst Spectrum Analyzer estimates average velocity, and the root mean square turbulent components of velocity. In Figures VI—3(a) to VI—7(a), contour plots generated from a triangulation on the nodal experimental  $x$ -velocity (streamwise velocity) are presented for each slice, overlaid with a vector plot indicating the direction of the swirling  $y$ - $z$  velocity components.



**Figures VI—3 and VI—4: Raw contours of  $x$ -velocity (m/s) on slices 1 and 2 from; (a) experimental LDA survey and (b) numerical  $k$ - $\omega$  results.**



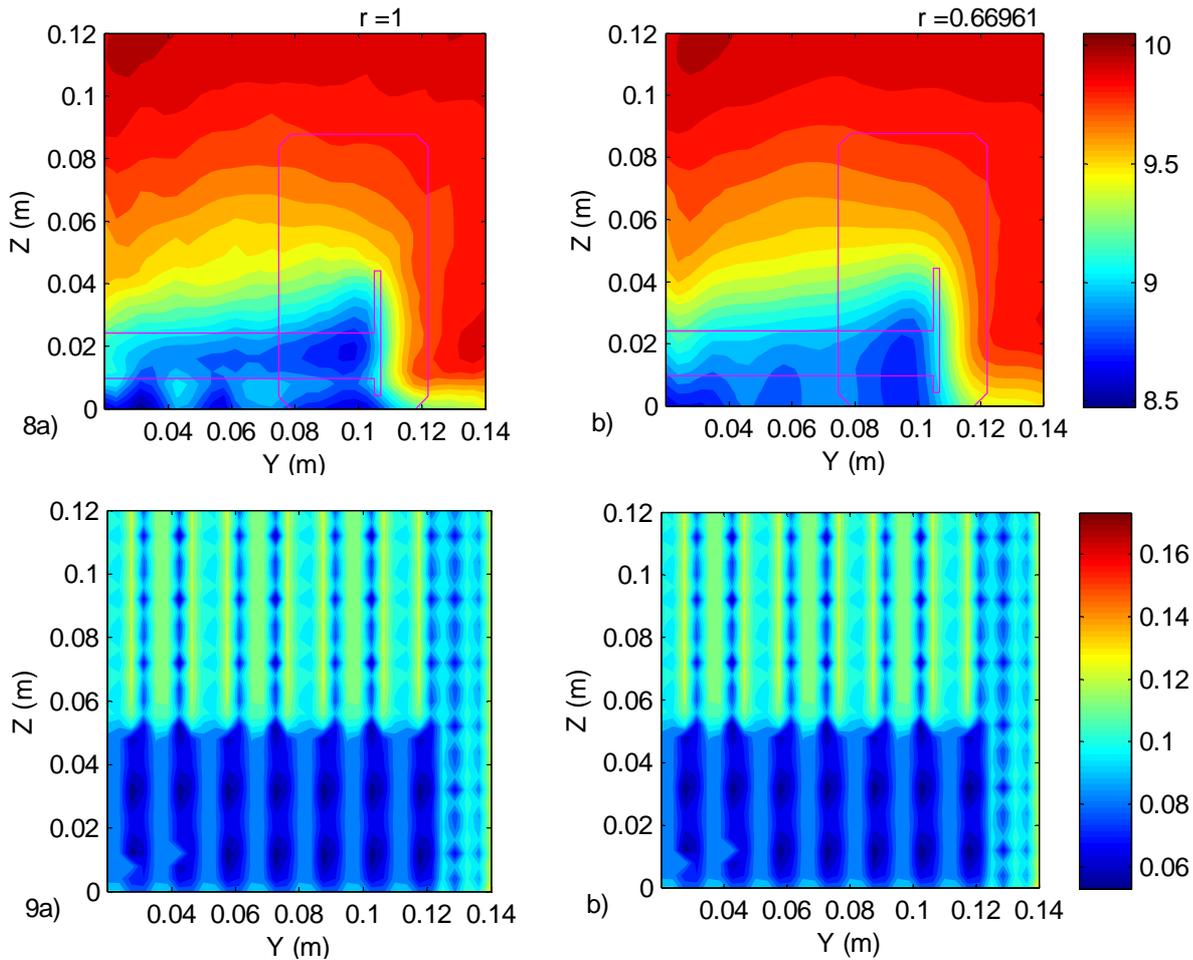
**Figures VI—5 to VI—7: Raw contours of  $x$ -velocity (m/s) on slices 3 to 5 from; (a) experimental LDA survey and (b) numerical  $k$ - $\omega$  results.**

Diasinos has also produced four numerical simulations of the experimental rig and wind tunnel that each use different turbulence models. The numerical models approximate a solution to the three-dimensional steady state Reynolds Averaged Navier-Stokes equations, using four different turbulence models to close the equations; the Spalart-Almaras model, the realisable  $k$ - $\varepsilon$  model, the  $k$ - $\omega$  model and

the  $k-\varepsilon$  model of renormalisation groups ( $k-\varepsilon$  RNG). For full details of the parameters used in these models, the reader is again referred to Diasinos [115, 116]. For now it suffices to observe that convergence work has been conducted on the numerical models: functional convergence of global lift and drag has been attained, and the solution is deemed insensitive to further iterative refinement. Diasinos came to the conclusion that the  $k-\omega$  model seemed to capture the basic physics of the problem best, so the raw contour plots of  $x$ -velocity calculated by this model are presented alongside the LDA results in Figures VI—3(b) to VI—7(b). Vector plots of the swirling  $y$ - $z$  components accompany, but note that vectors are not drawn from each computational node, as there are too many. The other computed results are presented later in Figures VI—24 to VI—28 for easier assessment of parity (with reference to a metric), and comparison with the cokriged plots.

The reasoning behind the selection of the  $k-\omega$  model centered around the rough replication of flow features, in particular the largest trailing vortices generated from the endplate and wheel. However, in general one notes that at least behind the wheel, the replication of the experimental flow-field by the numerical model is not that good. The introduction of the bluff body, the wheel, means that in these areas the flow is highly turbulent – with much of the turbulent energy contained in large *transient* fluctuations and structures. This complex behaviour is simply not captured by classical turbulence models as all such models are effectively steady-state approximations. In spite of this, the computational impossibility of performing a Direct Numerical Simulation or Large Eddy Simulation of this relatively complex geometry necessitates their use. Whilst it is clear that the  $k-\omega$  model does not capture all of the flow behaviour very accurately, it does a better job than the other turbulence models and nicely, it also exhibits similar variation of mean velocities. By contrast, all of the other models tend to simulate spatial variation that is actually of a smaller scale than that which was observed experimentally (as shall be seen in Figures VI—24 to VI—28).

The above are all good qualitative reasons to accept the  $k-\omega$  model over the others, but the present aim is to provide a quantitative indication of spatial agreement. Such an indication may only ever serve as an aid to validation, but it is

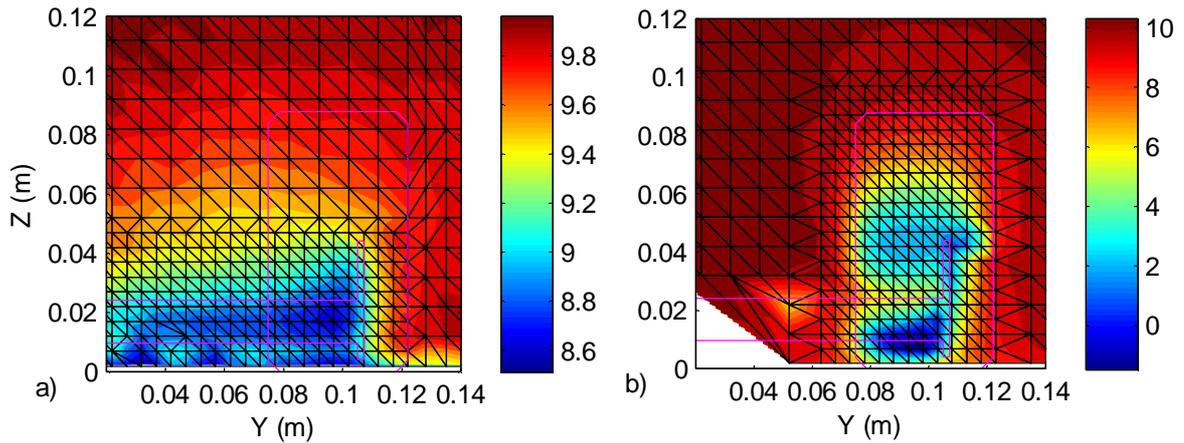


**Figures VI—8: Plot of kriged LDA  $x$ -velocity (m/s) contours on slice one; and VI—9: associated kriging standard deviation (m/s) where nugget effect is set at (a) zero, and (b) one third of total variance.**

still useful as a tool for making what is a difficult and largely qualitative decision often faced in engineering practice.

## VI - 2. Kriging and Smoothing

The kriging estimator may be introduced as a simple data interpolator and spatial smoother. It distinguishes itself from other interpolators and smoothing procedures by basing its estimations on an underlying spatial model, which is informed by spatial statistics generated on the data. A basic explanation of the immediate advantages of this approach follows, with regard to the simple generation of contour plots using univariate kriging.

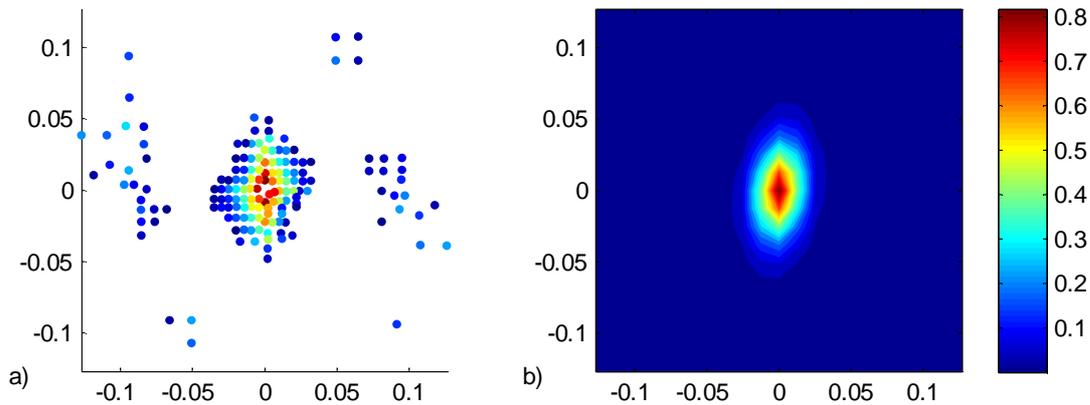


**Figure VI—10: The triangulation scheme behind nodal data contouring on (a) slice one and (b) slice two.**

### VI - 2.1. Univariate estimation

In Figures VI—3 to VI—7, variation between nodes is linear and dependent on an underlying triangulation scheme constructed between the nodes. This scheme is revealed for the first and second slices in Figure VI—10. Note that for first slice in particular, two nodes (Figure VI—10(a)) have been removed as outliers around ( $Y = 0.02$ ,  $Z = 0.01$ ), which has distorted the ‘structured’ triangulation around these removed points. Therefore, the use of a triangulation such as this may well introduced features that may not really be present – for the second slice an area behind the wing that is inaccessible to the LDA probe has not been surveyed, yet half of this area is contoured by badly stretched triangles between nodes at the boundary. Although in each case the adoption of a Delaunay triangulation would remove the distortions as best as is possible, such a scheme is still an arbitrary choice and its quality depends on the nodal data that it interpolates. Whilst the estimation is obviously reasonable at the nodes, it is unphysical away from them, comprising unrealistically smooth and homogeneous linear patches bordered by  $C^1$  discontinuities.

To demonstrate the basic capabilities of the kriging covariance model in estimation involving a single dataset, the contours in Figures VI—8(a) and VI—8(b) are generated from rasters (regular grids) of points estimated by kriging interpolation of the first slice of experimental results. In Figure VI—8(a) the covariance model that is fitted to the sample covariance statistics is forced to have a zero nugget effect,



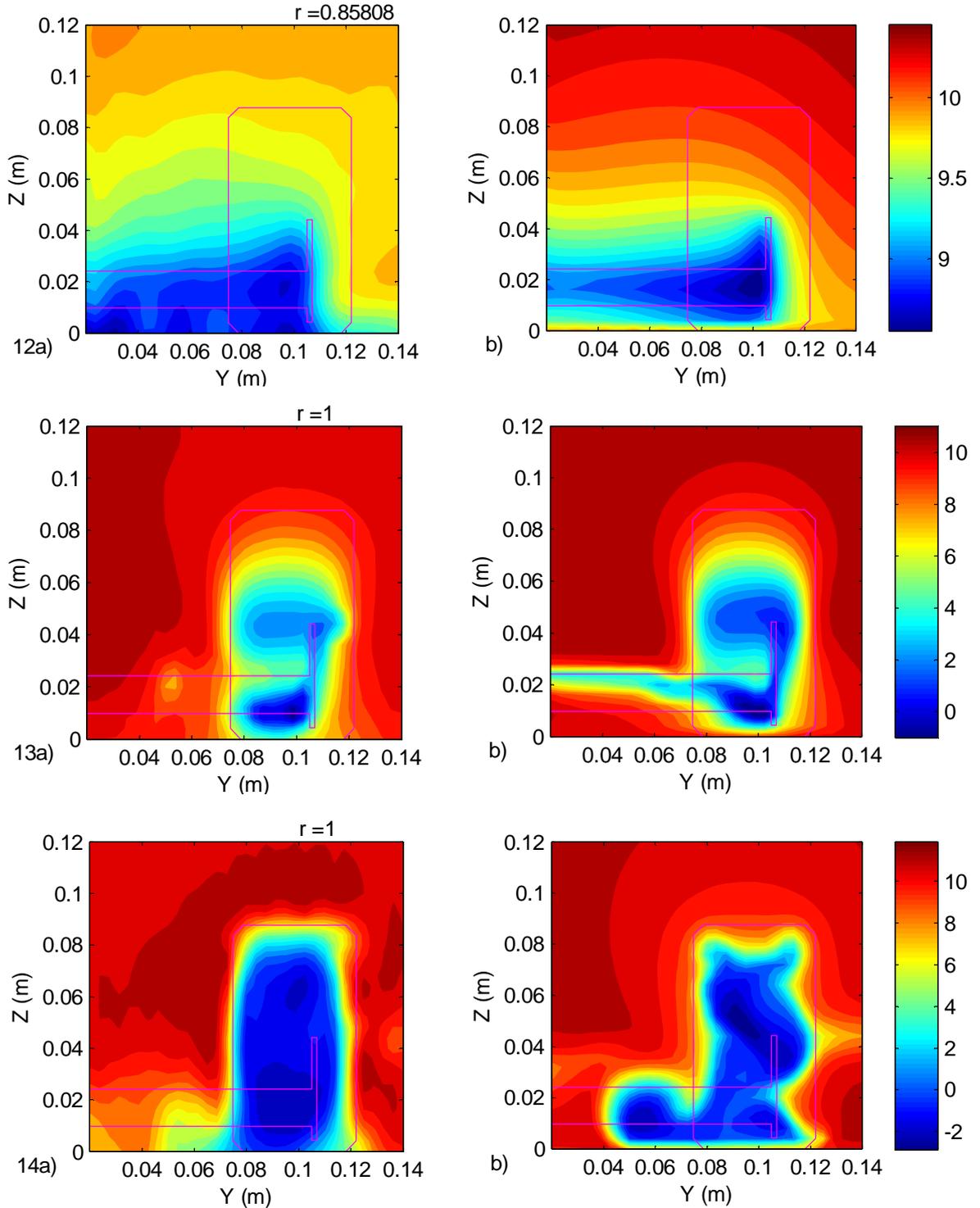
**Figure VI—11(a) Spatial statistics, and (b) corresponding normalised covariance function for  $x$ -velocity on slice 1 – nugget ‘floats’.**

whereas in Figure VI—8(b) the nugget effect is set to be one third of the total variance. Thus, in both cases the covariance model has been fitted using the second option in Section IV - 5.3, for which the two-sided constraint in Equation (4.50) is used. As was demonstrated in Chapter IV for a one dimensional time-series, the presence of the nugget effect has the effect of smoothing the now *spatial* data. The response surface is not constrained to pass through each data-point continuously and represents actually a moving average to some extent. Figure VI—8(a) more closely resembles the triangulated interpolation in Figure VI—10(a) in that it is not forcibly smoothed, but *unlike* Figure VI—10(a) (and Figure VI—3(a)), Figure VI—8(a) does not introduce arbitrary meshing artefacts.

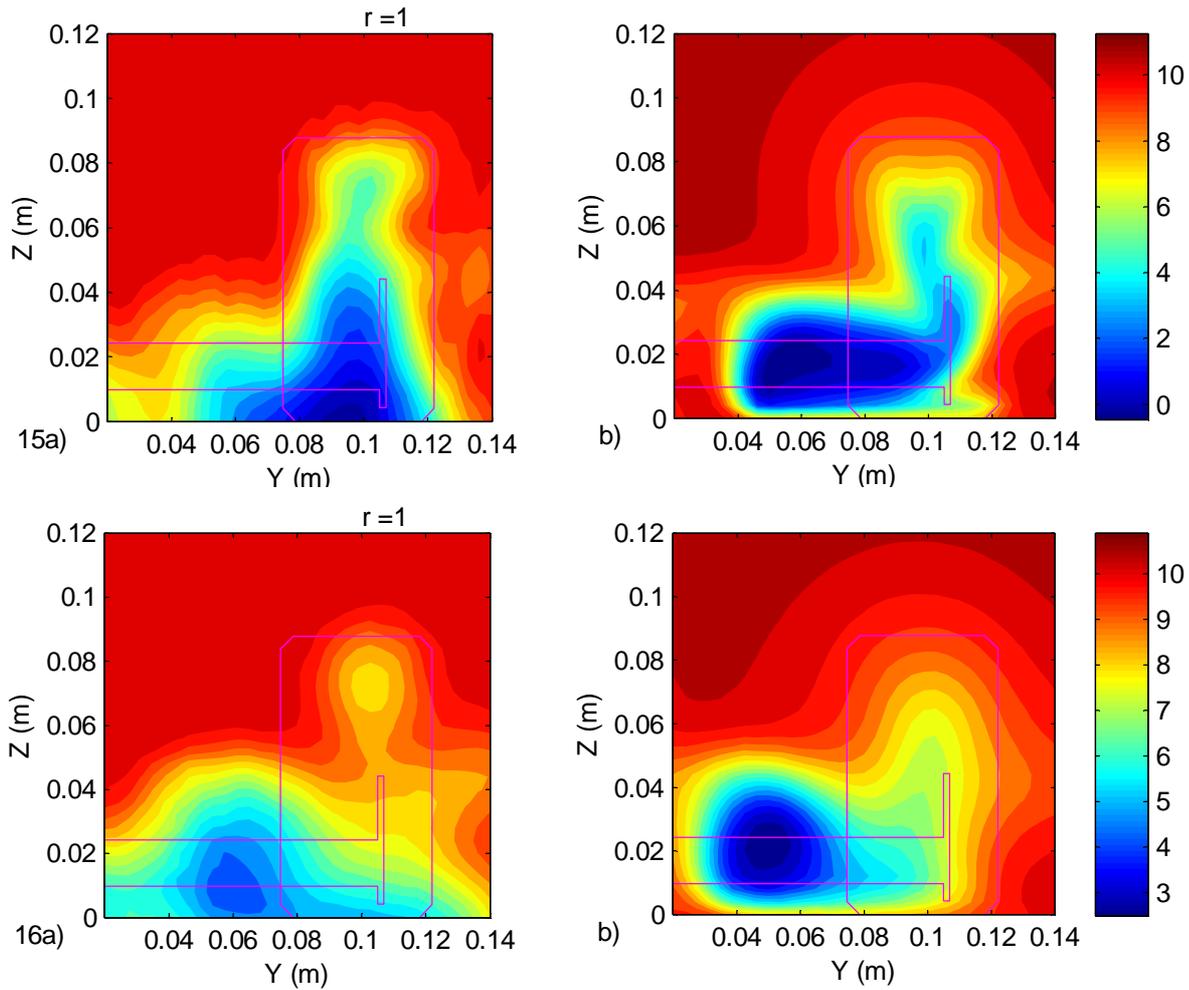
The second attraction of the kriged estimates, is that they also offer an estimation variance as expressed by Equation (4.58). This quantity is plotted in Figure VI—9 underneath the kriged rasters, over the same domains as Figures VI—8(a,b). This value provides an indication of the confidence one may have in the estimates locally, provided that the global covariance model is reasonable. Importantly although it is a local quantity, the kriging variance reflects a global covariance model. This means that plots of kriging variance in Figure VI—9 strongly reflect the spatial disposition of the nodes, and essentially provide a measure of the density of nodal information at a given location. The darker blue patch at the bottom of each figure is due to the extra density of the survey points in these areas, and the otherwise speckled nature of the plots is due to the regular coarse grid on which the point velocities were measured. Because the covariance model differs only in the

nugget effect, Figures VI—9(a) and VI—9(b) are almost exactly the same, and only differ as the data-points are approached. For the zero-nugget effect case, the variance must approach zero as one approaches each node.

The plots in Figures VI—8 and VI—9 have been made using zero-th order



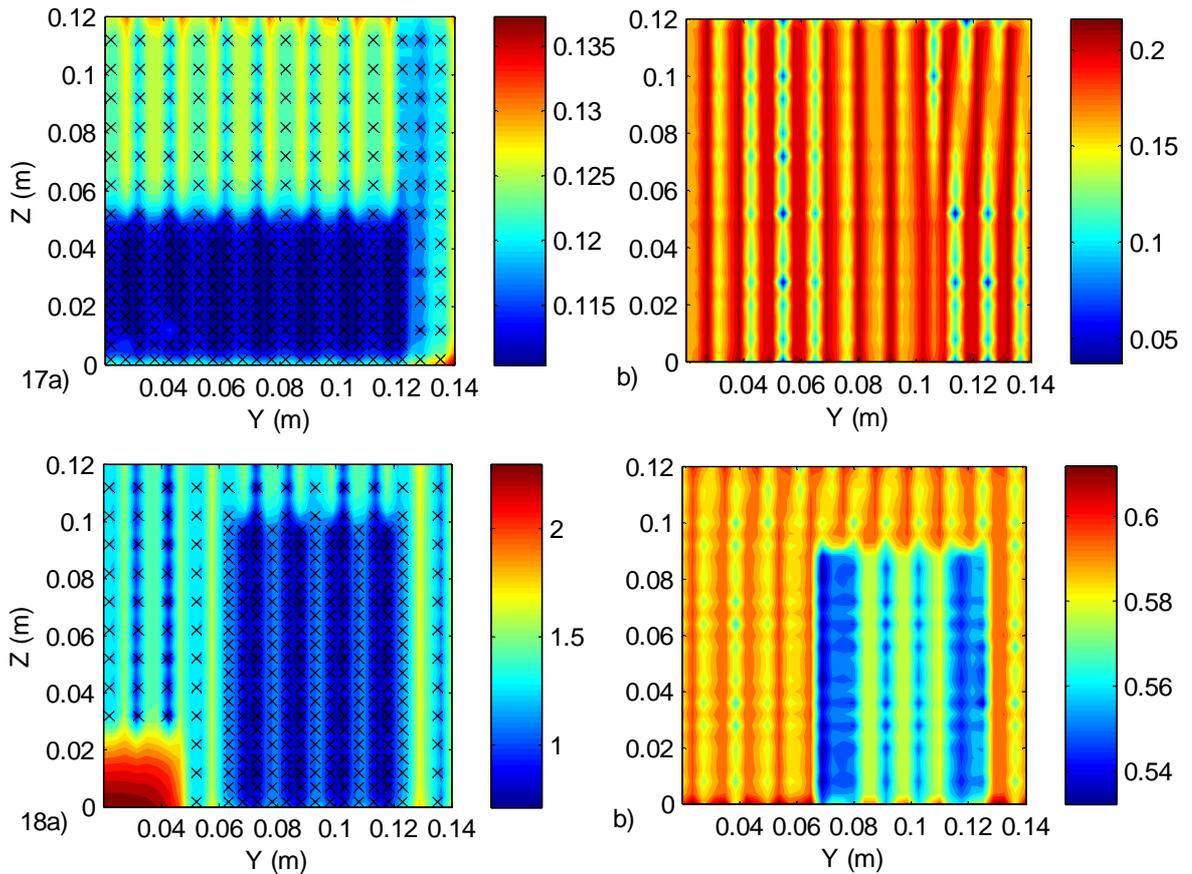
**Figures VI—12 to VI—14: Kriged contours of  $x$ -velocity (m/s) on slices one, two and three for; (a) experimental LDA results (b) CFD  $k$ - $\omega$  model.**



**Figures VI—15 and VI—16: Kriged contours of  $x$ -velocity (m/s) on slices four and five for; (a) experimental LDA results (b) CFD  $k$ - $\omega$  model.**

drifts and the supporting auto-covariant structure identification is by means of the variogram (Equations (4.7) and (4.17)), after preliminary linear detrending. These are all modelling options or decisions, however the estimates tend to be fairly robust against them – the primary way in which they are cosmetically affected is through the modelling of the nugget effect. To set this at a given value for the interpolations in Figure VI—8 is a significant modelling decision, which has been formed by preconceptions concerning what behaviour ought to be expected. However, it is possible to make this decision more scientifically by adjusting the nugget effect to fit the spatial statistics optimally, instead.

The auto-covariant statistics relating to the nodal data on the first slice are shown in Figure VI—11(a) with a corresponding covariance model in Figure VI—11(b). When fitting this model, the nugget effect was left to ‘float’ by using the one-sided constraint in Equation (4.49): its value is determined by a best fit to the



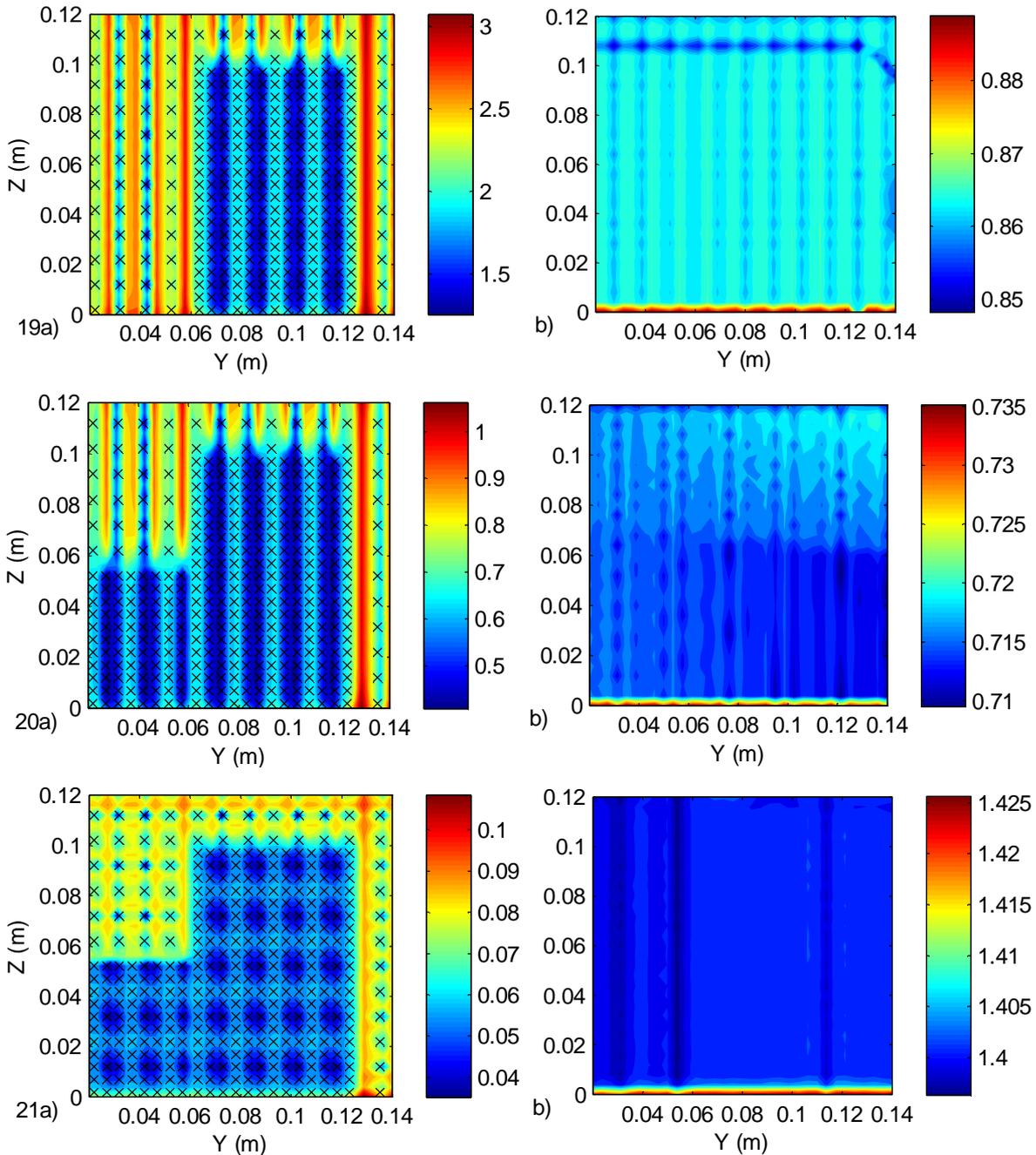
**Figures VI—17 and VI—18: Kriging standard deviation (m/s) on slices one and two for; (a) LDA estimates and (b)  $k-\omega$  model (corresponding to Figures VI—12 and VI—13)**

statistics. The interpolates on the first slice in Figure VI—12(a), are generated by this covariance model. Note that this plot is smoother than Figure VI—8(a) – where zero nugget effect was set to zero, but it is not as cosmetically oversmoothed as Figure VI—8(b). Given well behaved statistics and a reasonable fit to them, the determination of the nugget effect is no longer an arbitrary or cosmetic smoothing decision, but one based on what can be directly observed in the data. Note that if the nugget effect is equal to the total variance, the estimate will only be the average of the data in the domain of estimation – a simple moving average.

Further kriged plots of  $x$ -velocity have been produced for both the experimental and the  $k-\omega$  model results on the rest of the slices in Figures VI—12 through VI—16, for which the nugget effect was again left to float. The corresponding kriging variance is plotted in Figures VI—17 to VI—21, overlaid with the nodal locations of the measured LDA data. It is sensible to introduce a signal-to-noise ratio, which is defined as:

$$r = 1 - \frac{s^{nugget}}{2} \quad (6.1)$$

This ratio is included with the kriged contours in Figures VI—12 to VI—16. In all but the first slice, it is unity. This indicates that at least for the resolution of the relatively crude covariance models proposed, there is little spatial ‘noise’ with respect to the greater structure of the spatial variation, for most of the LDA results.



**Figures VI—19 to VI—21: Kriging standard deviation (m/s) on slices three to four for; (a) LDA estimates and (b)  $k$ - $\omega$  model (corresponding to Figures VI—14 through VI—16)**

This is actually broadly in line with the experimental observations: confidence intervals have been supplied (using the method of Benedict and Gould [117]) that indicate that the statistical uncertainty in each data-point is several orders of magnitude less than the ensemble variation of the nodal data. Thus a very significant nugget effect was not expected anyway.

However, the nugget effect for slice one *is* significant. This is probably a by-product of the more uniform flow on this slice, which lowers the total variance of the data here and makes the nugget effect relatively more significant. The accurate estimation of the nugget effect is a difficult problem which is hampered by the relative coarseness of the LDA nodal data. Coarser nodal data implies a greater minimum distance between vector lags on the sample covariance functions and therefore less densely packed spatial statistics around the origin of the correlation function. This means that there must necessarily be more guesswork about what is going on there, and consequently the optimisation problem presented in Section IV - 5.1 may suffer from indeterminacy.

In any case, it is entirely likely that the confidence intervals collected experimentally are *not* a reasonable estimate of unstructured variance. These statistics may underestimate the true unstructured variance simply because they preclude other unverifiable, systemic sources of variance. For example, unmeasured environmental changes may cause small changes in operating point as the LDA measurements are made, and this effectively adds point-to-point variability. The errors arising from such uncontrollable or unmeasured factors will vary from node to node, adding short-scale variance which is imperceptible to the variance calculation performed at each node. The same might also result from errors in probe alignment at each measurement station. By appealing to the raw nodal results, kriging makes more sensible estimates of signal-to-noise ratios and is thus more robust against unknown or unquantifiable sources of variance – which are necessarily the most awkward. However, should credible estimates of point variance be available, then it is entirely reasonable to set the nugget effect using these estimates, in the manner of Figures VI—8 and VI—9.

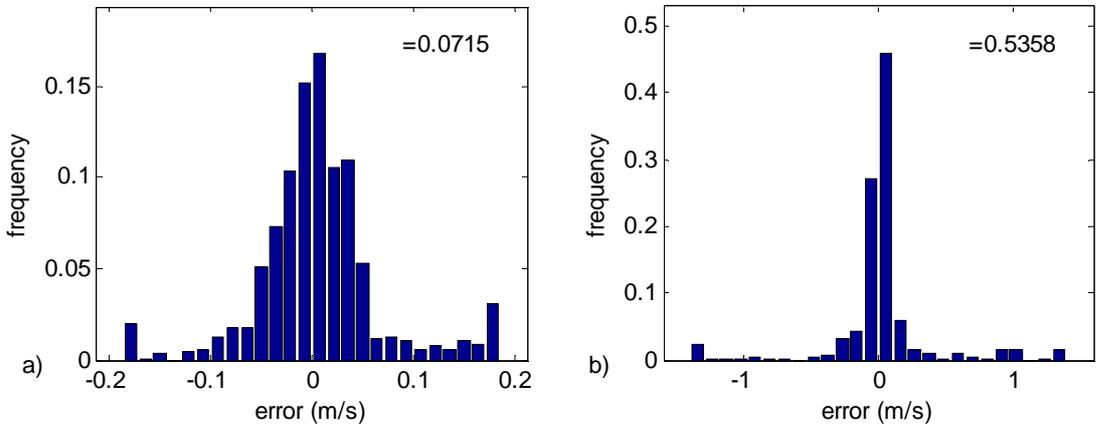
Comparing Figures VI—3 to VI—7 and Figures VI—12 to VI—16, apart from some smoother behaviour in the LDA results on slice one there is very little

difference between the kriged results and the conventional contours. As expected, this is especially the case for the kriged CFD results. Kriging affects estimates at the scale of the grid, and because the CFD results have been produced on a fine mesh, the contours are exactly similar to the results presented in Figures VI—3(b) through VI—7(b).

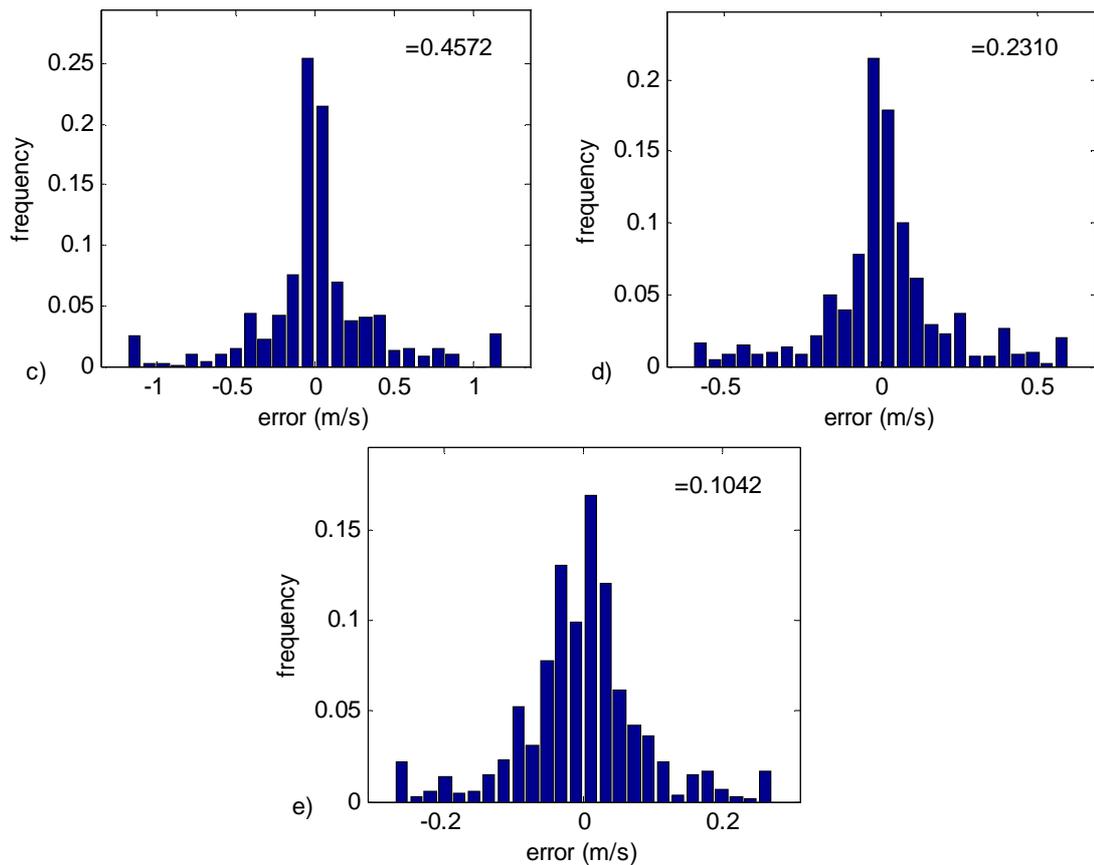
**VI - 2.2. Cross-validation**

It is often useful to check on the performance of spatial estimation procedures by means of a so-called *cross-validation* exercise. This involves producing an estimate at each of the data node locations after first removing the node, and comparing or cross-validating this value with the known (removed) value. For each estimate so produced, all the other nodes remain in place and the deviation of these estimates from their known values gives some indication of the performance of the algorithm.

The frequency histograms in Figure VI—22 are of cross-validation errors: the differences between the estimated value and the actual removed value that it estimates. These histograms have been prepared by randomly selecting points uniformly throughout the domain of estimation on each slice. Cross-validation was then performed at the node nearest to each randomly selected point, for the streamwise velocity. This randomisation was aimed at reducing the mal-effects of variable nodal data density – if the data is preferentially distributed in a particular



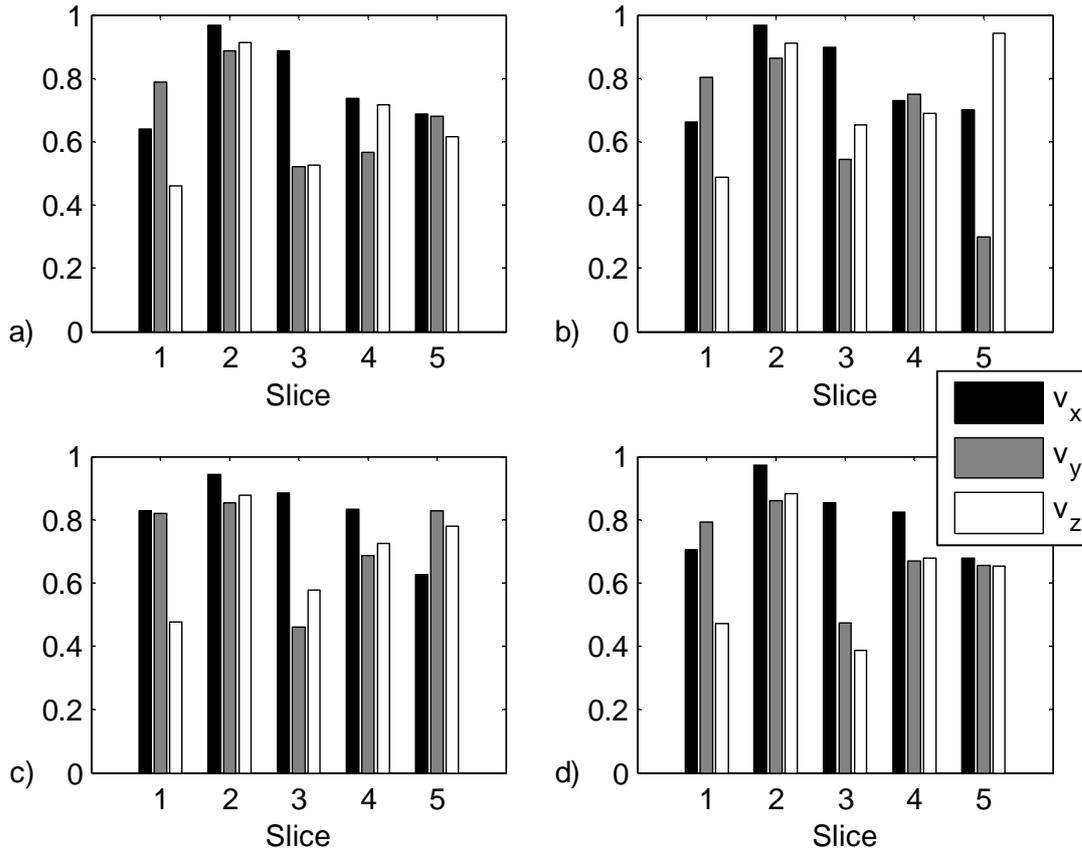
**Figure VI—22: Frequency histogram of LDA x-velocity (m/s) cross-validation errors on; (a) slice one, and (b) slice two.**



**Figure VI—22: Frequency histogram of LDA  $x$ -velocity (m/s) cross-validation errors on; (c) slice three, (d) slice four, and (e) slice five.**

area of the whole domain, the cross-validation deviations may be skewed by the local behaviour of the model there. A total of 13000 random points was chosen so as to ensure that there would be negligible statistical variation between possible histograms – this is two orders of ten more random selections than there are nodes. The plots are produced between  $\pm 2.5\sigma$ , where  $\sigma$  is the standard deviation calculated on the cross-validation errors, also presented in the plots.

What is generally seen in Figures VI—22(a-e) are slightly distorted but fairly symmetric bell-shaped plots centred around zero, with relatively heavy tails and sharp peaks. Zero-centred plots indicate that the estimation procedure is unbiased, and that the standard deviation presented in Figure VI—22 is a reasonable measure of dispersion. The limited number of spatial data generally prevent closer representations of the Gaussian distribution, which these plots should theoretically approach. Note that there are two other peaks at the extreme ends of these frequency histograms: these represent the frequency of the values in the first and last bins which consist of all those values smaller and greater than the cutoffs for the



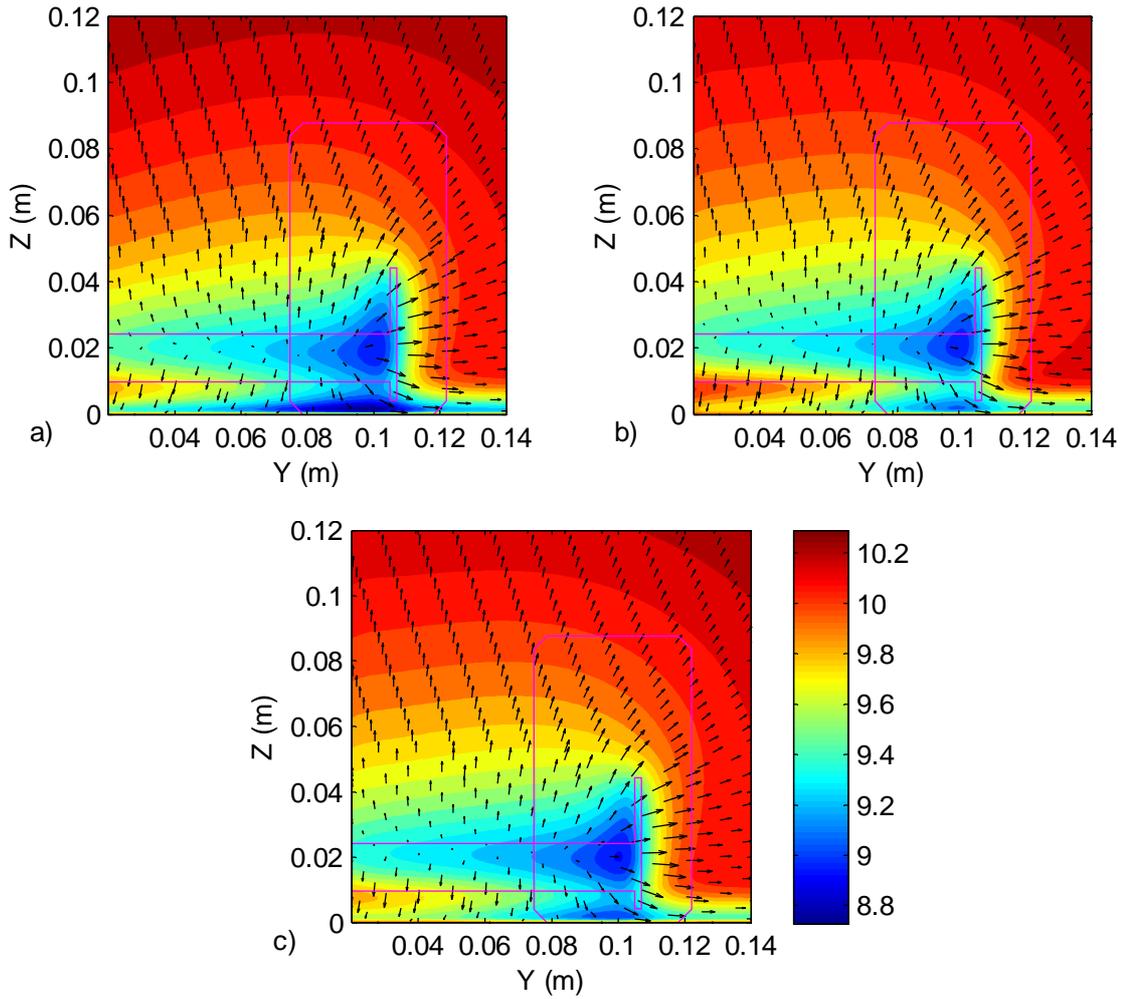
**Figure VI—23: Kriging correlation coefficient ( $\eta$ ) at each slice for each turbulence model; (a) Spalart-Almaras (b) realisable  $k-\varepsilon$  (c) RNG  $k-\varepsilon$  (d)  $k-\omega$  models.**

histogram. The heavy tails on these distributions are conceivably due to nodes in the domain which do not conform well to the stationary covariance model, or outliers.

### VI - 3. Spatial Similarity

As has been remarked, and briefly demonstrated in the appraisal of the relative merits of the  $k-\omega$  model in Section VI - 1, the comparison of disparate spatial datasets which otherwise attempt to model similar phenomena is an activity that is highly dependent on the intuition of the practitioner. In this section, a simple quantitative means of comparing spatial data based on cross-covariant structure identification is introduced and compared with more typical methods. This metric for spatial similarity is in no way intended to supplant a practitioner's intuition – but it is intended to aid the practitioner in making speedier and more reliable conclusions.

The comparison of spatial datasets is a central issue in this thesis. Quantification of spatial similarity proves to be a surprisingly difficult task – not



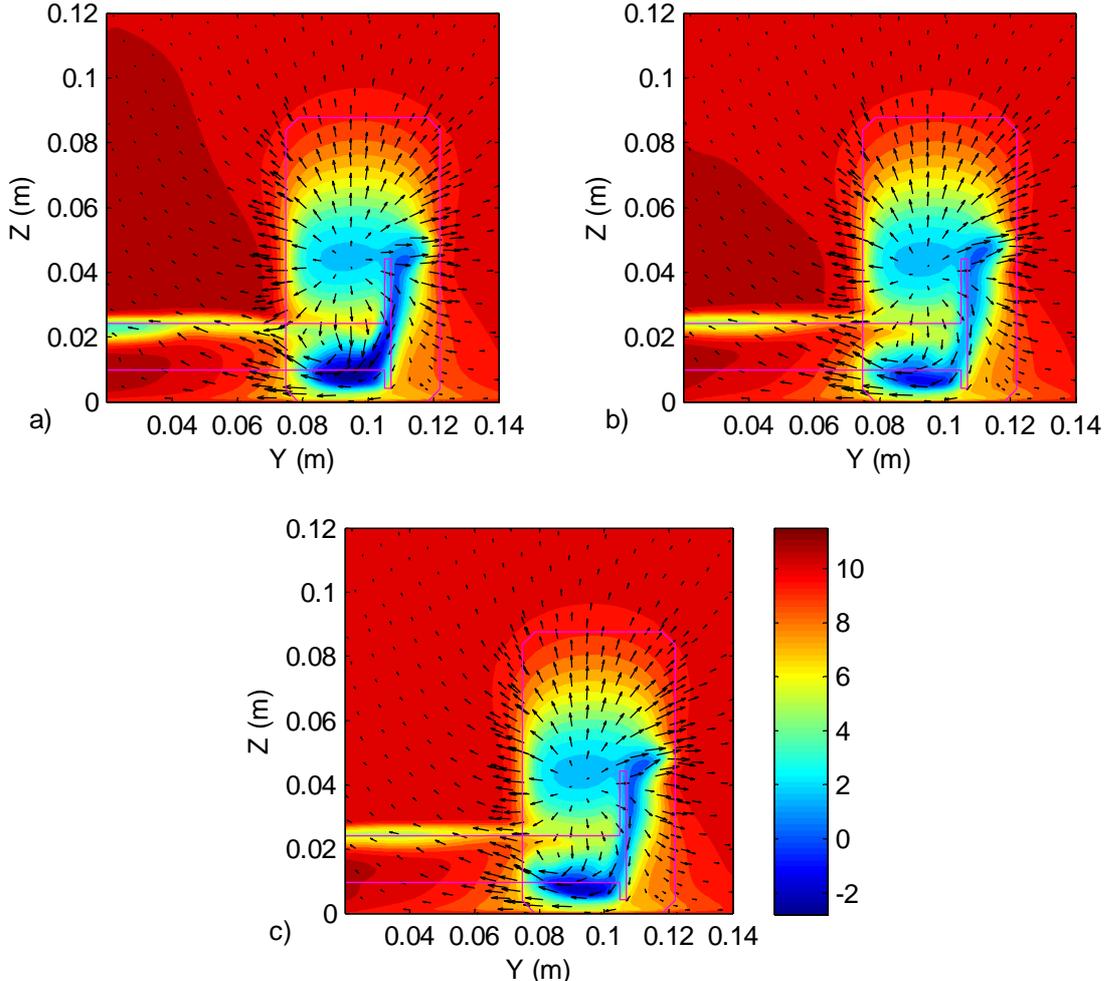
**Figure VI—24: Raw  $x$ -velocity (m/s) on slice one predicted by (a) Spalart-Almaras, (b)  $k$ - $\varepsilon$  realisable, (c)  $k$ - $\varepsilon$  RNG models.**

because a means of effecting it is unknown, but because it presents something of an open-ended question. There are many ways of measuring similarity, but many of them exhibit inconsistencies or are not entirely suitable for the present aims. Given that an assessment of the correlation between datasets is central to the development of the cokriging estimator, it is natural to use the cross-covariance models used in cokriging to answer this question. Where the coefficients  $s^i$  are the affine weights given to the coregionalised variogram models in Equation (3.69) for a given covariance model fit to a set of spatial statistics, the present author proposes that;

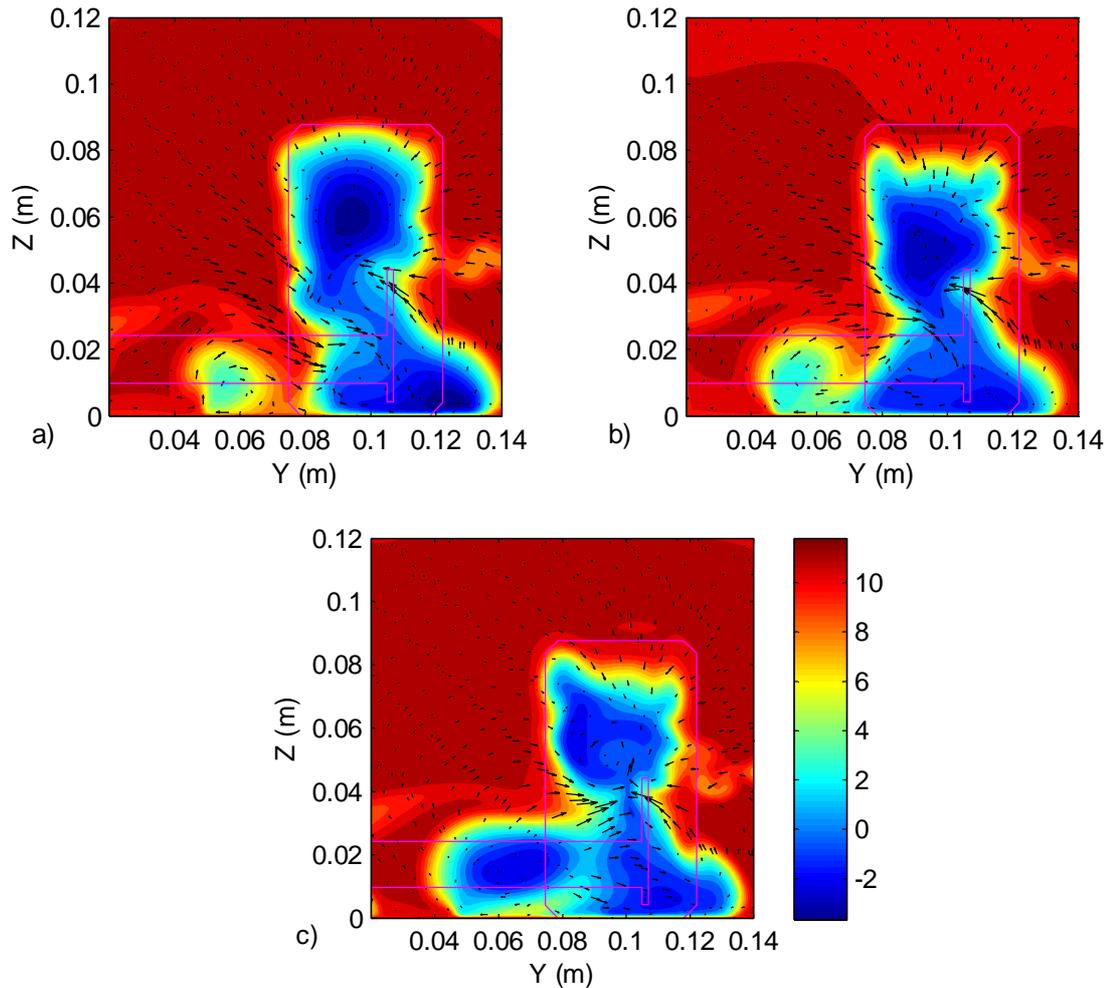
$$= \frac{\sum_{i=1} s^i}{\sqrt{\sum_{i=1} s^i \sum_{i=1} s^i}} \quad (6.2)$$

may be used as a metric for similarity between the datasets pertaining to the subscripts  $\alpha$  and  $\beta$ . Considering the relation in Equation (3.71), this is essentially an extension on a regular correlation coefficient or that offered by Matheron [118] (who is cited by Chilès and Delfiner [84]) for proportional covariance models. Note that like an ordinary correlation coefficient,  $\eta_{\beta}$  ranges between plus and minus one – although as positive correlation between the datasets may be expected, it is generally positive.

The coefficient  $\eta_{\alpha\beta}$  may be calculated in the structure identification steps arising in a cokriging involving the datasets  $\alpha$  and  $\beta$ . To simplify the inferences, the covariance model used in the subsequent discourse only extends to these two datasets: a bivariate structure identification is performed. The variables used in this structure identification consist of the LDA experimental dataset and each of the



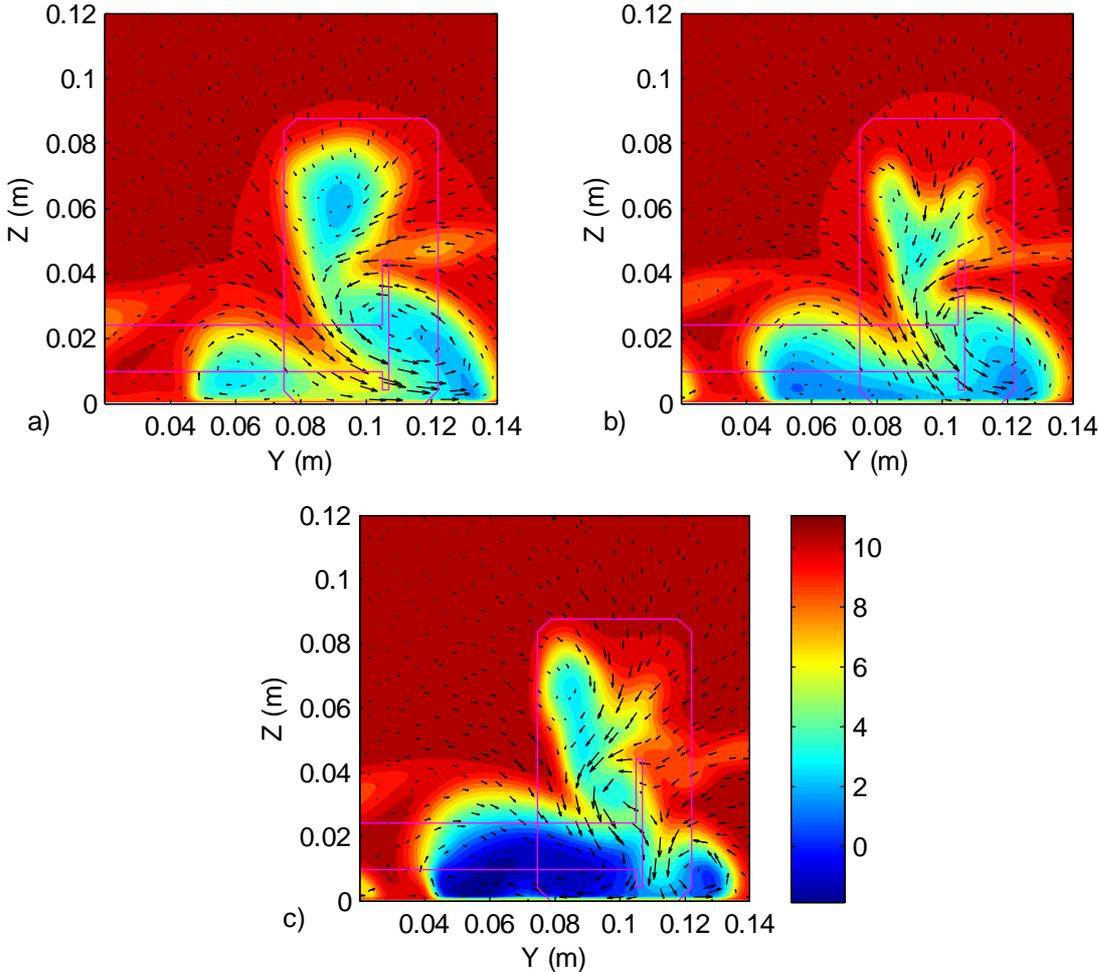
**Figure VI—25: Raw  $x$ -velocity (m/s) on slice two predicted by (a) Spalart-Almaras, (b)  $k$ - $\epsilon$  realisable, (c)  $k$ - $\epsilon$  RNG models.**



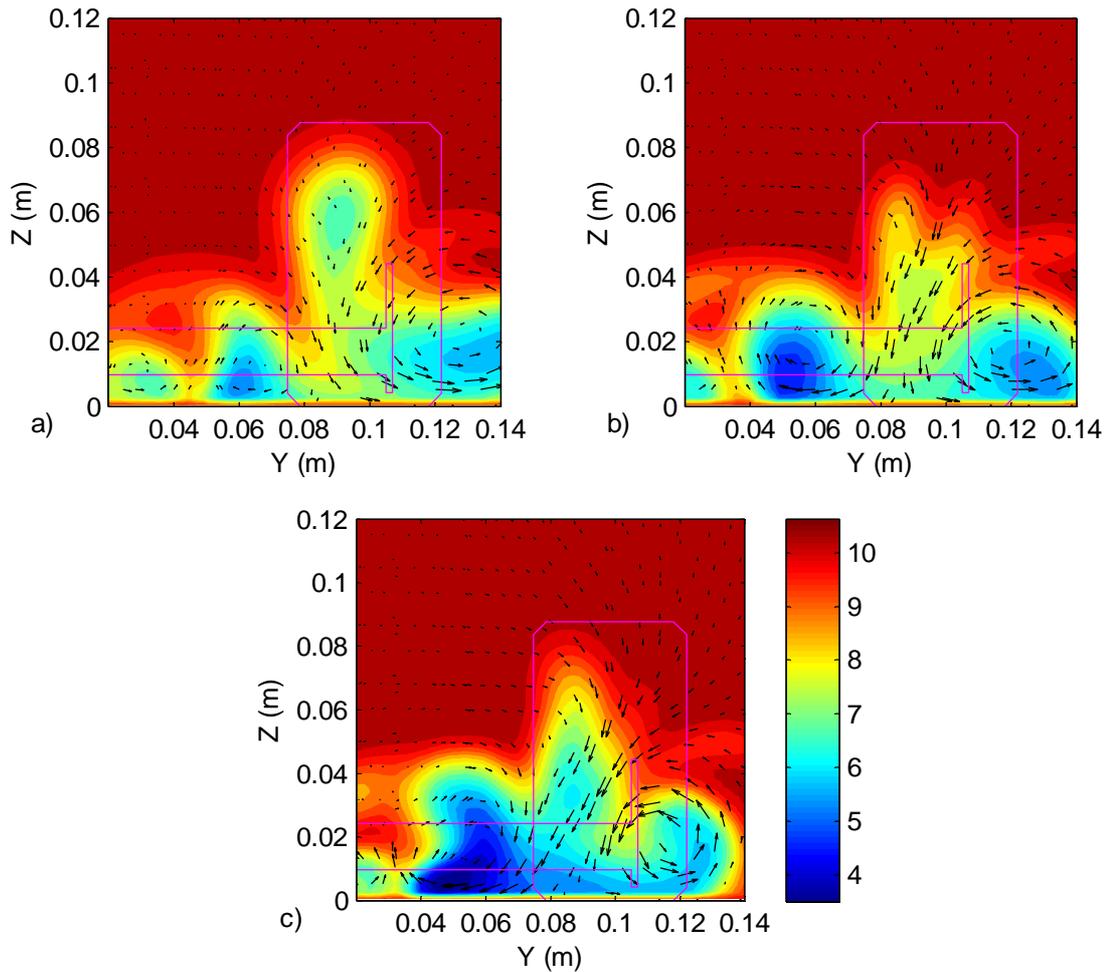
**Figure VI—26: Raw  $x$ -velocity (m/s) on slice three predicted by (a) Spalart-Almaras, (b)  $k$ - $\varepsilon$  realisable, (c)  $k$ - $\varepsilon$  RNG models.**

numerical datasets; comprising the results for the Spalart-Almaras model, the realisable  $k$ - $\varepsilon$  model, the  $k$ - $\omega$  model and the RNG  $k$ - $\varepsilon$  model. This structure identification is performed for all three velocity components – hence cross-covariance between the  $x$ -velocity fields obtained by LDA and CFD, the  $y$ -velocity fields obtained by LDA and CFD and the  $z$ -velocity fields obtained by LDA and CFD – for the four turbulence models. With or without then proceeding to an estimation step, the coefficient in Equation (6.2) can then be calculated from the coefficients  $s^i$  in the covariance model: these results are presented graphically in Figure VI—23. This figure has four parts, corresponding to the four turbulence models used, and in each part the correlation coefficients for the three velocity components are presented together, at each slice. The rest of the computed flow-fields are presented for reference in Figures VI—24 to VI—28, to complement the results for the  $k$ - $\omega$  model already presented in Figures VI—3 to VI—7.

The ‘kriging’ correlation  $\eta$  in Figure VI—23 relating to the  $x$ -velocity is quite similar in all models: there is a peak at the second slice, which subsequently falls off as the wake develops. This tapered reduction reflects the numerical results, wherein the numerical models only really start to differentiate themselves in the last two slices – the immediate effect of the bluff body in slice three is predominantly to develop a clear wake lee of the wheel, visible in Figure VI—5. Whilst this basic structure is captured, the more intricate behaviour of the trailing vortices downstream is relatively badly handled by the numerical models. Accordingly, for all models there is a sharp decrease in kriging correlation at the third slice for the  $y$  and  $z$ -velocity components, which generally improves thereafter. This suggests that the spurious structures in the streamwise velocities are evolving as a result of erroneous modelling of the swirling  $y$ - $z$  components in the immediate wake of the wheel. Overall, the  $k$ - $\omega$  model seems to achieve better and more consistent  $\eta$  correlation in



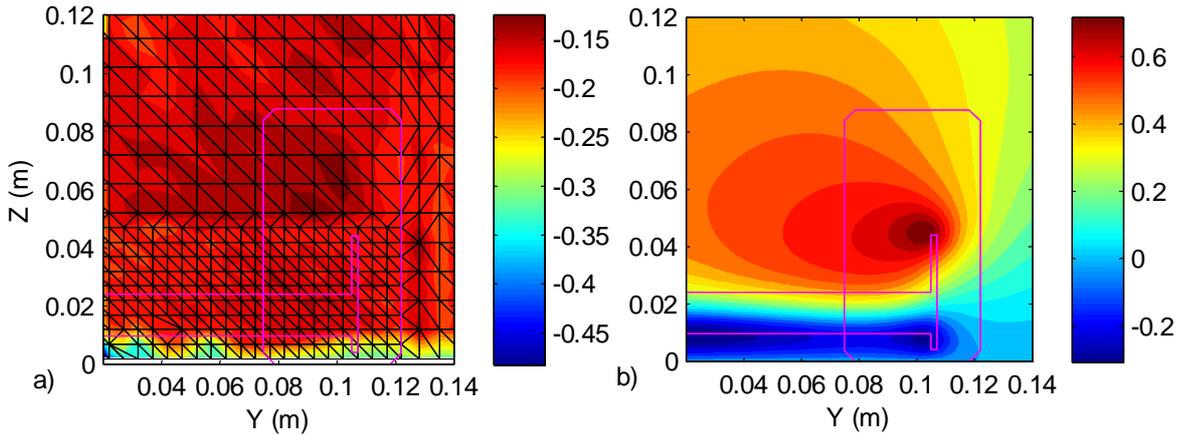
**Figure VI—27: Raw  $x$ -velocity (m/s) on slice four predicted by (a) Spalart-Almaras, (b)  $k$ - $\epsilon$  realisable, (c)  $k$ - $\epsilon$  RNG models.**



**Figure VI—28: Raw  $x$ -velocity (m/s) on slice five predicted by (a) Spalart-Almaras, (b)  $k$ - $\varepsilon$  realisable, (c)  $k$ - $\varepsilon$  RNG.**

Figure VI—23, especially as the wake evolves.

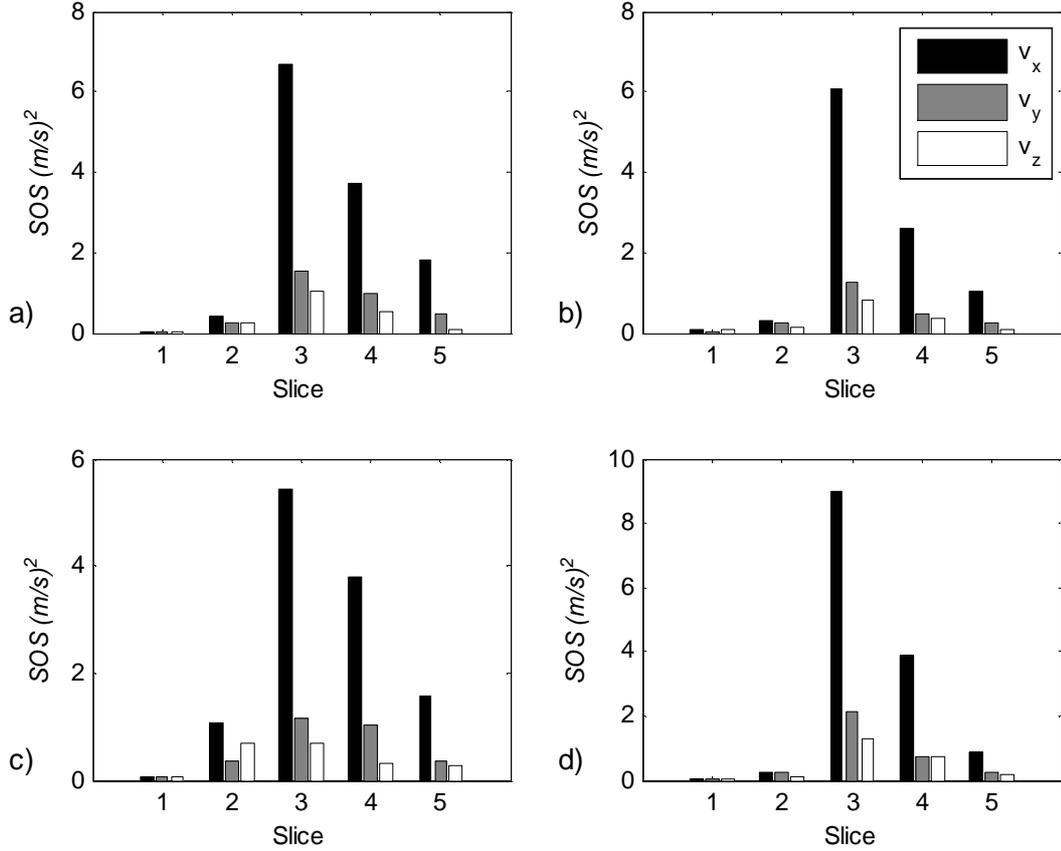
The kriging correlation seems to offer relatively erratic inferences for the swirling  $y$ - $z$  components lee of the wheel. This may be due to the strong interaction of the  $y$  and  $z$  components of velocity, which have not been accounted for in the cross-covariance model used to determine the kriging coefficient  $\eta$ . This model compares each component of velocity using independent bivariate structure identifications, thus interactions between the components are simply not considered. This is not so problematic for the streamwise velocities which present the evolving wake, but as the swirling components must interact strongly the kriging coefficient does not describe the extent of agreement completely – except in the manner of comparing the spatial features of the numerical and experimental velocity components in isolation.



**Figure VI—29:  $z$ -velocity (m/s) at slice one; (a) raw LDA results, (b) raw numerical ( $k$ - $\omega$  model) results.**

The kriging correlation on slice one also presents some interesting anomalies, as on this slice there is less structured variation in comparison with the variation due to spatial noise (as was seen in Figure VI—12(a)). This is especially apparent for the swirling components, as illustrated in Figure VI—29. Indeed, because the structures are so dramatically different, it is doubtful whether  $\eta$  is really representative for these components on this slice, which experimental results seem to be suffering some downwash or probe misalignment. The coregionalisation fitting will need to reconcile both the short-range structures apparent in Figure VI—29(a) and the long-range structures in Figure VI—29(b) – which it simply cannot do. The result is a bad fit to the spatial statistics and so kriging correlation  $\eta$  is unreliable for these components, on this slice. By contrast, the  $x$ -velocity on slice one achieves good kriging correlation with the  $k$ - $\omega$  model only: unlike the other turbulence models in Figure VI—24, this model does not predict an extended stagnation point from the wheel close to the ground.

It was noted that there are many ways of measuring spatial similarity – why then should the proposed metric possess advantages over any other? The vast majority of proposed validation techniques fundamentally compare data on a pointwise basis, which means that the correlation of the fields as a whole can be overlooked or obscured by the need to interpolate data to measured points. For the purposes of a validation study Michalek [61, 119] uses a metric for similarity resembling a variance calculation,

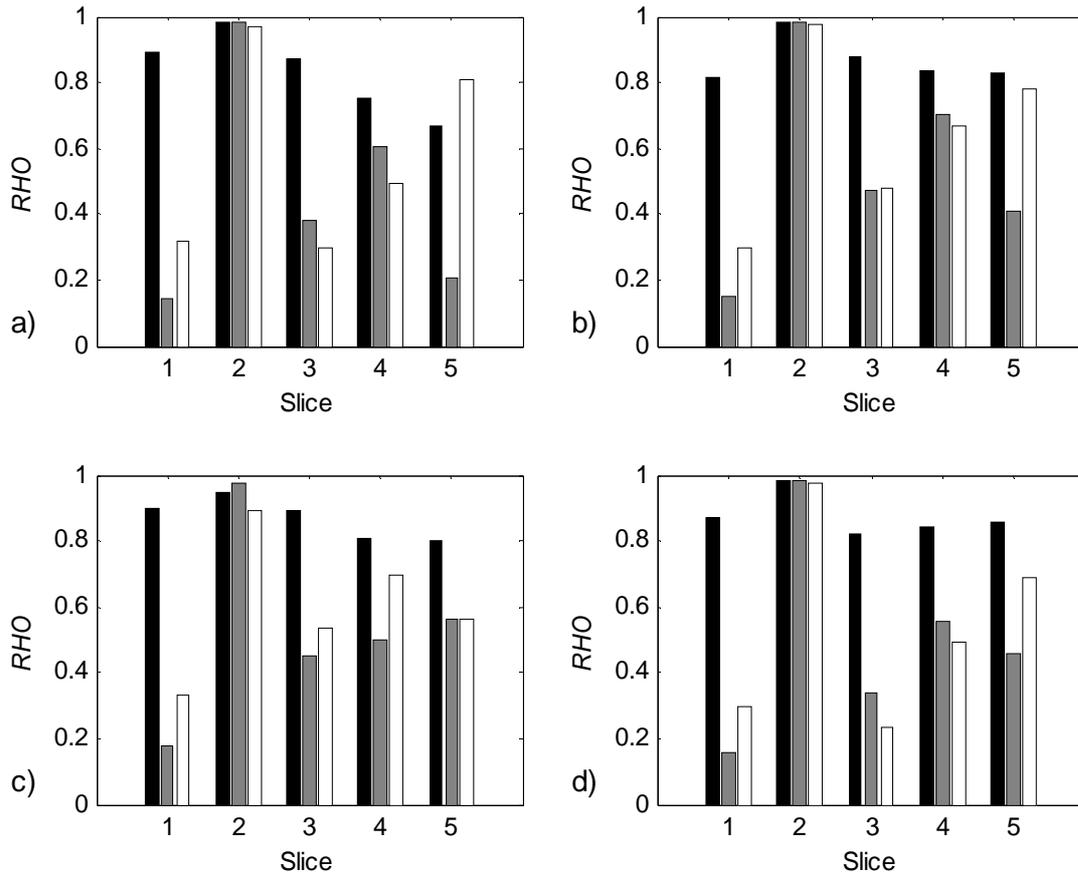


**Figure VI—30: Average sum-of-squares difference (m/s)<sup>2</sup> (SOS) at each slice for each turbulence model; (a) Spalart-Almaras (b) realisable  $k\text{-}\varepsilon$  (c) RNG  $k\text{-}\varepsilon$  (d)  $k\text{-}\omega$  models.**

$$SOS = \frac{1}{n} \sum_{i=1}^n (u_i^{LDA} - u_i^{CFD})^2 \quad (6.3)$$

in which the two datasets in question take comparable values at coincident nodes. It is noted here that this is merely adopted as a ‘common’ practice. Where data are not generated at the same locations in the physical and simulated domains, interpolation is used to reconcile the two sampling strategies, at the expense of some interpolation error [39]. Equation (6.3) was applied with a summation over the LDA measurement nodes, at which the computed results were interpolated (incidentally, using kriging) to provide coincident  $u^{CFD}$  nodal results. Because of the density of the computed results, it is expected that the error due to interpolation was relatively minor in this case. Results of the application of Equation (6.3) are summarised in Figure VI—30 for all turbulence models, measurement planes and velocity components.

Predictably, the squared difference articulated in Equation (6.3) is relatively small for the first two slices, but jumps after the introduction of the bluff body, reducing slightly in subsequent slices as the flow evens out and approaches



**Figure VI—31: Total pointwise correlation ( $RHO$ ) at each slice for each turbulence model; (a) Spalart-Almaras (b) realisable  $k-\varepsilon$  (c) RNG  $k-\varepsilon$  (d)  $k-\omega$  models.**

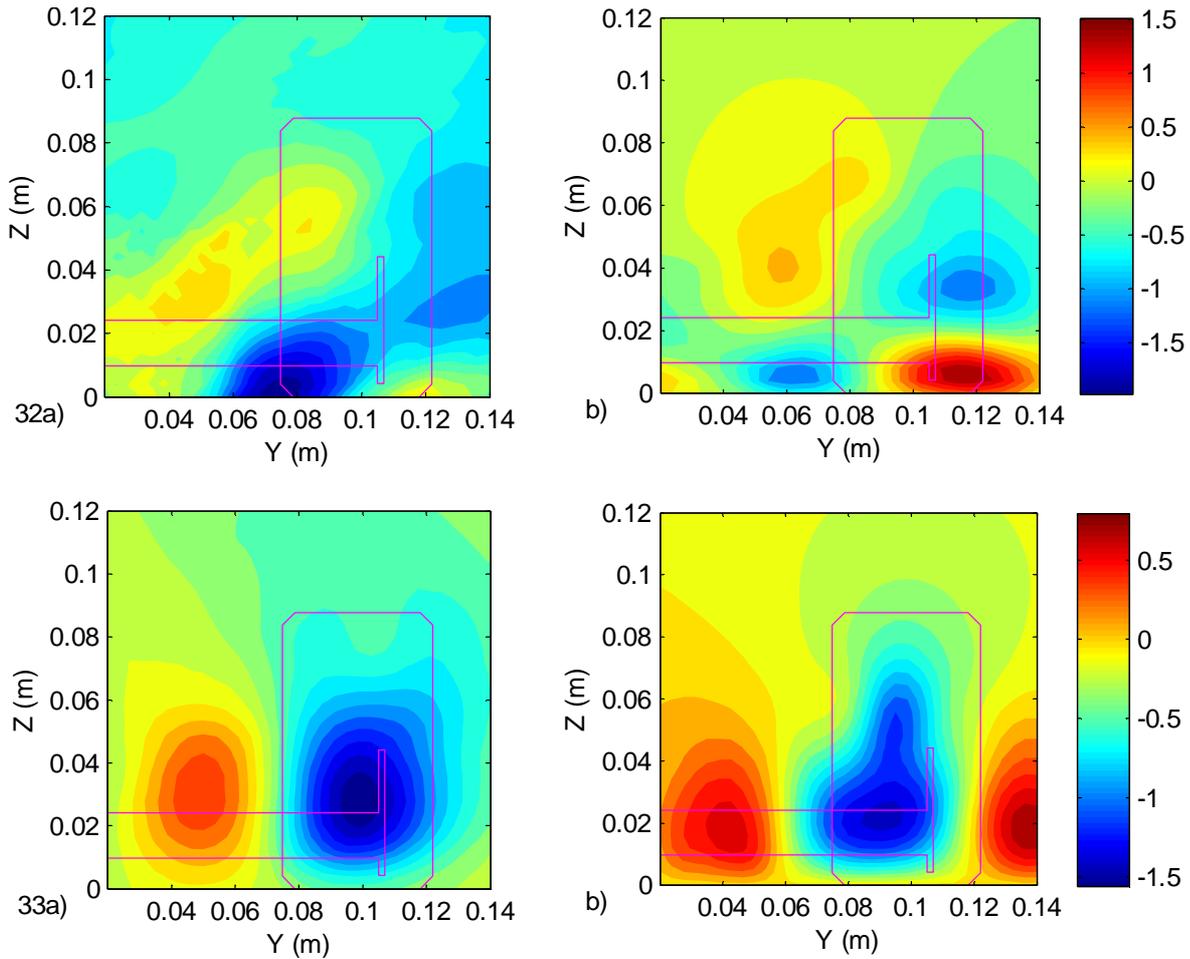
uniformity once again. Certainly Equation (6.3) works in the sense that the sum-of-squares will go to zero as the two datasets approach congruency, but it is not an entirely satisfactory metric away from this ideal. It obviously suffers from scaling problems – measurement planes on which there is a great deal of spatial variation or structure are penalised with respect to those planes that have a more limited range. This is apparent in the diminishing sum-of-squares in slices three to five. Here it is clear from the original results (Figures VI—3 to VI—7) that the turbulence models are performing no better as the flow develops, yet it appears in Figure VI—30 that they are. The very small sum-of-squares in the first slice does not reflect any better *actual* agreement – as evidenced by Figure VI—29, but merely the smaller range of velocities there. Furthermore, whilst the numerical results display appreciably different behaviour depending on the turbulence model used, these differences do not translate obviously to the sum-of-squares: all four graphs have similar characteristics.

A more sensible pointwise evaluation of field similarity would be the familiar correlation coefficient, but formed between pairs of coincident data over the domain:

$$RHO = \frac{1}{LDA} \frac{1}{CFD} \frac{1}{n} \sum_{i=1}^n (u_i^{LDA} - \bar{u}^{LDA})(u_i^{CFD} - \bar{u}^{CFD}) \quad (6.4)$$

This metric is calculated by interpolating the numerical results at the LDA measurement locations, and then using the resulting data pairs to evaluate the summation in Equation (6.4). The resulting correlation coefficients, presented in Figure VI—31, vary between zero and one. This pointwise correlation coefficient echoes the kriging correlation coefficient  $\eta$  (Figure VI—23) in many respects. This is because it is essentially an estimate of the normalised cross-covariance function evaluated at the origin  $C_{\beta}(\mathbf{0})$  – provided the interpolations to experimental nodes do not bias the statistic. In effect, this metric is therefore subsumed anyway by the more complex spatial covariance model  $C_{\beta}(\mathbf{h})$ . Note that the correlation coefficient does not suffer from the same scaling problems as the sum-of-squares, as it is normalised by the standard deviations  $\sigma^{LDA}$  and  $\sigma^{CFD}$ , and centred by the means  $\bar{u}^{LDA}$  and  $\bar{u}^{CFD}$  of each of the nodal data. In general, the kriging coefficient in Equation (6.2) finds better correlation between the computed and experimental results than its counterpart in Equation (6.4), and varies less. This may be because it incorporates more than just the pointwise difference in nodal value. By indirectly examining the data around the point, effectively more information about structural correlation is made available to the statistic in Equation (6.2).

A notable theoretical difference between the kriging coefficient and Equations (6.3) and (6.4) lies in the way that spatial distortions are represented. The pointwise metrics in Equations (6.3) and (6.4) perceive coherent spatial distortions between the LDA and CFD results in the same way as outright structural discrepancies. In other words, a totally spurious structure in one set of results might easily generate the same set of pointwise differences as a mild distortion or amplification of a bona-fide structure. An obvious example is that of a gross spatial shift of flow structures in the LDA results as compared to where they appear in the numerical work. A spatial shift of this nature will manifest itself as a correlation peak in the cross-covariance function that is shifted from the origin. Such spatial shifts are currently filtered in the generation of spatial statistics – the cross-covariance statistics are re-centred and the shift recorded – so they do not effect the statistic in Equation (6.2). Note that the decision to subsequently use the spatial shift in estimation has



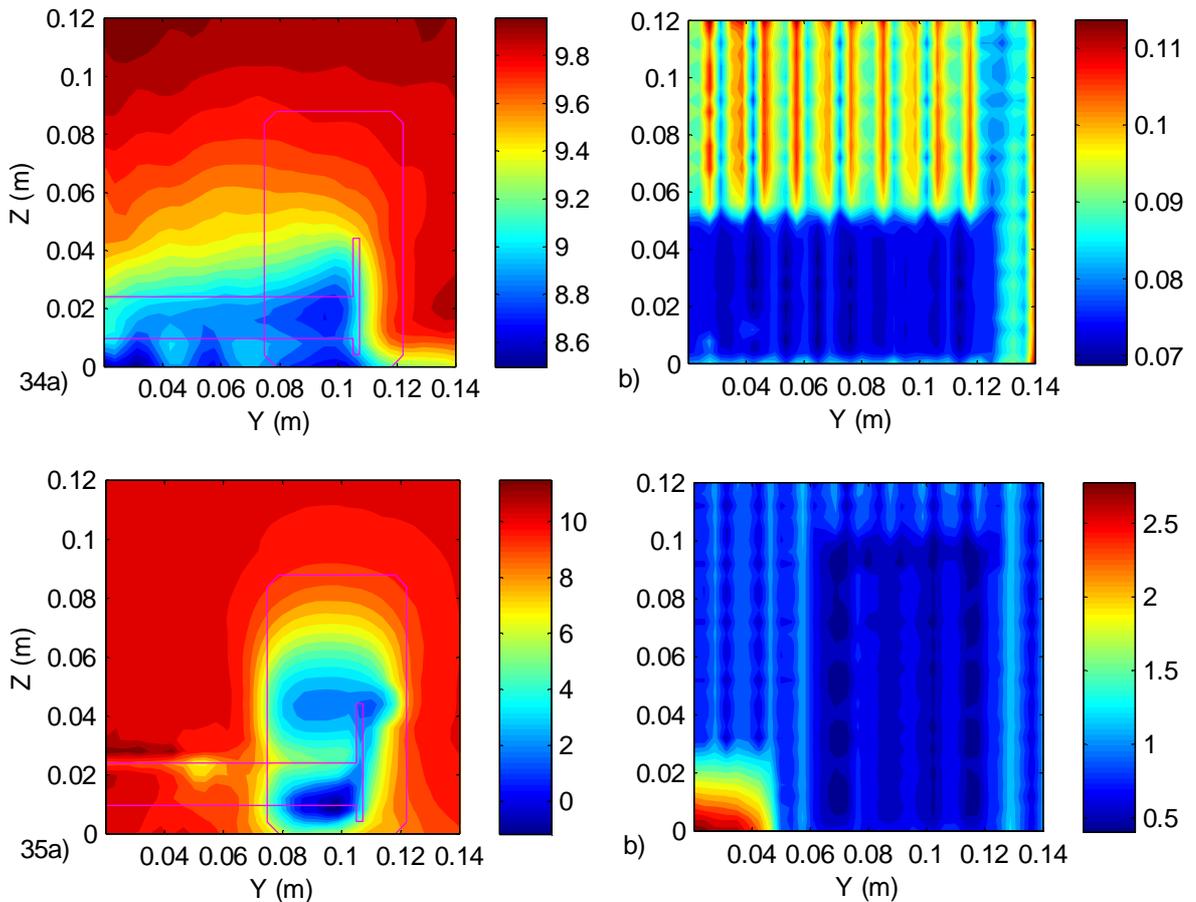
**Figures VI—32 and VI—33:  $y$ -velocity and  $z$ -velocity (m/s) respectively at slice five; (a) raw LDA results, (b) raw CFD results using RNG  $k$ - $\varepsilon$  model.**

then to be made separately. By contrast the metrics in Equations (6.3) and (6.4) will perceive a spatial shift to be no more remarkable than an unstructured dithering of results.

There are some peculiarities to the results in Figure VI—23 that may be put down to a lack of robustness in the covariance model fit. The  $y$  and  $z$ -velocity components at slice 5 of the RNG  $k$ - $\varepsilon$  model are of interest as they show respectively low and high kriging correlation,  $\eta$ . The original results in each case are shown in Figures VI—32 and VI—33. Whilst some structural similarity is apparent for both components, there is a fair amount of distortion and the fabrication of a structure in the CFD around  $Y = 0.13$ ,  $Z = 0.01$  – particularly apparent in the  $z$ -component. However, in spite of this region the  $z$ -velocities achieve better basic correlation coefficient (Equation (6.4)) in Figure VI—31, and otherwise display reasonable agreement. The metric in Equation (6.2) is doing broadly the right thing, however it

does seem strange that the  $z$ -velocity results should return such a good kriging correlation, which is thought to be due to sensitive covariance modelling, and a strong basic correlation coefficient. Furthermore, as has been previously remarked the inter-relation of the swirling components is not captured by the simple bivariate cokriging attempted here.

The validation metrics presented in Equations (6.3) and (6.4) are indicative of pointwise validation metrics in the literature, which are often more complex and have a greater emphasis on classical validation methodology. Oberkampf and Trucano [34] extend the basic principle with their validation metric, adding error normalisation and local hypothesis testing, and Stern and Wilson [36] propose an interval of validation, which is modelled after techniques in Modern Design of Experiment. These statistics are far more specialised towards classical validation problems, and both fundamentally overlook spatial correlation of the data fields.



**Figures VI—34 and VI—35: Cokriged LDA estimates incorporating  $k$ - $\omega$  results on slices one and two of; (a) experimental  $x$ -velocity (m/s), (b) associated kriging standard deviation.**

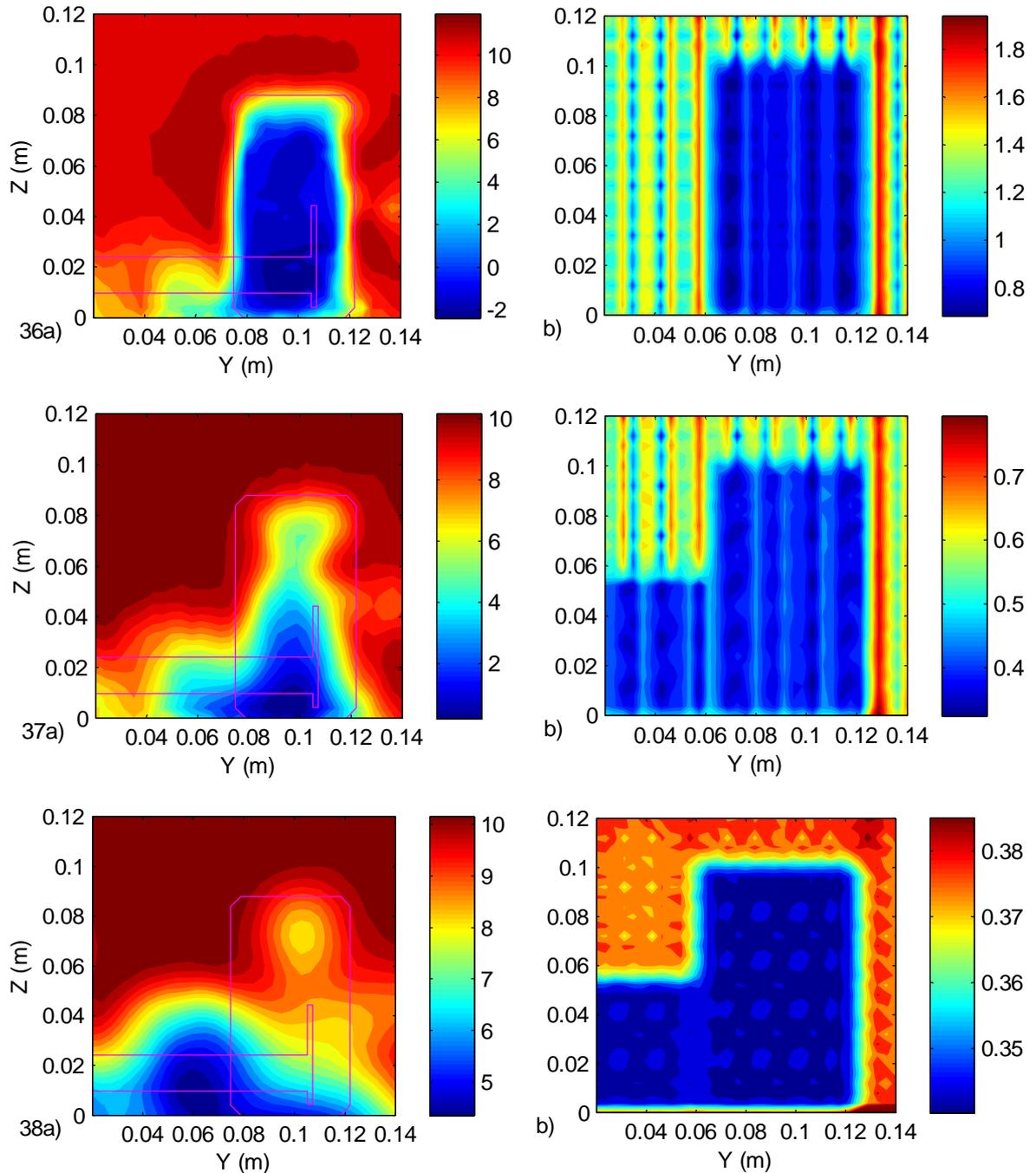
They aim to achieve spatial validation by considering many repeated instances of a functional validation. This is acceptable for non-spatial summaries – such as lift or drag, but entirely neglects the overall picture when the data are spatial field variables.

## VI - 4. Cokriging and Blending

Establishing spatial correlation between datasets also affects the individual interpretation of each dataset with respect to its estimation. Having identified cross-covariance models in the previous sections, a logical next step is to perform a cokriging which uses these models to support estimations. The result is a blend of primary data; the LDA results, and the secondary data; the numerical model results. These techniques are applied to the raw data provided by Diasinos, and then to an artificial modification of these raw data to demonstrate their utility.

### VI - 4.1. Cokriging in estimation of the raw field data

A bivariate cokriging of the LDA data, supported by corresponding  $k$ - $\omega$  numerical data, estimates rasters of points which are contoured and presented in Figures VI—34(a) to VI—38(a) – the kriging variance corresponding to these estimations is plotted in parts VI—34(b) to VI—38(b). The overall appearance of these kriging variance plots is essentially similar to that of those presented for univariate kriging in Figures VI—17(a) to VI—21(a). This is because the new spatial data are spread relatively evenly over the entire domain – the numerical grids supplied at these slices are relatively fine – i.e. the mesh is relatively well refined, and so any change in estimation variance is also uniform. Further cokriged estimates are presented without the accompanying kriging variance, which incorporate the results of the Spalart-Almaras model in Figures VI—39(a-e), the realisable  $k$ - $\epsilon$  model in Figures VI—40(a-e), and the RNG  $k$ - $\epsilon$  model in Figures VI—41(a-e). In all cokriged results the primary (estimated) variable is the LDA dataset, whose estimation is now supported by secondary numerical data – which was presented in Figures VI—3(b) to VI—7(b) and Figures VI—24 to VI—28. Whilst correlation is expected between the datasets, it is assumed that the data may manifest different

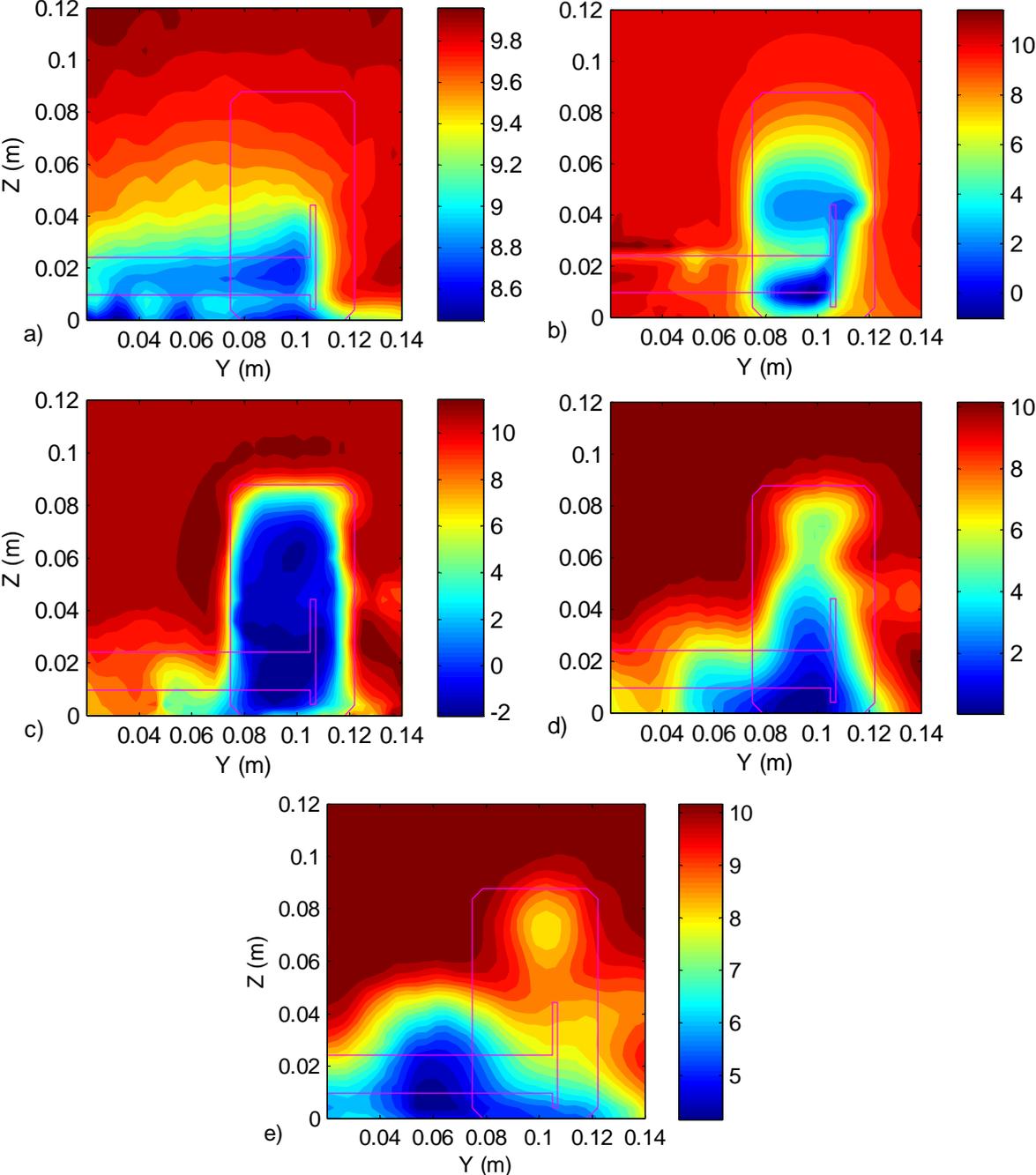


**Figures VI—36 to VI—38: Cokriged LDA estimates incorporating  $k-\omega$  results on slices three to five of; (a) experimental  $x$ -velocity (m/s), (b) associated kriging standard deviation.**

drifts, and thus the unbiasedness conditions described by Equation (3.54) and solved in Equation (3.55) are used.

One of the original questions that was asked in this thesis was how to blend disparate results and result-types meaningfully to make a best estimate of the reality of the physics. Cokriging addresses this question and the cokriged contour plots in Figures VI—34 to VI—41 are modified by the extra numerical data. However, note

that the cokriging still aims to make an unbiased estimate of the primary variable – the secondary information merely aids in the estimation, and should introduce no bias provided that the assumptions on the nature of the mean are correct. Therefore, the main effect of the cokriging is to modify the existing data away from primary node-points or where the primary data is sparse – for datasets that display good correlation. As the best kriging correlation is in general displayed by the  $k$ - $\omega$  model, it is primarily the features that emerge in this model that are discussed below.



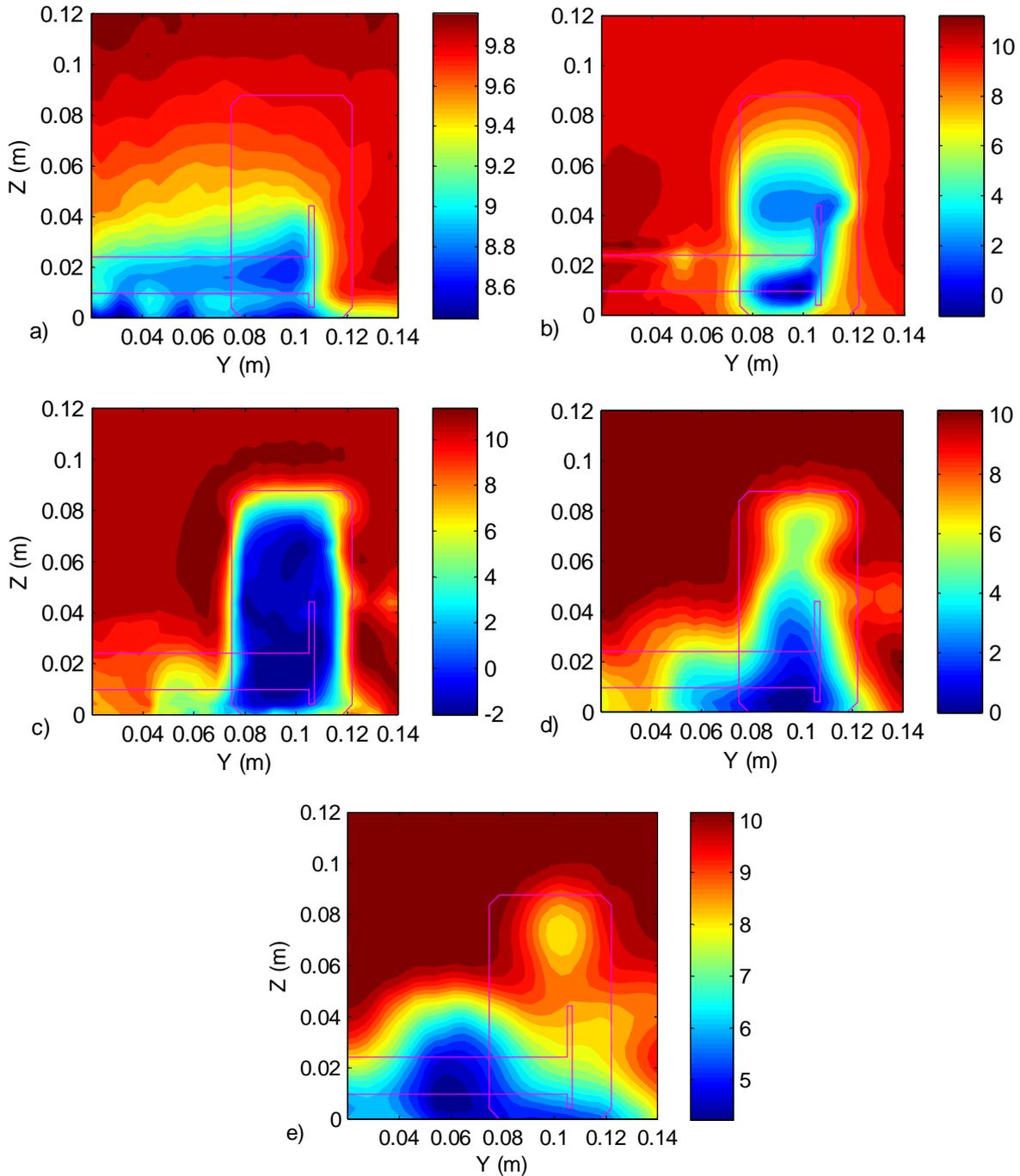
**Figure VI—39(a-e) Cokriged LDA estimates (m/s) incorporating Spalart-Almaras results on slices one to five.**

For the first slice presented in Figure VI—34, very little is actually modified by the cokriging, despite the reasonably good correlation suggested by the covariance model. The only real difference is that because the nugget effect happens to have been fitted differently – which is necessary to preserve the positive definite constraints implicit in the linear model of coregionalisation – the spatial smoothing is reduced by comparison with Figure VI—12. It often occurs that the introduction of the cross-covariant components and their modelling via the linear model of coregionalisation, leads to a less satisfactory fit to the spatial statistics for individual auto-covariant structures. The fit to individual sets of statistics  $S_\beta$  is sacrificed for a better global fit to all of the statistics, subject to the constraints imposed by the relation in Equation (3.70). The limited modification on this plane is due to the observation that the structural elements of the numerical and experimental results actually agree quite closely – there is simply nothing to modify. The only way in which they differ, is that the experimental results obviously comprise some statistical variation (spatial noise), which is absent from the numerical results. Note that apart from estimation, this also acts to reduce the kriging correlation  $\eta$  for the  $x$ -velocity on slice one.

The second slice is notable in that it displays good agreement and excellent kriging correlation across all turbulence models. Furthermore, unlike the first slice, there are some features that the computed results resolve that are not resolved by the LDA survey. Consequently, in Figures VI—35, VI—39(b), VI—40(b) and VI—41(b), there are substantial modifications to the original kriged estimates, shown in Figure VI—13(a). In particular, note the addition of a wake behind the wing running inboard (to the left in Figure VI—35, negative  $y$ -direction) of the wheel. In the original experimental survey, the LDA probe could not access the area immediately behind the wing on the second slice and there are actually no experimental results in this region, as evidenced by the triangulation in Figure VI—10(b). The kriged estimates in Figure VI—13(a) merely extrapolate an average velocity from local measurement points into this region. However, the cokriged results clearly show that the wake behind the wing has been blended into the experimental result because of the high spatial correlation of the numerical and experimental results. Furthermore there is a modified interaction with the moving ground underneath the wing which is

edited in, as well as a more pronounced boundary layer at the moving ground under the stagnation region in front of the wheel.

The cokriging at slice three is also modified by the extra data, if somewhat less dramatically because the kriging correlation diminishes here. The small wake behind the sting outboard of the wheel wake in Figures VI—5(a) and VI—14(a) has been sampled too coarsely by the LDA to resolve it properly, however the structure



**Figure VI—40(a-e) Cokriged LDA estimates (m/s) incorporating  $k-\epsilon$  realisable results on slices one to five.**

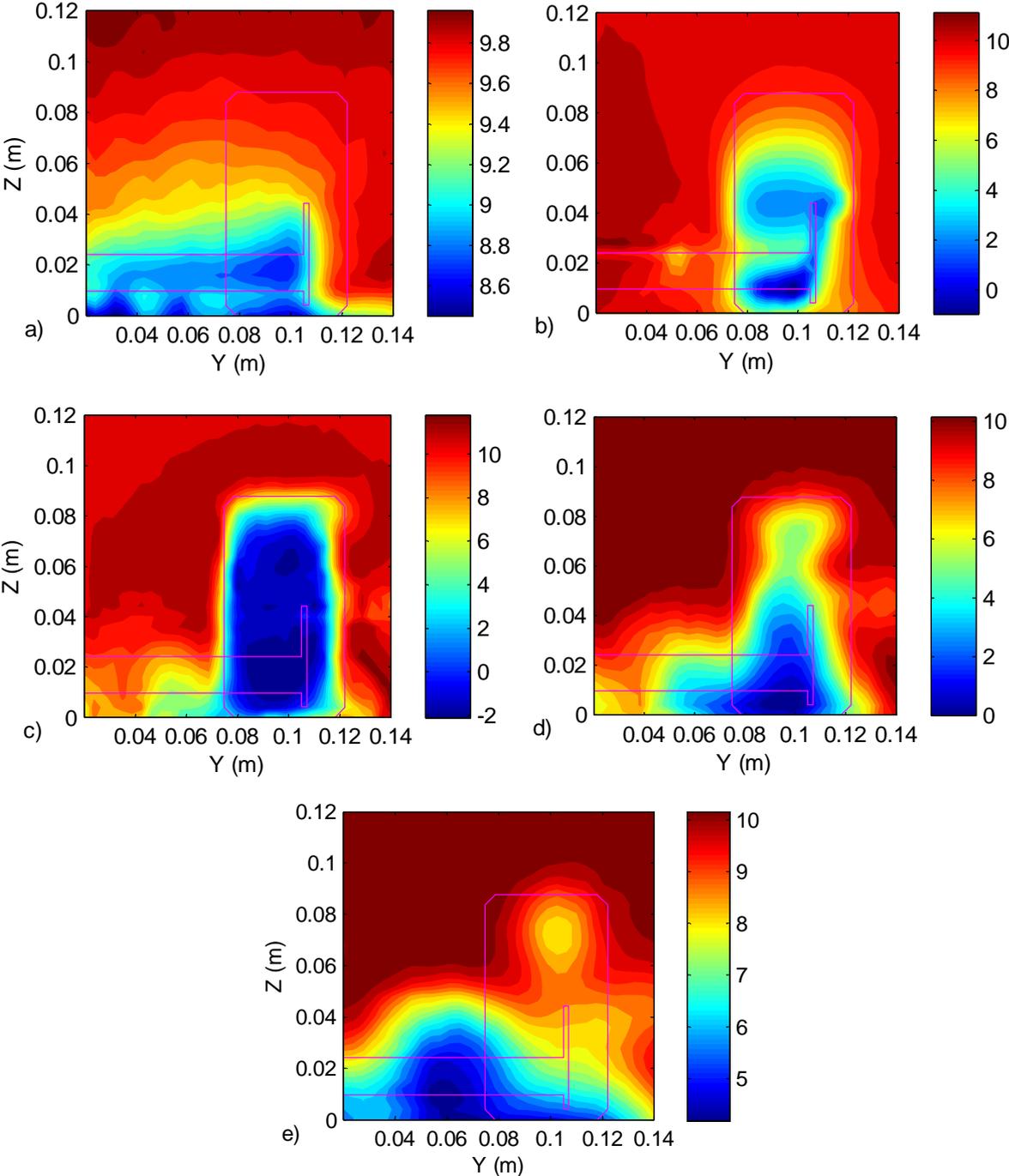
is resolved with better clarity when the numerical result is incorporated in Figure VI—36(a). It can be seen in Figure VI—19(a) that the region inboard of the wheel-wake is also sampled coarsely, which means that in the cokriged estimates, a number of structures are modified. The wake area centered on the vortex inboard of the wing (centre  $Y = 0.05$ ,  $Z = 0.01$ ) is previously roughly captured in Figure VI—14(a), but in VI—36(a) it is amplified somewhat as it has been well described by the extra numerical data. Also, the still-dissipating wake of the wing in Figure VI—14(b) now also presents weakly in Figure VI—36(a). More noticeably there is a strong boundary layer at the bottom of Figure VI—36(a) as the moving ground is approached. Experimental data is relatively sparse (and in any case unreliable) close to the walls and boundaries of the wind tunnel, but numerical data must resolve the boundary layer whether by wall functions or local mesh refinement. The cokriging has blended these two extremes.

There is also some modification to the region of low  $x$ -velocity directly in the wake of the wheel shown in the third slice (Figure VI—36(a)), wherein the rather intricate wake and vortex structure predicted by the numerical work is now faintly evident. This is interesting, as one might be doubtful of the physicality of the numerical model here – the intricate steady state average velocity gradients are not perceived to any conclusive extent in the raw experimental data, nor would they be expected in a realistic, transient turbulent flow. This draws attention to the fact that the cokriging is not an ‘intelligent’ procedure. It has blindly smoothed results on the basis of the overall correlation on slice three regardless of local variations in the cross-covariant behaviours, or of what one’s intuition or prior knowledge would suggest. However, if one accepts the narrow definitions of expectation and variance as defined by the covariance model, then these estimations are at least ‘best’ by some quantifiable measure. This is an example of the inability of the covariance model to supplant actual problem understanding, in spite of its usefulness as an impartial tool.

The cokriged estimates on the fourth and fifth slices; in Figures VI—37(a), VI—38(a), and parts (d) and (e) of Figures VI—39, VI—40 and VI—41, by virtue of their relatively lower kriging correlation  $\eta$  display less modification to the original contours. In most of the cokriged results on slice four, there is some improved representation of the continuing wake behind the sting and some small changes to the

moving ground boundary layer. These effects are rather minor compared with those seen previously, and the same might be said of slice five. What this means is that unless there is reasonable spatial correlation between the datasets, it makes no sense to blend them – they are representing essentially different phenomena.

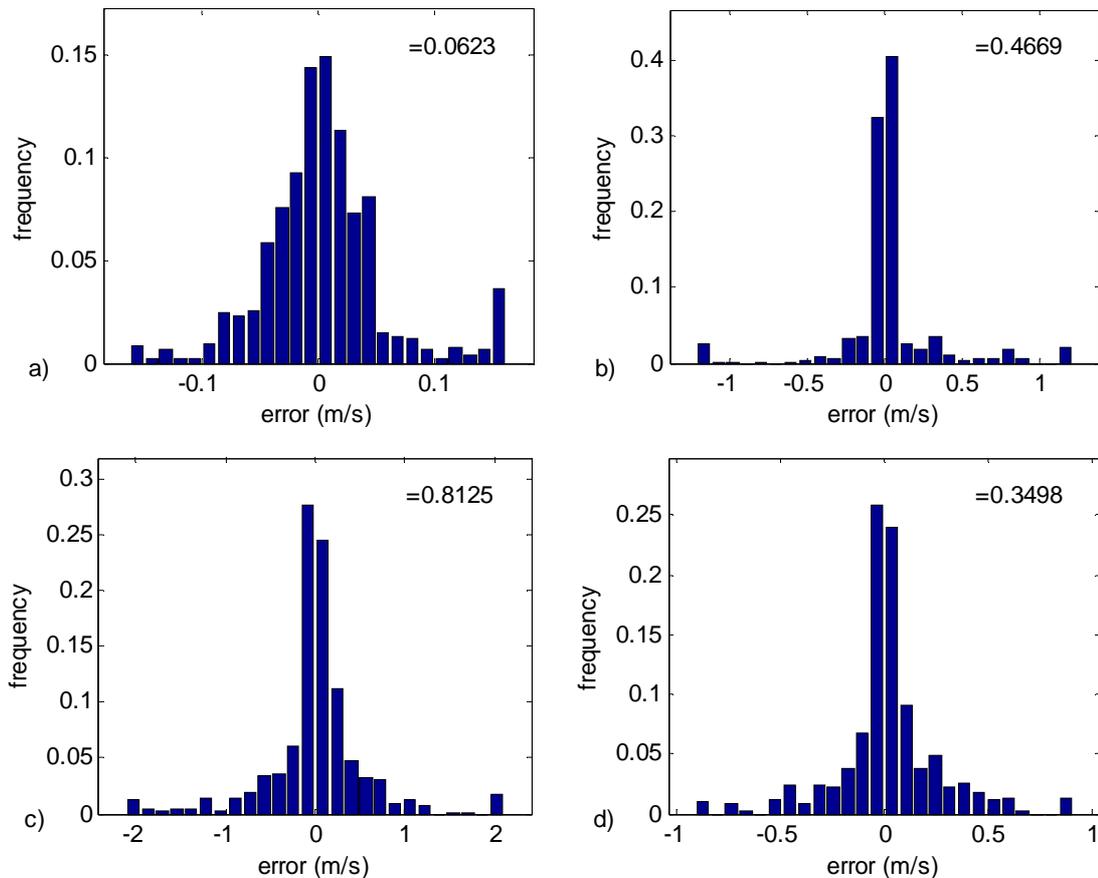
Histograms of cross-validation errors for the cokriging estimator used with the  $k-\omega$   $x$ -velocity results and independent linear drifts are shown in Figures



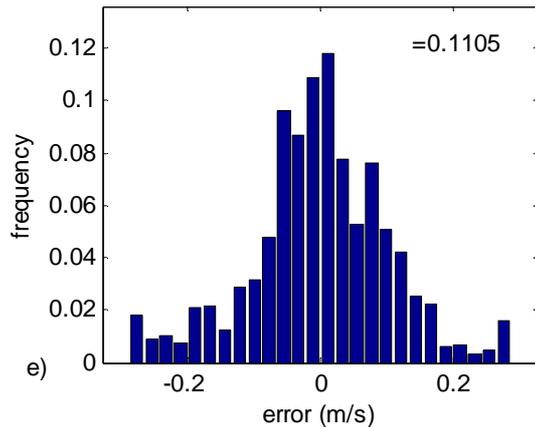
**Figure VI—41(a-e) Cokriged LDA estimates (m/s) incorporating  $k-\varepsilon$  RNG results on slices one to five.**

VI—42(a-e). In the first and second slices an improvement of around 15% is seen in the standard deviation of cross-validation errors. This is very important as it indicates that in regard to a leave-out-then-estimate exercise, the ‘background’ numerical data is aiding the estimation of the experimental data. However, on the other slices the spread of cross-validation errors is larger for cokriged estimation, reinforcing the observation that the agreement of the numerical models is rather poor in the wake. It is interesting that this is also the case for slice three which, notwithstanding its poor performance in cross-validation, does fare quite well with respect to the kriging correlation  $\eta$  in Figure VI—23(c). This indicates that it was possible to fit a quite flattering covariance model for the data here, possibly because the broad structures agreed quite well. Nevertheless, because of the intricate structures in the numerical results, the cokriging estimation adds extra detail which then increases the standard deviation of cross-validation errors.

The overall appearances of Figures VI—42(a-e) are however generally



**Figure VI—42: Frequency histogram of LDA  $x$ -velocity (m/s) cokriged cross-validation errors using secondary  $k$ - $\omega$  results on; (a) slice one, (b) slice two, (c) slice three, and (d) slice four.**

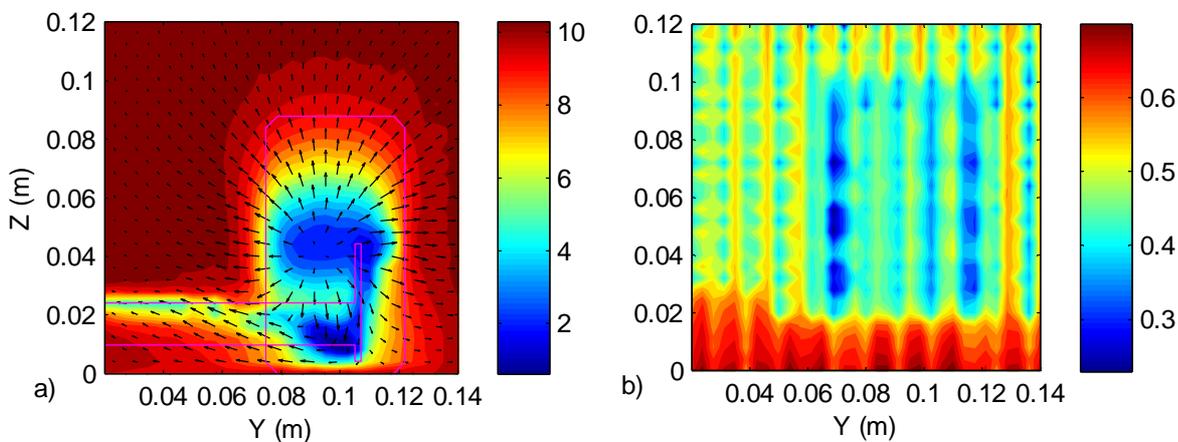


**Figure VI—42(e) Frequency histogram of LDA  $x$ -velocity (m/s) cokriged cross-validation errors using secondary  $k$ - $\omega$  results on slice five.**

comparable to those histograms seen previously in Figures VI—22(a-e): bell-shaped histograms with an exaggerated peak and rather heavy tails result. They are nevertheless centred around zero cross-validation error, apart from a set of positive outliers in Figure VI—42(a) which are probably caused by some badly modelled nodes. In other words, the blending of secondary data does not affect the basic properties of the estimator.

#### VI - 4.2. Cokriging for the inference of modified field data

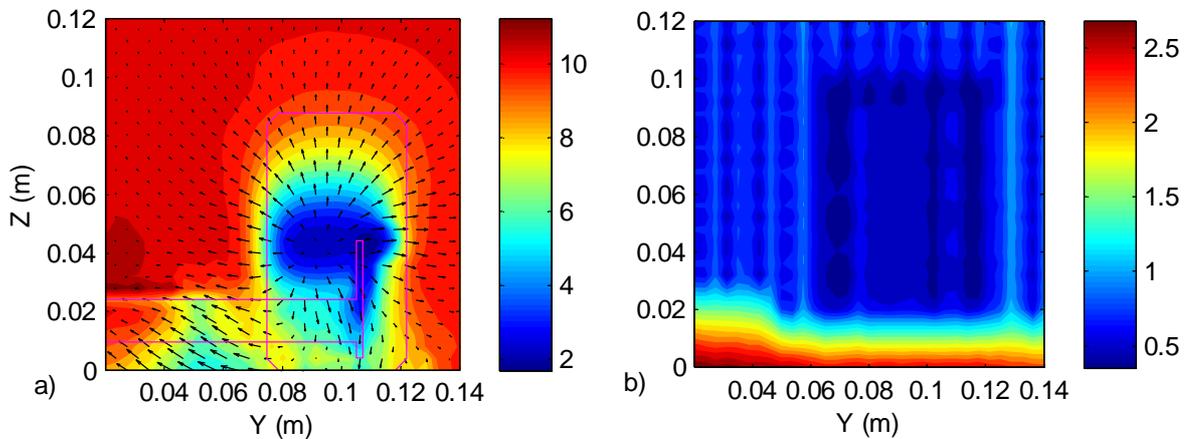
It was remarked in the previous section that the wake of the wing in slice two, that was not captured by the LDA probe, was effectively edited in by cokriging estimation in Figure VI—35(a). This has occurred as the secondary cokriged data are



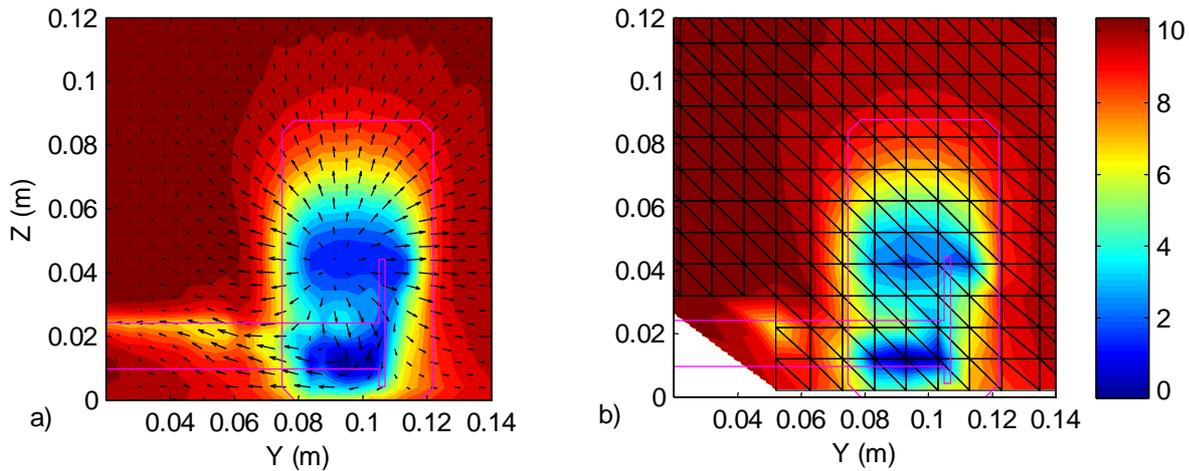
**Figure VI—43(a) Cokriged LDA velocity (m/s) on slice two, using  $k$ - $\omega$  results as a secondary variable to reconstruct blanked LDA data from  $z < 0.02\text{m}$  and, (b) attendant kriging variance for the  $x$ -velocity estimates.**

spatially well correlated to the primary LDA data. In this section, this kind of blending is extended further by the artificial removal of LDA data – the flow-field is then reconstructed in these areas using the CFD data instead.

In Figure VI—43(a) a contour plot of  $x$ -velocity over the second slice is presented overlaid with a vector plot of in-plane components. This is a cokriged blend of the LDA and CFD data estimating each of the velocity components independently, wherein the LDA is the estimated (primary) variable. What is remarkable about this plot is that the nodal LDA data below  $Z = 0.02$  m have been blanked out for the purposes of estimation, and so all structures that are observed for  $Z < 0.02$  m have been informed solely on the basis of the CFD data. These structures include the wing's wake and its interaction with the stagnation area in front of the wheel, and the lower stagnation point above and upstream of the wheel's contact patch. However, unlike the plots in Figures VI—34 through VI—41, this interpolation has been produced under the assumption of a common drift, as described in Section III - 6.2. This has the effect of making the blending strong by comparison with an interpolation based on independent drifts, shown in Figure VI—44. This is because under the assumption of a common drift, it is not just the random components that are assumed to bear correlation. The reason why common drifts were not adopted in the previous section was that more generally, they tend to distort cross-validation plots – indicating that the estimator may become biased as a result of the stronger assumptions on the mean. This is overlooked for the purposes of utility here, but further discussed later in Section VI - 6.1. Plots of the root of



**Figure VI—44(a) Cokriged LDA velocity (m/s) on slice two, using independent drifts and  $k$ - $\omega$  results as a secondary variable to reconstruct blanked LDA data from  $z < 0.02$ m and, (b) attendant kriging variance for the  $x$ -velocity estimates**



**Figure VI—45(a) Cokriged LDA velocity (m/s) on slice two, using  $k-\omega$  results as a secondary variable to reconstruct coarsely sampled LDA data shown in (b) raw contours of  $x$ -velocity based on a coarsely sampled grid.**

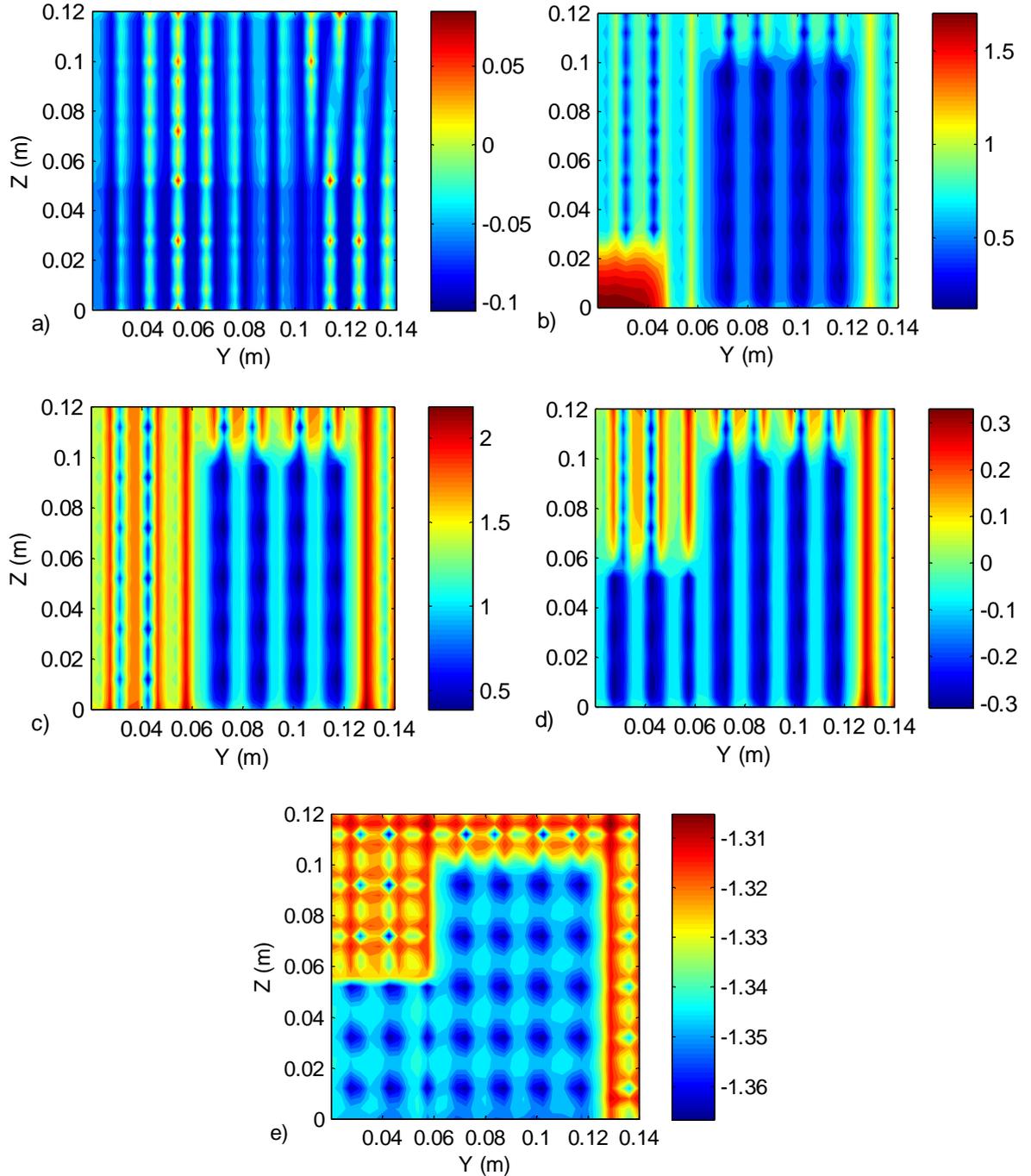
kriging variance associated with the  $x$ -velocity estimations in Figures VI—43(a) and VI—44(a) are presented alongside in Figures VI—43(b) and VI—44(b). Kriging variance is now much higher where data was blanked out.

The results of a second such data blanking exercise on the  $x$ -velocities on slice two are presented in Figure VI—45. Here, instead of blanking out and then reconstructing a contiguous area of nodal data, LDA data nodes are stripped from the area of high resolution sampling that occurs directly in front of the wheel, seen in Figure VI—10(b). These nodes are stripped out in such a way as to leave the underlying coarse, square sampling design – as shown in Figure VI—45(b), where a contour plot is drawn on the resulting triangulation. In this conventional plot, stripping out the data has left the stagnation area in front of the wheel badly resolved. However, in the cokriged interpolation presented in Figure VI—45(a) the stagnation area is properly resolved once more due to the incorporation of the secondary computed data.

The ability to infer an experimental result on the basis of strong correlation with a numerical model is potentially very useful. Wind tunnel data are often constrained by probe access requirements, for which case flow visualisation may be enhanced. In particular, the opportunity for high-resolution experimental surveys to be curtailed and then inferred on the basis of good numerical correlation promises better use of wind-tunnel resources.

## VI - 5. Variance Reduction

Generally in Figures VI—34 to VI—41, it is found that the estimation is most influenced by the computed results where the primary data are sparse and the correlation between the spatial fields is good. Naturally, if the primary data and secondary data correlate perfectly, reproducing exactly the same auto and cross-



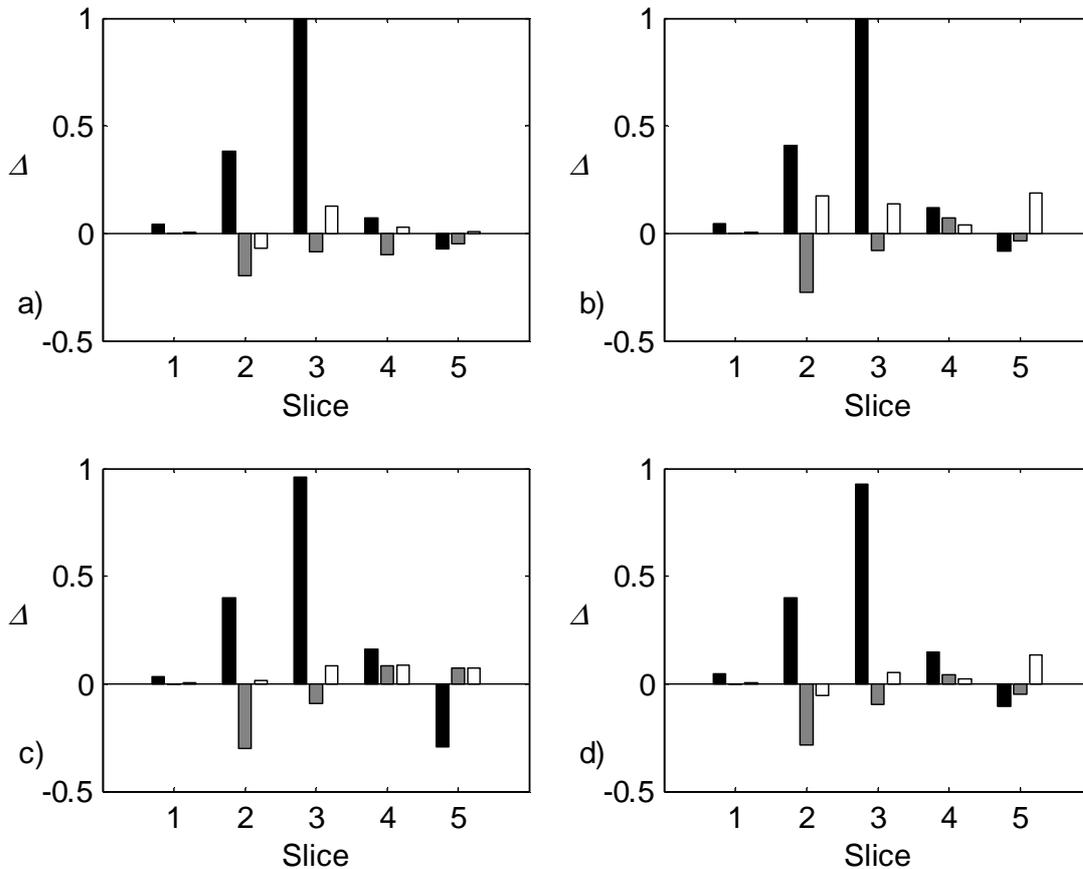
**Figure VI—46(a-e) Local reduction  $v(x)$  in kriging standard deviation (m/s) upon cokriging with  $k-\omega$  model CFD results, slices one to five.**

covariance functions, there will be no change between the kriged and cokriged contours. However, what will change is the kriging variance associated with the estimation, which ought to diminish in comparison to the univariate kriging variance. The square root of the kriging variance, or rather cokriging variance, associated with the cokriged estimates in Figures VI—34(a) to VI—38(a) is presented alongside them in Figures VI—34(b) to VI—38(b).

The addition of extra, correlated data ought to reduce the modelled kriging variance, as there will now be an extra term in the equation for kriging variance. Using the vector matrix notation in Equation (3.37) the kriging variance in Equation (3.36) may be rewritten as

$$\begin{aligned} \sigma_k^2 &= \sigma_p^2 + \sum_{i,j} w_i w_j \text{Cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) - 2 \sum_i w_i \text{Cov}(P(\mathbf{x}_i), P(\mathbf{x}_0)) \\ &= \sigma_1^2 + \mathbf{w}^T \mathbf{C} \mathbf{w} - 2 \mathbf{w}^T \mathbf{c}_0 \end{aligned} \quad (6.5)(a)$$

to which the basis functions  $\mathbf{F}$  and coefficients  $\mathbf{a}$  can be incorporated,



**Figure VI—47:** Average reduction  $\nu$  in kriging standard deviation (m/s) at each slice for each turbulence model; (a) Spalart-Almaras (b) realisable  $k-\varepsilon$  (c) RNG  $k-\varepsilon$  (d)  $k-\omega$  models.

$$= \sigma_1^2 + \begin{bmatrix} \mathbf{w}^T & \mathbf{a}^T \end{bmatrix} \begin{bmatrix} \mathbf{C} & \mathbf{F}^T \\ \mathbf{F} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \mathbf{a} \end{pmatrix} - 2\mathbf{a}^T \mathbf{F} \mathbf{w} - 2\mathbf{w}^T \mathbf{c}_0 \quad (6.5)(b)$$

which using Equations (3.37) and (3.34), and expanding the second term becomes,

$$= \sigma_1^2 + \begin{pmatrix} \mathbf{w}^T & \mathbf{a}^T \end{pmatrix} \begin{pmatrix} \mathbf{c}_0 \\ \mathbf{f}_0 \end{pmatrix} - 2\mathbf{a}^T \mathbf{f}_0 - 2\mathbf{w}^T \mathbf{c}_0, \text{ so that} \quad (6.5)(c)$$

$$\sigma_K^2 = \sigma_1^2 - \mathbf{w}^T \mathbf{c}_0 - \mathbf{a}^T \mathbf{f}_0. \quad (6.5)$$

In the above relation, the dot product  $\mathbf{w}^T \mathbf{c}_0$  acts to reduce the kriging variance  $\sigma_K^2$  – it accounts for some of the total variance  $\sigma_1^2$ . For the cokriging variance, there are two datasets so that the vector of weights  $\mathbf{w}$  concatenates  $\mathbf{w}^1$  and  $\mathbf{w}^2$ , and the vector of covariances  $\mathbf{c}_0$  concatenates  $\mathbf{c}_0^1$  and  $\mathbf{c}_0^2$ . Therefore, the secondary information serves to decrease  $\sigma_K^2$  by the dot product  $\mathbf{w}^{2T} \mathbf{c}_0^2$ . An analogous relation can also be derived for the kriging equations in Equation (3.58), with common drifts.

The relation in Equation (6.5) is demonstrated when comparing the plots of cokriging variance in Figures VI—34(b) to VI—38(b) with the original plots of kriging variance in Figures VI—17(a) to VI—21(a). The estimation variance generally decreases with the addition of the new dataset. As a measure of local similarity  $v(\mathbf{x})$ , it is sensible therefore to examine the decrease in kriging variance:

$$v(\mathbf{x}) = \sigma_K(\mathbf{x}) - \sigma_{CK}(\mathbf{x}) \quad (6.6)$$

where  $\sigma_{CK}(\mathbf{x})$  is the cokriging variance. This quantity is plotted in Figure VI—46 for the cokriging of the  $k$ - $\omega$  results; the reduction between the cokriging variance in VI—34(b) to VI—38(b), and the original kriging variance in VI—17(a) to VI—21(a). Because it is based on the modelled kriging variance,  $v(\mathbf{x})$  is of similar appearance – it too will reduce to zero exactly at node points and is broadly indicative of the density of the nodal data. In general, large values for  $v$  hence large reductions in kriging variance between the kriged and cokriged results, indicate good agreement whereas small values suggest the opposite. However in Figure VI—46, the metric is based on two kriging estimations which use two different models for covariance, and therefore it is not necessarily positive. The covariance model for the primary variable changes in order to meet the additional requirements of

coregionalisation – if it changes too much, the dot product  $\mathbf{w}^{1T} \mathbf{c}_0^1$  in Equation (6.5) will also change and  $v(\mathbf{x})$  may no longer be representative of the reduction  $\mathbf{w}^{2T} \mathbf{c}_0^2$ .

Note that unlike the metrics presented earlier in Equations (6.2), (6.3) and (6.4), Equation (6.6) is a strictly local measure of agreement – although its calculation is subject to a global model for the covariance functions. There may be situations, involving irregular spatial arrangements of data, or where examining localised phenomena, when it is preferable to use this metric. However, considering the global nature of the assumptions of stationarity outlined in Section III - 1, one would expect Equation (6.2) to provide a more reliable overarching summary of the relationship between the spatial data fields. One way in which the metric in Equation (6.6) can be turned into a global summary, is by simply integrating it over the entire domain and calculating an average value for each slice – presented in Figure VI—47. The domain of the integration is arbitrary and taken here to be the area of the plots. The average reduction in variance is then a function both of the covariance model, and the relative spatial disposition of the nodes in question.

## VI - 6. Alternative Schemes

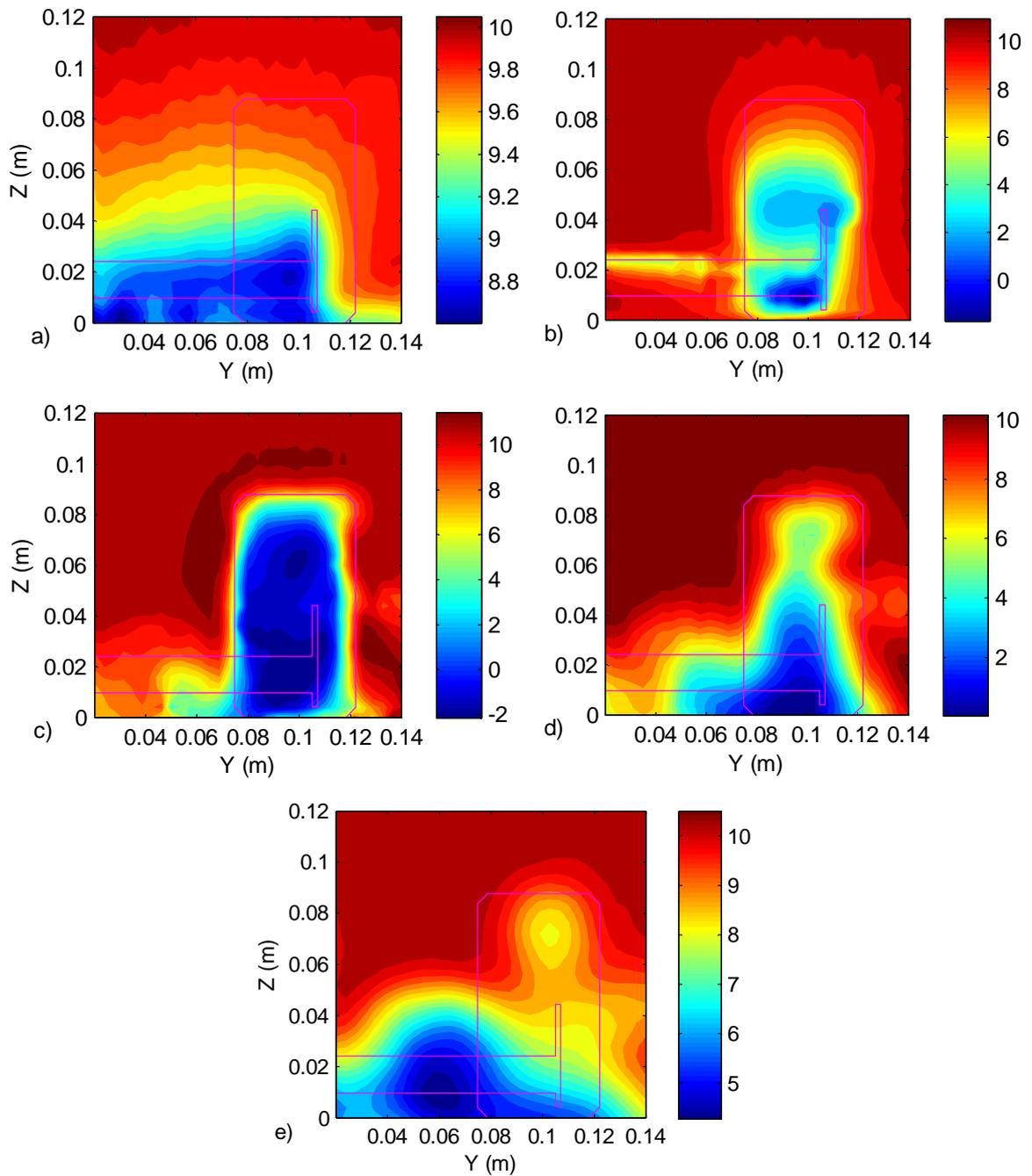
The specific assumptions driving the foregoing cokriging estimations are not exclusive. There are other means within the kriging estimation framework by which such a blending of results can be achieved, and these are examined and evaluated in this section.

### VI - 6.1. Common drift functions

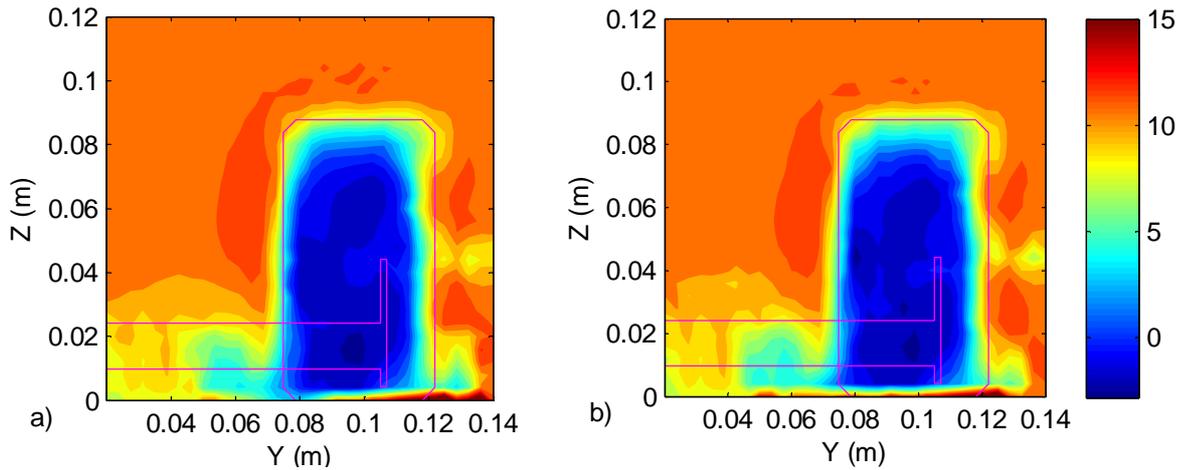
In the cokriged results discussed so far, it has been assumed that the datasets possess independent drift functions, as described in Section III - 6.1. Theoretically, this means that each dataset is estimating a separate phenomenon but there are notionally random parts to the detail of the variation of the phenomena which are correlated. Another way of going about estimation is to assume that the datasets share a common mean, as detailed in Section III - 6.2. This is a stronger assumption, but justifiable in the context of many cokriging estimation problems. In this case the

existence of a common mean can perhaps be representative of some greater truth, towards which all datasets are aimed. Whilst this is a useful conceit, it is flawed in that the datasets will possess different ‘means’, especially if they are describing totally different phenomena.

In Figure VI—48, the cokriged contour plots incorporating the  $k-\omega$  results (previously presented in Figures VI—34 to VI—38) are again produced, but using



**Figure VI—48(a-e) Cokriged LDA estimates (m/s) incorporating  $k-\omega$  results using a common drift model on slices one to five.**

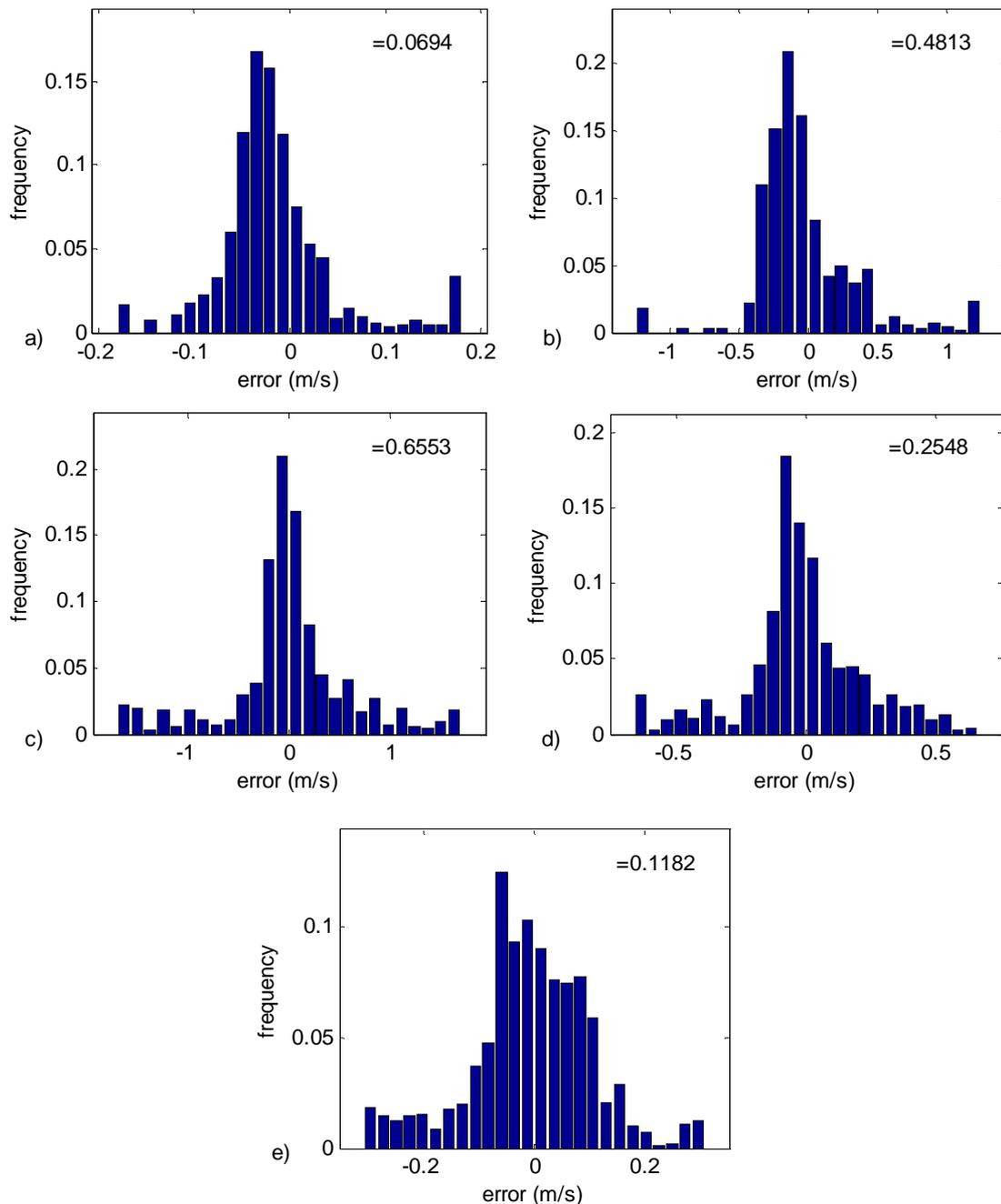


**Figure VI—49: Cokriged LDA estimates (m/s) using a common drift model on slice three incorporating; (a) Spalart-Almaras and (b)  $k-\varepsilon$  realisable results.**

the algebraically dependent means introduced in Section III - 6.2 – the means are identical. Thus the unbiasedness conditions outlined in Equations (3.54) and solved in Equation (3.55) have been used. This allows the sensitivity of the final estimates to the assumptions on the drift model to be assessed. In comparison with Figures VI—34 to VI—38, it is apparent that there is not a huge difference in the estimates – except that one may suspect that there is a stronger adjustment when it is assumed that the results share the same mean. As one of the aims of cokriging estimation was to provide a sensible means of blending results, for this narrow context it may be preferable to use a common drift – indeed, this assumption was adopted in Section VI - 4.2 to reconstruct contiguous areas of blanked data in Figure VI—43.

However there are problems with making a stronger assumption about the mean of the random fluctuation. The estimations depicted in Figure VI—49, like those in Figure VI—48, are made on the assumption that the mean is common to all datasets. By contrast, in Figure VI—49 the Spalart-Almaras and  $k-\varepsilon$  numerical results have been used as secondary datasets, instead of the  $k-\omega$  numerical results used previously. Again the usual extra structures appear, but now there are inconsistencies. Whereas one would expect that the boundary layer here should tend towards a velocity equal to the free-stream velocity of 10m/s, the cokriged estimates range up to 15m/s in Figure VI—49: hence, the boundary layer at the floor of the moving ground wind tunnel is clearly unphysical. To check the validity of the assumptions made in common-drift estimation, a cross-validation exercise is performed on the cokriging estimator used to generate the original interpolations in

Figure VI—49. It emerges that even though these estimations appear reasonable, the histograms of cross-validation error presented in Figures VI—50(a-e) are now variously misshapen or badly skewed away from zero estimation bias. This indicates that the estimations are now biased to some degree. Clearly, to assume that the datasets share a common mean is not a reasonable assumption in this case.



**Figure VI—50: Frequency histogram of LDA  $x$ -velocity (m/s) cokriged cross-validation errors using common drifts and secondary  $k$ - $\omega$  results on; (a) slice one, (b) slice two, (c) slice three, (d) slice four and (e) slice five.**

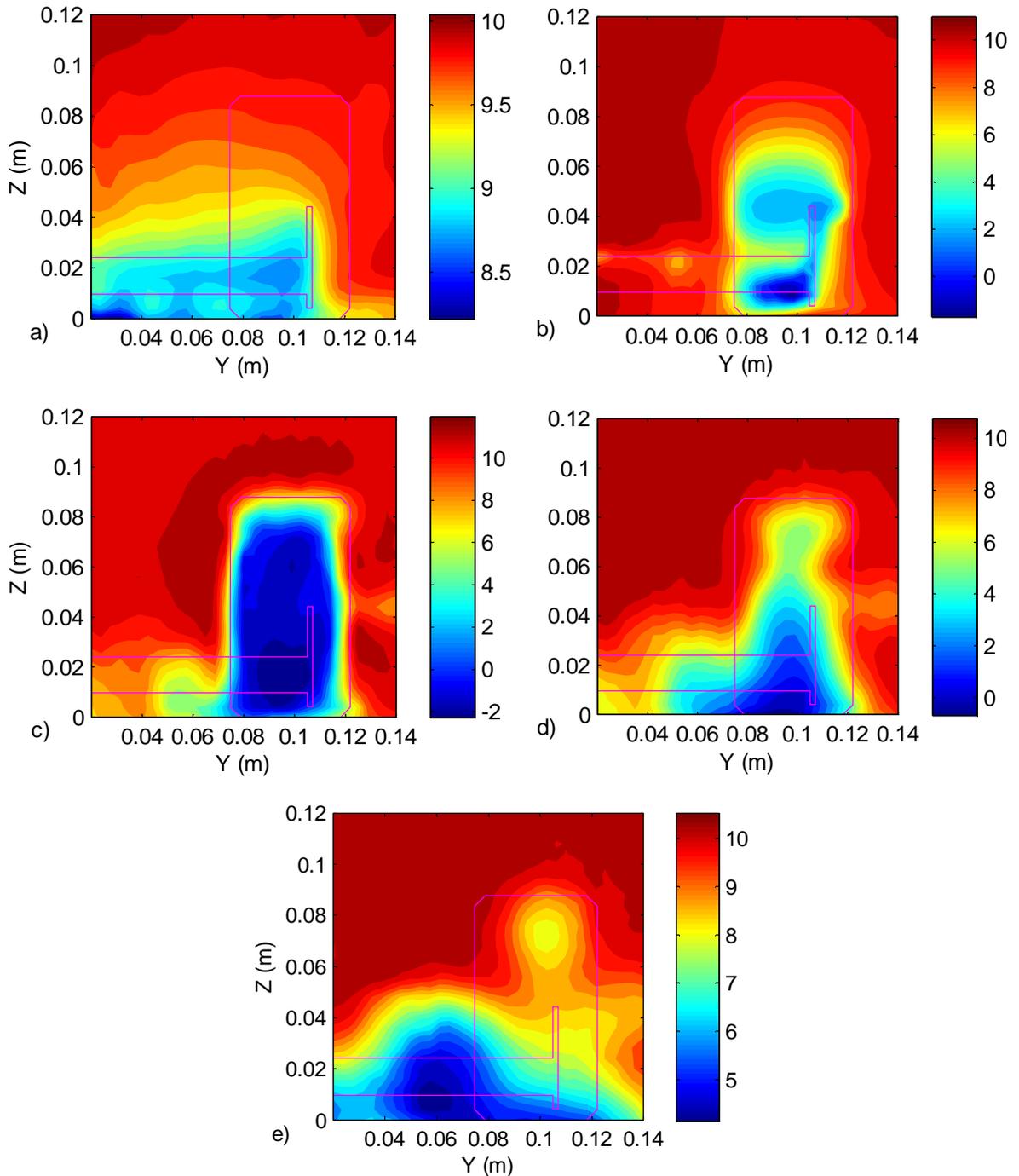
## VI - 6.2. Interpolated drift functions

It was mentioned in Chapter IV that the basis functions  $f^i(\mathbf{x})$  for the drift, as expressed in Equation (3.29) for universal kriging, did not necessarily need to be monomial functions. There is also the option of using interpolated drift functions: an auxiliary spatial dataset is interpolated at all locations where the drift functions are evaluated and these drift function values are used instead of monomial evaluations. Therefore, in addition to blending via direct cokriging, it is also possible to blend linear combinations of datasets by extending the techniques of univariate universal kriging, which circumvents the complexity of coregionalisation and cross-covariant structure identification.

The kriged estimations in Figure VI—51 have been produced in this manner, wherein the  $k$ - $\omega$  results have been blended into the primary LDA data as quasi-drift functions by interpolation at all primary (LDA) data nodes and estimation points. Note that whilst some blending effect is perceptible the adjustments to the raw LDA results are not as pronounced as those of the direct cokriging – by comparison with either independent or common cokriging drifts. It appears that there is some instability in the estimations on slice five, Figure VI—51(e). This is possibly due to the rather poor parity of the LDA and computed results here, which are forming the underlying drift function. This was less of a problem in Figure VI—38(a) for which the covariance matrix decouples the computed results from the primary variable by their poor correlation – however representing the mean by the computed results is a strong assumption. Furthermore there is no parallel to the correlation coefficient  $\eta$ , and for coarse secondary nodal data the drift function interpolation – which has also been effected by kriging – will introduce another source of uncertainty. However, the more straightforward interpretation of the secondary numerical data as a simple analytic function is theoretically attractive.

The last point raises the great weakness of the direct cokriging performed in the previous sections. Namely, it is assumed that some part of the spatial variation of the deterministic numerical data can be modelled as a random fluctuation over the domain of the results. The numerical results are clearly not random – they can be repeated perfectly with effectively zero variance, but they do comprise uncertainties.

Many of these uncertainties may be related to the spatial variation of the numerical solution in space – as was demonstrated in Chapter VI, thus it can be argued that the numerical solutions are sufficiently complex to allow a quasi-random interpretation. In the geosciences where kriging originated, spatial covariance must be characterised from a single realisation – mountains do not move greatly between measurements.



**Figures VI—51(a-e) LDA estimates (m/s) incorporating  $k-\omega$  results as an interpolated drift function on slices one to five.**

The numerical work is similar, in that there is only one realisation of it, and the nodal data it presents are related by unknown mechanisms away from their immediate neighbours.

## VI - 7. Afterword

The use of kriging and basic geostatistical techniques for data assimilation of nodal data arising in practical engineering fluid mechanics has been demonstrated in this chapter. It is felt that the introduction of these techniques to the specialism of fluid mechanics is timely, especially as there is currently a distinct lack of procedure for spatial data assimilation in this field. Whilst there are many theoretical advantages to the proposed techniques, their weakness lies in the difficulty of developing robust algorithms to implement them. This has been made more difficult by also attempting to automate the fitting of covariance models so that the techniques are accessible to fluid dynamicists. As might be expected, these difficulties become more pronounced as the ideal of perfect results correlation is departed from.

The natural ambiguity of the status of the random component in kriging is attenuated to a great extent by this case-study, in regard to its existence in deterministic numerical data. The assumptions of spatial stationarity are used in geostatistics as they allow the quantification of random behaviour from a single realisation of a spatial random event. This is entirely necessary as there are no other realisations with which to infer a covariance model: there can only be one such mountain, or one such valley. However, the only way in which the numerical results can be aleatorically random is in the possible chaotic evolution of their solution schemes, and this is in any case deliberately bounded. Otherwise, they approximate invariant steady-state solutions to the RANS equations (given that they exist) upon which some higher-order, unknown error function is then superimposed – an epistemic uncertainty, therefore. Nevertheless, it has been demonstrated that it is useful to treat the spatial data arising from the macroscopic behaviour of these numerical solutions as random. It can be considered that the relationship between numerical nodes that are not in direct contact is exceptionally complex and sensitive for the majority of problems of engineering interest. In this respect, the numerical

analysis is treated as a black box, whose results are as inscrutable as single realisations of a bona-fide random phenomenon.

The advantage of exploiting geostatistical theory as a tool in this way, is simply that it answers the questions originally asked of this candidature. Stationary covariance functions provide a useful means of articulating the similarity of spatial datasets produced at zero replication, and cokriging provides a means of blending results on the basis of this similarity. For the purposes of the scenario addressed, it would seem that cokriging using independent drifts provides a reasonable estimation platform, but this model is not exclusive. It has been preferred in this case as it provides a reasonable capability to blend results, without greatly biasing the estimator. Furthermore, it is attractive for its simplicity, generality and direct theoretical interpretation. There is no need for special treatment of any particular dataset, such as the interpolation at secondary nodes required in Section VI - 6.2.

All inferences are largely derivative of the quality of the spatial statistics and their covariance modelling, both of which stand to be improved. Practically, the methods presented here are restricted by the need to introduce largely automated covariance modelling – comprehensible to typical fluids specialists, with typical fluids datasets. More robust covariance modelling is required to produce more reliable inferences, particularly when the spatial datasets are in bad agreement and an aggressive sum-of-squares optimality criterion is counter-productive. Overall, it is however surprising that the current implementations work as well as they do, in spite of their lack of sophistication.

# Chapter VII - Conclusions

The aim of this thesis was to provide a method of synthesising the various, sometimes disparate, spatial data that are routinely collected in aerodynamic investigations. This broad objective has been addressed by the creation of a comparatively narrow software tool – but more importantly in the investigation of a problem that is always addressed at some level in science and engineering: the way in which the sum data serves to describe a greater picture is fundamental to the development of any understanding of its parts, from the point of view of the experimentalist, or the numerical analyst, or the designer. This thesis has examined in detail the tools by which the greater picture is put together, which are sometimes neglected for attention to the formidable difficulties in obtaining its parts.

The recognition of the data comparison problem forms the first part of this thesis. As evidenced by the literature review, one of the core difficulties in addressing this problem, is the vast context in which such a method must eventually be applicable. The absence of such pre-existing frameworks has guided this work towards very general statistical tools and models, concentrating on interpolation at its core. Whilst apparently banal, well-founded interpolation is essential because it implicates broader behaviours from incomplete information by implicating a model. Engineering data is always incomplete, often unreliable and for fluid mechanics,

frequently and increasingly spatial. The present author has found geostatistical tools to be a sensible way of addressing the data comparison problem, as they assume the least and observe the most of the data under consideration. Importantly, the model that informs the estimation is expressly based on the *immediate* data – not data that will come to hand, or is meant to be at hand, or a history of similar data, but the sum data that is available at any given point in a study.

## VII - 1. Findings

The spatial data assimilation problem was addressed by the creation of a software tool for characterising spatial correlation, and smoothing or blending nodal datasets. The algorithms used for this tool were demonstrated in the main case-study in this work, presented in Chapter VI, in which nodal LDA data was compared with a CFD model. The interpretation of stationary spatial statistics on numerical data was also addressed as part of this work, and forms the first case-study in which a link is made between numerical solution convergence and the spatial smoothness of the numerical result.

The first case study in Chapter V provides evidence for a link between numerical convergence and solution smoothness as expressed by a stationary covariance model. This is not yet theoretically conclusive but on a practical level it is very useful, given the context of a generalised verification and validation tool: such a link allows verification to be addressed within the same theoretical framework as validation, and dataset comparison. It was observed that by using the basic structure identification algorithms outlined in Chapter IV on a series of successive grid refinements on a benchmark by De Vahl-Davis [112], smoother behaviour of the stationary covariance function at the origin could be associated with better grid resolution. In some situations however, the identification of this trend was confounded by the presence of the underlying drift – which in this case may be idealised as the actual error-free PDE solution. To filter this distortion more systematically, a higher-order *intrinsic* random function model was applied to an even simpler heat conduction problem. In this case, progression towards a cubic generalised covariance function was convincingly demonstrated under successive

grid refinement – the solutions were made smoother by the extra levels of discretisation, which removed higher-order truncation errors. Whilst this is practically useful, the broader theoretical implication is that truncation error in numerical PDE solutions may be idealised as being a particular realisation of a so-called *intrinsically* random function, at least at some order.

The second case-study, presented in Chapter VI, applies the algorithms described in Chapter IV to a realistic problem; the comparison of LDA and CFD nodal data both modelling a wing and wheel rolling in ground-effect, and the evaluation of the turbulence models used in the CFD simulations. The first aspect of this problem lies in the quantification of the spatial similarity of a set of alternative numerical models with reference to a parallel experimental survey. In Equation (6.2) Section VI - 3, the author proposes a metric for spatial similarity based on the covariance model used in cokriging estimation. This metric is demonstrated to have superior properties to the classical pointwise metrics presented in Equations (6.3) and (6.4), especially with regards to stability, scaleability and robustness against dataset distortion. It can be concluded from this investigation that purely pointwise comparisons fail to address the spatial relationship between nodal datasets and thus do not make as much, or even enough, use of the data at hand.

Also demonstrated in Chapter VI in Figures VI—34 to VI—38, is the use of cokriging to effect non-biased estimation of experimental LDA datasets, whilst blending numerical CFD nodal data into the estimation. This capability is further explored in relation to a hypothetical situation in which some of the original LDA data have been blanked, to demonstrate how missing data in a given survey may be inferred. If there exists good spatial correlation between numerical and experimental data in the first instance, flow-field reconstruction from less stringent experimental surveys with a greater reliance on supplementary numerical results is possible. Cross-validation studies were conducted to assess the assumptions made in cokriging about the mean random function component, or drift. The assumption of a common drift blends the datasets to a greater extent, but may bias the estimation of the primary LDA data. Blending is thus implicit when using a common-drift assumption, described in Section III - 6.2, but if the means are assumed to be independent as described in Section III - 6.1, the estimation of the primary LDA variable remains

largely unbiased – as seen for Figures VI—34 to VI—38. The applicability of either assumption depends on the aims of the estimation – blending implies that there is a greater truth that the datasets are aiming to represent, whereas unbiased estimation is a property of the estimator (kriging) subject to the appropriateness of the random function model. The smoothing capabilities of simple single-variable kriging are also explored in Chapter VI, especially in relation to the quantification of small-scale variability via the nugget effect. It is evident that the magnitude of this spatial noise may not reflect the sample variance often calculated on data collection.

For both numerical and experimental datasets, it is realised that the stationarity principle is essential to draw conclusions from spatial data that have been produced at a single replication. In spite of the abundance of nodal data there are usually no repetitions of the entire dataset, which might allow a complete non-stationary covariance model to be inferred. Consequently in this thesis, uncertainty must be viewed as both aleatoric – in the case of experimental data, and epistemic – in the case of numerical data. It is assumed in each case that at some order there is a random process driving the uncertainty.

## VII - 2. Significance

Unlike classical verification and validation methodology, the present work concentrates entirely on the comparison of detailed fields of spatial data. This allows a departure from semantics and process in favour of theoretical assumption and quantitative mathematical description. Whereas traditional verification and validation centres around developing absolutes, the covariance model used in this work aims to develop relations between the fields of data. To this model, more specific error models can be added.

The two core questions that were raised at the very start of this work were; how to compare spatial data in fluid mechanics, and subsequently how to assimilate them? Common techniques in geostatistics and spatial estimation have provided the platform for a synthetic method that addresses these two concerns. There remains much work to improve the basic capability and robustness of the software tool, but the method has been successfully demonstrated. A relative metric for similarity has

been proposed, and spatial data are blended subject to clear assumptions concerning their spatial correlation. Moreover, the proposed method provides a broad framework for the comparison and assimilation of uncertain spatial data, which appears sufficiently flexible to accommodate particularities in data collection or generation, and incorporate techniques in classical verification and validation.

## VII - 3. Future Work

The greater part of this work was spent writing code, which inevitably accretes a list of routines and modules that are to be improved, replaced or more fully tested. Operational issues shall be left to the appendices, and the more fundamental areas in which the implementation can be improved shall be described here. There are theoretical limitations and possibilities in the implementation that also require further investigation.

### VII - 3.1. Covariance modelling

The pattern of code development followed three rough stages; conception, sophistication and simplification. What is apparent from the current product, is that the methods described in Chapter V work best when the underlying models are simplest. By contrast, there is scope in the code for a very complicated covariance model. Whilst this is possible, in hindsight it is not particularly necessary or even desirable.

If the overall covariance model is to be improved by introducing additional variogram models, the possible complexity of their combination must be limited in a meaningful way. In many cases, it was found that the optimisation algorithms fitting the model to the statistics would arrive at a solution where only one or two of the basic models, identified in Section IV - 5.1 were playing any significant part. Thus, to simplify the model and importantly clarify any interpretation of the covariance model, more attention should be paid to selecting 'good' or useful structures in the first place. This may require some input from the user, in spite of the intention to provide an automated tool.

The emphasis on an aggressive best fit to the statistics is also simplistic. There is little point in achieving the very best fit if the resultant structures are unrealistic or not robust against minor deviations to the underlying data and statistics. The use of a robust estimation procedure or a maximum likelihood model would be beneficial in this regard, as again would the provision of user input regarding expected anisotropies or behaviours. It should be noted that the preoccupation with identifying spatial anisotropy in Section IV - 5 was motivated largely by the initial aim of interpolation in non-spatial dimensions. In these problems anisotropy is pronounced as entire structures may move around as geometric design variables change. This remains an important capability, although it was not eventually explored in this dissertation.

It has been argued that adopting a stationary model is necessary for the purposes of drawing some statistical inference from necessarily limited, summarised or single replication data. The stationarity assumption is certainly very useful, but it also presents significant weaknesses – it is difficult to verify and often obfuscates structure identification. It was noted how in Chapter V, the non-removal of a global drift component typically engenders strong anisotropic biases in the stationary statistics, which often complicates fitting a model. Whilst it is possible to remedy the effects of a drift on estimation by using appropriate windowing, it is more difficult to filter out the effects of a drift in structure identification. The most general and powerful way to avoid this problem is to move to a higher order intrinsically stationary model [84], as briefly introduced in Chapter V. The adoption of metrics that annihilate polynomial structures and leave the random component only, neatly sidesteps the need to remove the drift. Given the existence of drift components that are intimately related to the underlying physics and partial differential relationships, intrinsic models present the next obvious phase of development.

A way in which the covariance modelling might be more easily but less fundamentally improved is to adopt a spectral representation of the covariance functions. This is demonstrated by Yao & Journel [120] who use a two-dimensional Fourier representation of the (cross-)covariance functions and apply Bochner's theorem to smooth the functions in the frequency domain to for positive definiteness. The obvious advantage of this technique is that it immediately avails one of the Fast

Fourier Transform algorithms and concomitant spectral theory. The one possible disadvantage is that numerical performance may be compromised at higher non-spatial dimensionalities, as the dimensionality of the Fourier transform is matched by that of the spatial domain. Be that as it may, the method does offer excellent fits to the spatial statistics especially in the cross-covariant case, without the obvious simplifications of the linear model of coregionalisation.

So far, little attention has been paid to the frequency distribution of the spatial data, which should approximate a Gaussian distribution if the data is strictly second order stationary. This is not always the case, although thankfully because of its emphasis on local estimation, kriging is apparently quite robust against such departures. If the probability density function is of special interest and needs to be thoroughly quantified and examined, or does happen to distort the local estimation properties of the technique, there exist extensions to the essential theory such as indicator kriging and disjunctive kriging [86]. It should be noted that distribution estimation is a more difficult problem than is usually addressed by ordinary kriging, which is why it has not been pursued at the current stage of development.

The nugget effect has been implemented in a relatively flexible manner to allow for artificial modelling of behaviours peculiar to the data at hand. If some heuristic or meaningful link can be proposed between the point uncertainties and model uncertainties it may be worthwhile to codify this relation and fix the nugget effect or noise at some level, perhaps on the basis of some metric like Roache's GCI. However in this thesis, the nugget effect has usually been fitted on the basis of the observed data, or best fit to it – as by and large this seems to yield the most consistent results.

## **VII - 3.2. Extension**

There remains a great deal of existing capability in the code which has yet to be fully explored. This is because the algorithms have been designed to work very generally, in up to ten interpolating dimensions and four related data-sets. A relatively small subset of this capability was fully explored in this thesis, so it is worthwhile to surmise how the proposed methods will fare under extension.

It was mentioned in Section II - 7 that the data comparison problem for spatial data might be viewed as a solution clustering problem. It is expected that the expansion of the techniques in Chapter VI to more than just two datasets will yield interpretations of the covariance model which complement this view. In Equation (6.2), a simple spatial correlation coefficient was proposed based on the structured components of the linear model of coregionalisation. When there are multiple datasets, there is obviously more than one such coefficient – indeed for  $N$  datasets, there are  $N(N+1)/2$  such coefficients, including those for auto-correlation which are unity. If these values are arranged in a matrix;

$$\begin{bmatrix} 11 & 12 & \dots \\ 21 & 22 & \\ \vdots & & \ddots \end{bmatrix}, \quad =$$

a very simple clustering of ‘like’ datasets is then formed by the elements comprising the matrix eigenvectors. The most dominant clustering is represented in the eigenvector pertaining to the largest eigenvalue, and subsequently weaker clusters are formed by the other eigenvectors. Note that because the above matrix is positive definite, the eigenvalues are all real and positive and the eigenvectors are orthogonal. More complex singular value decompositions may also be possible, such as canonical correlation analysis (CCA) or other decompositions on the original estimation matrices. This would develop correlations between groups of node points from dataset to dataset – although it must be owned that any interpretation at this level is still only derivative of the original model, which is assessed anyway by Equation (6.2).

So far, it seems that the computational burden is not particularly heavy. The most numerically intensive algorithm is the splitting algorithm to generate the spatial statistics described in Section IV - 4.1, which requires data to be swapped on and off disk. Even though the quantity of data necessitates this, there are ways to speed up this procedure: out-of-core splitting problems have been studied extensively and there are a great many algorithms that can do the job better. These were not pursued for simplicity, and their relevance to the topic at hand. It is envisaged in any case that this step need only eventually only be performed once; when new spatial datasets are added to pre-existing datasets. Therefore, the overall computational burden should

scale with the number of datasets already examined, as a cross-covariant structure will have to be generated with respect to each – thus the computational burden should scale roughly with project size. An aspect of modelling that may not scale as well is the coregionalisation fit described in Section IV - 5.2, although it is noted that this is not currently very numerically intensive.

### **VII - 3.3. Numerical uncertainty**

The first case study was motivated by a curiosity for deeper interpretations of the stationary covariance structures used to model numerical data in the rest of the thesis. Somewhat unexpectedly, this investigation led to some interesting conclusions that may possibly offer another way of assessing the accuracy of spatial discretisation schemes.

Methods for the estimation of error in spatial discretisations of partial differential equations usually attempt to estimate or bound the error using physical or non-physical energy norms, or variants of Richardson's extrapolation. Although heuristic methods also exist, there are few forcibly statistical appraisals of numerical solutions. Should there be a link between numerical solutions over spatial domains and stationary spatial statistics on these solutions, the implication would be for a new kind of error estimator or grid convergence index. Whilst all arguments for this are at this stage putative, the potential usefulness of such a technique makes it very worthwhile to explore.

# References

1. Fefferman, C.L. *Existence and Smoothness of the Navier-Stokes Equation*. 2000 [cited 2008 16/12/2008]; Available from: [http://www.claymath.org/millennium/Navier-Stokes\\_Equations/navierstokes.pdf](http://www.claymath.org/millennium/Navier-Stokes_Equations/navierstokes.pdf).
2. Montgomery, D.C., *Design and Analysis of Experiments*. 6 ed. 2003, Hoboken, NJ: John Wiley and Sons.
3. AIAA, *Guide to verification and Validation of CFD results*. 2001, Reston, Virginia: AIAA. 19.
4. Lorenz, E.N., *Deterministic Non-Periodic Flow*. Journal of the Atmospheric Sciences, 1963. **20**: p. 130-141.
5. Roache, P.J., *Criticisms of the "Correction Factor" Verification Method*. ASME Journal of Fluids Engineering - Technical Briefs, 2002. **125**: p. 732,733.
6. Daley, R., *Atmospheric Data Analysis*. Cambridge atmospheric and space science series 2. 1991, Cambridge, New York: Cambridge University Press.
7. Isaaks, E.H. and R.M. Srivastava, *An Introduction to Applied Geostatistics*. 1st ed. 1989, New York: Oxford University Press.
8. Versteeg, H.K. and W. Malalasekera, *An Introduction to Computational Fluid Dynamics*. 1 ed. 1995, Harlow: Pearson Prentice-Hall.
9. Zienkiewicz, O.C., R.L. Taylor, and J.Z. Zhu, *The Finite Element Method: its basis and fundamentals*. 6th ed. Vol. 1. 2005, Amsterdam: Elsevier Butterworth-Heinemann.
10. Richardson, L.F., *The Deferred Approach to the Limit*. Transactions of the Royal Society, 1927. **A(226)**: p. 229-361.
11. Roache, P.J., *Verification and Validation in Computational Science and Engineering*. 1st ed. 1998, Albuquerque, New Mexico: Hermosa.
12. Demuren, A.O. and R.V. Wilson, *Estimating Uncertainty in Computations of Two-Dimensional Separated Flows*. ASME Journal of Fluids Engineering, 1994. **116**: p. 216-220.
13. Massey, B.S. and A.J. Ward-Smith, *Mechanics of Fluids*. 8 ed. 2006, London: Taylor & Francis.
14. Dieter, G.E., *Engineering Design*. 3rd ed. 2000, Boston: McGraw-Hill.
15. Box, G.E.P. and N.R. Draper, *Empirical Model Building and Response Surfaces*. 1987, New York: Wiley.
16. Roache, P.J., *Fundamentals of Computational Fluid Dynamics*. 1st ed. 1998, Albuquerque, New Mexico: Hermosa.
17. IEEE, *IEEE Standard Glossary of Software Engineering Terminology*. 1991, New York: IEEE.
18. Richardson, L.F., *The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam*. Transactions of the Royal Society, 1910. **A(210)**: p. 307-357.
19. Celik, I. and K. Ozgur, *Numerical Experiments on Application of Richardson Extrapolation With Nonuniform Grids*. ASME Journal of Fluids Engineering, 1997. **119**: p. 584-590.

20. Cadafalch, J., et al., *Verification of Finite Volume Computations on Steady State Fluid Flow and Heat Transfer*. ASME Journal of Fluids Engineering, 2002. **124**: p. 11-21.
21. Roache, P.J., *Perspective: A Method for Uniform Reporting of Grid Refinement Studies*. ASME Journal of Fluids Engineering, 1994. **116**: p. 405-413.
22. Zienkiewicz, O.C., R.L. Taylor, and P. Nithiarasu, *The Finite Element Method for Fluid Dynamics*. 6th ed. Vol. 3. 2005, Oxford: Elsevier/Butterworth Heinemann.
23. Hess, J.L. and A.M.O. Smith, *Calculation of potential flow about arbitrary bodies*. Progress in Aerospace Sciences, 1967. **8**: p. 1-138.
24. LeQuéré, P., *Accurate Solutions to the Square Thermally Driven Cavity at High Rayleigh Number*. Computers and Fluids, 1991. **20**(1): p. 29-41.
25. Kelly, D.W., et al., *A Posteriori Estimates of the Solution Error Caused by Discretization in the Finite Element, Finite Difference and Boundary Element Methods*. International Journal for Numerical Methods in Engineering, 1987. **24**: p. 1921-1939.
26. Naterer, G.F. and J.A. Camberos, *Entropy and the Second Law Fluid Flow and Heat Transfer Simulation*. AIAA Journal of Thermophysics and Heat Transfer, 2003. **17**(3): p. 360-371.
27. Fluent. *FLUENT 6.1 User's Guide*. 2008 29 January 2003 10:50:24 AM [cited 2008 October 10]; Available from: <http://202.185.100.7/homepage/fluent/index.htm>.
28. Strang, G., *Introduction to Linear Algebra*. 3rd ed. 2003, Wellesley, Massachusetts: Wellesley-Cambridge Press.
29. Ruelle, D., ed. *Turbulence, Strange Attractors and Chaos*. World Scientific series on nonlinear science. Vol. 16. 1995, World Scientific: Singapore.
30. Von Neumann, J. and R.D. Richtmyer, *A Method for the Numerical Calculation of Hydrodynamic Shocks*. Journal of Applied Physics, 1950. **21**: p. 232-357.
31. Ivancevic, V.G. and T.T. Ivancevic, *High-Dimensional Chaotic and Attractor Systems: A Comprehensive Introduction*. 2007, Springer: Dordrecht. p. 714.
32. Stern, F. and R.V. Wilson, *Closure to "Discussion of 'Comprehensive Approach to the Verification and Validation of CFD Simulations - Part I: Methodology and Procedures' " (2002, ASME J. Fluids Eng., 124, p.809)*. ASME Journal of Fluids Engineering, 2002. **124**: p. 810-811.
33. Hatton, L., *The T experiments: errors in scientific software*. IEEE Computational Science and Engineering, 1997. **4**(2): p. 27-38.
34. Oberkampf, W.L. and T.G. Trucano, *Verification and Validation in Computational Fluid Dynamics*. Progress in Aerospace Sciences, 2002. **38**: p. 209-272.
35. Oberkampf, W.L. and M.F. Barone, *Measures of Agreement Between Computation and Experiment: Validation Metrics*, in *34th AIAA Fluid Dynamics Conference and Exhibit*. 2004, AIAA: Portland, Oregon.
36. Stern, F., et al., *Comprehensive Approach to Verification and Validation of CFD Simulations - Part I: Methodology and Procedures*. ASME Journal of Fluids Engineering, 2001. **123**: p. 793-802.

37. Wilson, R.V., et al., *Comprehensive Approach to Verification and Validation of CFD Simulations - Part II: Application for RANS Simulation*. ASME Journal of Fluids Engineering, 2001. **123**: p. 803-810.
38. Stern, F., R.V. Wilson, and J. Shao, *Quantitative V&V of CFD simulations and certification of CFD codes with examples*, in *Computational Heat Transfer - '04 3rd International Symposium on Advances in Computational Heat Transfer*. 2004, Begell House, New York: Bergen, Norway.
39. Stern, F., R.V. Wilson, and J. Shao, *Quantitative V&V of CFD simulations and certification of CFD codes*. International Journal for Numerical Methods in Fluids, 2006. **50**: p. 1335-1355.
40. Stern, F., et al., *Statistical Approach for Estimating Intervals of Certification or Biases of Facilities or Measurement Systems Including Uncertainties*. Transactions of the ASME, 2005. **127**: p. 604-610.
41. Coleman, H.W. and W.G. Steele, *Some considerations in the propagation of bias and precision errors into an experimental result*, in *ASME Winter Annual Meeting - Experimental Uncertainty in Fluid Measurement*, E.P. Rood, Editor. 1987, ASME, Fluids Engineering Division: Boston.
42. Hensch, M.J., *Statistical Analysis of Computational Fluid Dynamics Solutions from the Drag Prediction Workshop*. AIAA Journal of Aircraft, 2004. **41**(1): p. 95-103.
43. Oberkampf, W.L., *Discussion: "Comprehensive Approach to the Verification and Validation of CFD Simulations - Part I: Methodology and Procedures" (Stern, F., Wilson, R.V., Coleman, H.W., and Paterson, E.G., 2001, ASME J. Fluids Eng., 123, pp.793-802)*. ASME Journal of Fluids Engineering, 2002. **124**: p. 809-810.
44. Carrica, P.M., R.V. Wilson, and F. Stern, *Unsteady RANS simulation of the ship forward speed diffraction problem*. Computers and Fluids, 2006. **35**: p. 545-570.
45. Pelletier, D., et al., *Adaptivity, Sensitivity, and Uncertainty: Toward Standards of Good Practice in Computational Fluid Dynamics*. AIAA Journal, 2003. **41**(10): p. 1925-1933.
46. Jaques, J., C. Lavergne, and N. Devictor, *Sensitivity Analysis in presence of model uncertainty and correlated inputs*, in *The 4th International Conference on Sensitivity Analysis of Model Output*. 2004, Los Alamos National Laboratory: Santa Fe, New Mexico.
47. Saltelli, A., S. Tarantola, and F. Campolongo, *Sensitivity Analysis in Practice: a guide to assessing scientific models*. 2004, Hoboken, NJ: Wiley.
48. DeLoach, R., *Applications of Modern Experiment Design to Wind Tunnel Testing at NASA Langley Research Center*, in *36th Aerospace Sciences Meeting and Exhibit*. 1998, AIAA: Reno, Nevada.
49. Dowding, K.J. and B.F. Blackwell, *Sensitivity Analysis for Non-linear Heat Conduction*. ASME Journal of Heat Transfer, 2001. **123**(1): p. 1-10.
50. Gu, Y.X., et al., *A Sensitivity Analysis Method for Linear and Nonlinear Transient Heat Conduction with Precise Time Integration*. Structural Multidisciplinary Optimization, 2002. **24**: p. 23-37.
51. Baysal, O. and M.E. Eleshaky, *Aerodynamic Design Optimization Using Sensitivity Analysis and Computational Fluid Dynamics*. AIAA Journal, 1992. **30**(3): p. 718-725.

52. Pironneau, O., *On optimum design in fluid mechanics*. Journal of Fluid Mechanics, 1974. **64**(1): p. 97-110.
53. Baysal, O. and K. Ghayour, *Continuous Adjoint Sensitivities for Optimization with General Cost Functionals on Unstructured Meshes*. AIAA Journal, 2001. **39**(1): p. 48-55.
54. Brezillon, J. and N.R. Gauger, *2D and 3D aerodynamic shape optimisation using the adjoint approach*. Aerospace Science and Technology, 2004. **8**: p. 715-727.
55. Kim, C.S., C. Kim, and O.H. Rho, *Sensitivity Analysis for the Navier-Stokes Equations with Two-Equation Turbulence Models*. AIAA Journal, 2001. **39**(5): p. 838-845.
56. Lu, S.-Y. and P. Sagaut, *Direct sensitivity analysis for smooth unsteady compressible flows using complex differentiation*. International Journal for Numerical Methods in Fluids, 2007. **53**: p. 1863-1886.
57. Anderson, W.K., et al., *Sensitivity Analysis for Navier-Stokes Equations on Unstructured Meshes Using Complex Variables*. AIAA Journal, 2001. **39**(1): p. 56-63.
58. Burg, C.O.E. and J.C. Newman, *Computationally efficient, numerically exact design space derivatives via the complex Taylor's series expansion method*. Computers and Fluids, 2003. **32**: p. 373-383.
59. Mader, C.A. and J.R.R.A. Martins, *ADJoint: An Approach for the Rapid Development of Discrete Adjoint Solvers*. AIAA Journal, 2008. **46**(4): p. 863-873.
60. Bischof, C., et al., *Automatic differentiation of advanced CFD codes for multidisciplinary design*. Computing Systems in Engineering, 1992. **3**(6): p. 625-637.
61. Michalek, T., *A method for the verification and validation of thermal and viscous flows*, in *Department of Mechanics and Physics of Fluids*. 2006, Polish Academy of Sciences: Krakow.
62. Devore, J. and N.R. Farnum, *Applied Statistics for Scientists and Engineers*. 2nd ed. 2004, London: Duxbury Press.
63. Draper, N.R. and H. Smith, *Applied Regression Analysis*. 3rd ed. Wiley Series in Probability and Statistics. 1998, New York: Wiley.
64. Fisher, R.A., *The Design of Experiments*. 8 ed. 1966, Edinburgh: Oliver and Boyd.
65. Marquardt, D., *An Algorithm for Least Squares Estimation of Nonlinear Parameters*. SIAM Journal of Applied Mathematics, 1963. **11**: p. 431-441.
66. Fletcher, R., *Practical Methods of Optimization*. 2nd ed. 1987, New York: John Wiley & Sons.
67. Sadler, D.R., *Numerical Methods for Non-linear Regression*. 1975, St Lucia, Queensland: University of Queensland Press.
68. Shepard, D., *A two-dimensional interpolation function for irregularly spaced data*, in *23rd ACM National Conference*. 1968, Association for Computing Machinery.
69. Dahlquist, G. and A. Björk, *Numerical Methods*. 1974, Englewood Cliffs, NJ: Prentice-Hall.
70. Runge, C., *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*. Zeitschrift für Mathematik und Physik, 1901. **46**: p. 224-243.

71. Bengtsson, B.E. and S. Nordbeck, *Construction of isarithms and isarithmic maps by computers*. BIT Numerical Mathematics, 1964. **4**: p. 87-105.
72. Wahba, G., *Spline Models for Observational Data*. 1990, Philadelphia: SIAM.
73. Lancaster, P. and K. Salkauskas, *Surfaces Generated by Moving Least Squares Methods*. Mathematics of Computation, 1981. **37**(155): p. 141-158.
74. Piegl, L.A. and W. Tiller, *The NURBS book*. 2nd ed. 1997, New York: Springer.
75. de Boor, C., *A practical guide to splines*. 1978, New York: Springer-Verlag.
76. Liszka, T., *An interpolation method for an irregular net of nodes*. International Journal for Numerical Methods in Engineering, 1984. **20**(9): p. 1599-1612.
77. Coleman, H.W. and W.G. Steele, *Experimentation and Uncertainty Analysis for Engineers*. 2nd ed. 1999, New York: Wiley.
78. DeLoach, R., *Improved quality in aerospace testing through the modern design of experiments*. AIAA Journal, 2000.
79. Lo, C.F., J.L. Zhao, and R. DeLoach, *Application of Neural Networks to Wind Tunnel Data Response Surface Methods*, in *21st AIAA Aerodynamic Measurement Technology and Ground Testing Conference*. 2000, AIAA: Denver, Colorado.
80. Krige, D.G., *A statistical approach to some basic mine valuation problems on the Witwatersrand*. Journal of the Chemical, Metallurgical and Mining Society of South Africa, 1951. **52**: p. 119-138.
81. Krige, D.G., *Two-dimensional weighted moving average trend surfaces for ore evaluation*. Journal of South African Institute of Mining and Metallurgy, 1966. **Special edition** (Proceedings of Symposium on Mathematical Statistics and Computer Applications in Ore Valuation): p. 13-38.
82. Matheron, G.F., *La théorie des variables régionalisées et ses applications*, in *Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau*. 1970, Ecole Supérieure des Mines de Paris: Paris.
83. Matheron, G.F., *Estimer et choisir*, in *Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau*. 1978, Ecole Supérieure des Mines de Paris: Paris.
84. Chilès, J.-P. and P. Delfiner, *Geostatistics Modeling Spatial Uncertainty*. Wiley Series in Probability and Statistics. 1999, New York: Wiley-Interscience.
85. Myers, J.C., *Geostatistical Error Management: Quantifying uncertainty for environmental sampling and mapping*. 1997, New York: Van Nostrand Reinhold.
86. Wackernagel, H., *Multivariate Geostatistics: An Introduction with Applications*. 2003, Berlin: Springer-Verlag.
87. Guarascio, M. *Improving the uranium deposits estimations (the Novazza case)*. in *Advanced Geostatistics in the Mining Industry, NATO Advanced Study Institute*. 1975. University of Rome, Italy: D. Reidel, Dordrecht Holland.
88. Chung, H.-S. and J.J. Alonso, *Using Gradients to Construct Cokriging Approximation Models for High Dimensional Design Optimization Problems*, in *40th AIAA Aerospace Science Meeting and Exhibit*. 2002, AIAA: Reno, NV.

89. Laurenceau, J. and P. Sagaut, *Building Efficient Response Surfaces of Aerodynamic Functions with Kriging and Cokriging*. AIAA Journal, 2008. **46**(2): p. 498-507.
90. Martin, J.D. and T.W. Simpson, *Use of Kriging Models to Approximate Deterministic Computer Models*. AIAA Journal, 2005. **43**(4): p. 853-863.
91. Amtec, *Tecplot 360 User's Manual*. 2008, Bellevue, WA: Tecplot Inc.
92. Jouhaud, J.C., P. Sagaut, and B. Labeyrie, *A Kriging Approach for CFD/Wind-Tunnel Data Comparison*. ASME Journal of Fluids Engineering, 2006. **128**: p. 847-855.
93. Sacks, J., et al., *Design and Analysis of Computer Experiments*. Statistical Science, 1989. **4**(4): p. 409-423.
94. Creutin, J.G., G. Delrieu, and T. Lebel, *Rain Measurement by Rainage-Radar Combination: A Geostatistical Approach*. Journal of Atmospheric and Oceanic Technology, 1988. **5**(1): p. 102-115.
95. Kyriakidis, P.C., J. Kim, and N.L. Miller, *Geostatistical Mapping of Precipitation from Rain-Gauge Data Using Atmospheric and Terrain Characteristics*. Journal of Applied Meteorology, 2001. **40**: p. 1855-1877.
96. Basili, P., et al., *Retrieving Temperature Profiles by Microwave Radiometry Using a Priori Information on Atmospheric Spatial-Temporal Evolution*. IEEE Transactions on Geoscience and Remote Sensing, 2001. **39**(9): p. 1896-1905.
97. Delhomme, J.P., *Kriging in the hydrosociences*. Advances in Water Resources, 1978. **1**(5): p. 251-266.
98. Wen, X.H., et al., *A program to create permeability fields that honor single-phase flow rate and pressure data*. Computers and Geosciences, 1999. **25**: p. 217-230.
99. Chilès, J.-P. *How to adapt kriging to non-classical problems: Three case studies*. in *Advanced Geostatistics in the Mining Industry*, NATO Advanced Study Institute. 1975. University of Rome, Italy: D. Reidel Dordrecht Holland.
100. Sobczyk, K., *Stochastic Differential Equations: With Applications to Physics and Engineering*. 1991, Dordrecht, Netherlands: Kluwer.
101. Gu, L., *Moving kriging interpolation and element-free Galerkin method*. International Journal for Numerical Methods in Engineering, 2003. **56**(1): p. 1-11.
102. Sayakoummane, V. and W. Kanok-Nukulchai, *A Meshless Analysis of Shells Based on Moving Kriging Interpolation*. International Journal of Computational Methods, 2007. **4**(4): p. 543-565.
103. Mardia, K.V., et al., *Kriging and splines with derivative information*. Biometrika, 1996. **83**(1): p. 207-221.
104. Cressie, N., *The Origins of Kriging*. Mathematical Geology, 1990. **22**(3): p. 239-252.
105. Gandin, L.S., *Ob"ektivnyi analiz meteorologicheskikh polei*. 1963, Leningrad: Gidrometeorologicheskoe Izdatel'stvo.
106. Kalman, R.E., *A New Approach to Linear Filtering and Prediction Problems*. ASME Journal of Basic Engineering, 1960. **82**(1): p. 35-45.
107. Congdon, P., *Applied Bayesian modelling*. Wiley series in probability and statistics. 2003, New York: Wiley.

108. Chassignet, E.P. and J. Verron, eds. *Ocean weather forecasting: an integrated view of oceanography*. 2006, Springer: Dordrecht.
109. Cao, Y., et al., *A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition*. *International Journal for Numerical Methods in Fluids*, 2007. **53**: p. pp. 1571-1583.
110. Cressie, N., *Statistics for Spatial Data*. Revised ed. 1993, New York: John Wiley & Sons.
111. Galoul, V. and T.J. Barber, *A Study of an Inverted Wing with Endplates in Ground Effect*, in *16th Australasian Fluid Mechanics Conference*, P. Jacobs, Editor. 2007, School of Engineering, Univ. Queensland: Gold Coast, Australia. p. 919-924.
112. De Vahl Davis, G. and I.P. Jones, *Natural convection in a square cavity: a comparison exercise*. *International Journal for Numerical Methods in Fluids*, 1983. **3**(3): p. 249-264.
113. Diasinos, S., et al., *A Two-Dimensional Analysis of the Effect of a Rotating Cylinder on an Inverted Aerofoil in Ground Effect*, in *The 15th Australasian Fluid Mechanics Conference*. 2004, University of Sydney: University of Sydney.
114. Diasinos, S., et al. *Validation of a 2D CFD Model for the Implementation of a Moving Ground in the UNSW 3×4 ft Wind Tunnel*. in *The 5th Pacific Symposium on Flow Visualization and Image Processing*. 2005: University of New South Wales.
115. Diasinos, S., et al. *An Experimental and Numerical Two-dimensional Analysis of the Effect of a Rotating Cylinder on an Inverted Aerofoil in Ground Effect*. in *The 5th Pacific Symposium on Flow Visualization and Image Processing*. 2005: University of New South Wales.
116. Diasinos, S. and A. Gatto, *Experimental investigation into wing span and angle-of-attack effects on sub-scale race car wing/wheel interaction aerodynamics*. *Experiments in Fluids*, 2008. **45**: p. 537-546.
117. Benedict, L. and R.D. Gould, *Towards better uncertainty estimates for turbulence statistics*. *Experiments in Fluids*, 1996. **22**(2): p. 129-136.
118. Matheron, G.F., *Les variables régionalisées et leur estimation. Une application de la théorie des fonctions aléatoires aux Sciences de la Nature*. 1965, Paris: Masson.
119. Michalek, T. and T.A. Kowalewski, *Natural Convection for Anomalous Density Variation of Water: Numerical Benchmark*. *Progress in Computational Fluid Dynamics*, 2005. **5**(3/4/5): p. 158-170.
120. Yao, T. and A.G. Journel, *Automatic Modeling of (Cross) Covariance Tables Using Fast Fourier Transform*. *Mathematical Geology*, 1998. **30**(6): p. 589-615.
121. Lloyd, S.P., *Least Squares Quantization in PCM*. *IEEE Transactions on Information Theory*, 1982. **28**(2): p. 129-137.

# Appendix A - Demonstration:

## MATLAB listing

In Chapter IV, a one-dimensional example of kriging estimation was offered, based on a time-series of velocity measurements taken by an LDA in the trailing vortex of a wing. These measurements were taken by Vincent Galoul whilst at UNSW for his Masters practicum. As a guide to this explanatory example and the basic algorithm, we have supplied part of the the MATLAB code used to produce Figures III—8 and III—9.

```
vinces_timeseries = ...
'C:\Documents and Settings\z2250722\vince\LDA_DATA_2.txt';
fidv = fopen(vinces_timeseries);
fscanf(fidv,'TIME VELOCITY');
[TVINCE] = fscanf(fidv,'%g %g', [2 Inf]);
fclose(fidv);
TVINCE = TVINCE';
TVINCE = sortrows(TVINCE);
% reads raw data into TVINCE
COVINCE = covariogen(TVINCE(:,1), TVINCE(:,2));
% generates spatial statistics

% figure III-8
% various covariance function fits to the above spatial statistics

figure;
h6 = gca;
nsamp = size(COVINCE);
plot(h6,COVINCE(2:nsamp(1),1),COVINCE(2:nsamp(1),2),'ok')
hold on
plot(h6,COVINCE(1,1),COVINCE(1,2),'ok','MarkerFaceColor','k')
% best fit to covariance with nugget
VMOD(:, :, 1) = [ 0.033666  0.01  0.0072  0.025  0.0 0.0;
                0.0      0.0005 0.011  0.0038  0.0 0.0;
                0.0      0.0    0.0    1461.2  0.0 0.0;];
H = 0:(COVINCE(nsamp(1),1)/500):(1.05*COVINCE(nsamp(1),1));
```

```

C = covariograms(VMOD,H,'R');
plot(h6,H,C,'k','LineWidth',2)
plot(h6,H(1),C(1),'ok')
% best fit to covariance with zero nugget effect
VMOD(:,:,1) = [ 0.0  0.043666  0.0072  0.025  0.0  0.0;
                0.0  0.0005  0.011  0.0038  0.0  0.0;
                0.0  0.0  0.0  1461.2  0.0  0.0;];
C = covariograms(VMOD,H,'R');
plot(h6,H,C,'--b','LineWidth',1)
% only Gaussian model
VMOD(:,:,1) = [ 0.0 0.0 0.0 0.0 0.0 0.075867;
                0.0 0.0 0.0 0.0 0.0 0.0025 ;
                0.0 0.0 0.0 0.0 0.0 0.0 ;];
C = covariograms(VMOD,H,'R');
plot(h6,H,C,'r','LineWidth',1)
hold off

% figure IV-9
% a close-up of kriged interpolation

figure;
h8 = gca;
t_est = ((TVINCE(10000)-0.00005):0.000005:(TVINCE(10030)+0.00005))';
% only Gaussian model
VMOD(:,:,1) = [ 0.0 0.0 0.0 0.0 0.0 0.075867;
                0.0 0.0 0.0 0.0 0.0 0.0025 ;
                0.0 0.0 0.0 0.0 0.0 0.0 ;];
INTRPOL = krige(vinces_timeseries,t_est,VMOD);
axes(h8);
plot(h8,TVINCE(10000:10031,1),TVINCE(10000:10031,2),'ok')
hold on;
plot(h8,INTRPOL(:,1),INTRPOL(:,2),'r','LineWidth',1.5)
% best fit to covariance with zero nugget effect
VMOD(:,:,1) = [ 0.0  0.043666  0.0072  0.025  0.0 0.0;
                0.0  0.0005  0.011  0.0038  0.0 0.0;
                0.0  0.0  0.0  1461.2  0.0 0.0;];
INTRPOL = krige(vinces_timeseries,t_est,VMOD);
plot(h8,INTRPOL(:,1),INTRPOL(:,2),'--b','LineWidth',1.5)
% best fit to covariance with nugget
VMOD(:,:,1) = [ 0.033666  0.01  0.0072  0.025  0.0 0.0;
                0.0  0.0005  0.011  0.0038  0.0 0.0;
                0.0  0.0  0.0  1461.2  0.0 0.0;];
INTRPOL = krige(vinces_timeseries,t_est,VMOD);
plot(h8,INTRPOL(:,1),INTRPOL(:,2),'k','LineWidth',1.5)
hold off;

```

This code fragment requires the functions covariogen, covariograms, krige and fitpts listed below.

```

function COVOUT = covariogen(T,V)
% generates covariance function spatial statistics using
% Equation (3.8)
nres = 20;
band = 250;
vmean = mean(V);
npt = size(T);
k = 0;
space = band*npt(1);
SAMPL = NaN([space 2]);
for ia=2:npt
    nst = max(1,(ia-band));
    for ib=nst:(ia-1)
        k = k + 1;
        SAMPL(k,1) = T(ia) - T(ib);
        tmp = V(ia) - vmean;
        tmp = tmp*(V(ib) - vmean);
        SAMPL(k,2) = tmp;
    end
end
% generates lags and
% summation terms in covariance functions
SAMPL((k+1):space,:) = [];
SAMPL = sortrows(SAMPL);
sport = floor(k/nres);
start = 1;
finish = sport;
VARI = [];
for ia=1:nres
    VARI(ia,:) = mean(SAMPL(start:finish,:));
    VARI(ia,:) = VARI(ia,:)*0.5;
    start = finish + 1;
    finish = finish + sport;
end
% splits lags into groups and calculates covariance
COVOUT = [ 0, var(V); VARI; ];

function [GAMMA]=covariograms(MODEL,T,side)

```

```

% calculates covariance function at lags T on the basis of MODEL
% function for Equation (3.25)
% side is a character 'L' or 'R' to include
% or filter the nugget effect
% note function is vectorised

% MODEL format
% each matrix in MODEL represents
% the variogram models in the columns as follows
% column 1 : Nugget effect MODEL(1,1,1) only
% column 2 : Spherical model; sill, range
% column 3 : Gaussian model; sill, range
% column 4 : Hole-effect model; sill, range, omega
% column 5 : Exponential model; sill, range
% column 6 : Cubic model; sill, range

TOT = zeros(size(T));
T = abs(T);
% nugget effect model (noise)
if (MODEL(1,1,1)~=0.0)&&(side=='L')
    mask = find(T==0.0);
    TOT(mask) = TOT(mask) + MODEL(1,1,1);
End

% loops through all models, evaluates and sums them
ia = 1;
while norm(MODEL(1,:,ia))~=0.0
% spherical model
    if MODEL(1,2,ia)~=0.0;
        VARG = zeros(size(T));
        mask = find(T<MODEL(2,2,ia));
        TMP = T/MODEL(2,2,ia);
        VARG(mask) = 1.0 - 1.5*TMP(mask)...
        + 0.5*(TMP(mask).*TMP(mask).*TMP(mask));
        VARG = VARG*MODEL(1,2,ia);
        TOT = TOT + VARG;
    end
% gaussian model
    if MODEL(1,3,ia)~=0.0
        TMP = T/MODEL(2,3,ia);
        VARG = exp(-3*(TMP.*TMP));
        VARG = VARG*MODEL(1,3,ia);
        TOT = TOT + VARG;
    end
% hole-effect model

```

```

    if MODEL(1,4,ia)~=0.0
        TMP = T./MODEL(2,4,ia);
        VARG = (exp(-3*TMP)).*(cos(MODEL(3,4,ia)*T));
        VARG = VARG*MODEL(1,4,ia);
        TOT = TOT + VARG;
    end
% exponential model
    if MODEL(1,5,ia)~=0.0
        TMP = T/MODEL(2,5,ia);
        VARG = exp(-3*TMP);
        VARG = VARG*MODEL(1,5,ia);
        TOT = TOT + VARG;
    end
% cubic model
    if MODEL(1,6,ia)~=0.0
        VARG = zeros(size(T));
        mask = find(T<MODEL(2,6,ia));
        TMP = T/MODEL(2,6,ia);
        VARG(mask) = 1 - 7.0*TMP(mask).*TMP(mask)...
            +8.75*TMP(mask).*TMP(mask).*TMP(mask)...
            - 3.5*(TMP(mask).^5) + 0.75*(TMP(mask)).^7;
        VARG = VARG*MODEL(1,6,ia);
        TOT = TOT + VARG;
    end
    ia = ia + 1;
end
GAMMA = TOT;

function [KROUT] = krige(filein,INT,VARMODL)
% performs the 1D kriging estimation, with function fitpts
% uses the points in filein to interpolate from
% uses VARMODL for the covariance function
% interpolates at points INT
fida = fopen(filein,'rt');
fidb = fopen('krigout.txt','wt');
% reads data into V, T vectors
fscanf(fida,'TIME VELOCITY');
[TV, npt] = fscanf(fidb,'%g %g', [2 Inf]);
TV = TV';
TV = sortrows(TV);
T = TV(:,1);
V = TV(:,2);
VMODEL = zeros(3,6,8);
nmods = size(VARMODL);
ntmp = size(nmods);

```

```

if ntmp(2)==2
    nummods = 1;
else
    nummods = nmods(3);
end
% in case VARMODL is a singleton dimension in the last index
for ia=1:nummods
    VMODEL(:, :, ia) = VARMODL(:, :, ia);
end
[WORM, KRVAR] = fitpts(T, V, INT, VMODEL);
TV = [INT WORM KRVAR];
KROUT = TV;
% outputs interpolation locations, interpolations and kriging variance
% at these locations
fclose(fida);
fclose(fidb);

function [VOUT, VAROUT]=fitpts(T, V, TIN, MODEL)
% interpolates from known data in T and V
% interpolates at points in TIN
% uses MODEL for the covariance function
window = 20;
prox = 30;
% interpolates from nearest thirty data-points
ntmp = size(T);
ndat = ntmp(1);
ntmp = size(TIN);
npts = ntmp(1);
TIN = sort(TIN);
vari = sum(MODEL(1, :));
% total variance vari
VOUT = zeros([npts 1]);
VAROUT = zeros([npts 1]);
nat = 1;
kat = 1;
% nat indexes T and V
% kat indexes TIN
while (nat<=ndat)&&(kat<=npts)
% bracket moves through T and V, 'window' points at a time
% also includes 'prox' points to either side
    next = nat+window-1;
    if (next>ndat), next=ndat; end
    n1 = nat-prox;
    if (n1<1), n1=nat; end
    n2 = next+prox;

```

```

if (n2>ndat), n2=ndat; end
PART = T(n1:n2);
VELOC = [V(n1:n2) ; 0.0];
nsolv = 0;
D = [];
cat = kat;
while TIN(kat)<T(next)
    TMP = PART-TIN(kat);
    VAR = covariograms(MODEL,TMP,'R');
    VAR = vari - VAR;
    VAR = [VAR;1];
    D = [D,VAR];
    kat = kat + 1;
    nsolv = nsolv + 1;
    if kat>npts, break, end
end
% constructs RHS (Equation (3.21))
% for all points TIN in the central bracket
if nsolv~=0
    eqns = n2-n1+1;
    COV = ones(eqns+1);
    TMP = [];
    for ic=1:eqns
        TMP = [TMP,(PART(ic)-PART)];
    end
    VAR = covariograms(MODEL,TMP,'L');
    VAR = vari - VAR;
    COV(1:eqns,1:eqns) = VAR;
    COV(eqns+1,eqns+1) = 0.0;
% constructs LHS (Equation (3.21)) covariance matrix from the points T
% occurring in the central bracket ('window' points long)
% and also the 'prox' points to either side
    W = COV\D;
    VOUT(cat:(kat-1)) = W'*VELOC;
    kc = 0;
    for ic=cat:(kat-1)
        kc = kc+1;
        VAROUT(ic) = W(:,kc)'*D(:,kc);
    end
% solves for estimates (Equation (3.14))
% and kriging variance (Equation (3.16), properly (4.58))
    end
    nat = next + 1;
end
% performs same procedure for the tail-end points not covered so far

```

```

TMP = [];
for ia=kat:npts
    TMP = [TMP, (PART-TIN(ia))];
end
if kat<=npts
    D = ones( (eqns+1), npts-kat+1 );
    D(1:eqns,:) = covariograms(MODEL,TMP,'R');
    D(1:eqns,:) = vari - D(1:eqns,:);
    W = COV\D;
    VOUT(kat:npts) = W'*VELOC;
    kc = 0;
    for ic=kat:npts
        kc = kc+1;
        VAROUT(ic) = W(:,kc)'*D(:,kc);
    end
end
end

```

The above MATLAB code is an abridged form of the original code used to produce Figures III—8 and III—9. This code and that used to produce all MATLAB figures in this text is included on the CD-R at the back of this thesis in the directory \code\matlab\.... The foregoing MATLAB code is not the principal code developed in this thesis. This code was written in Fortran90 and its operation is described in the next Appendix.

# Appendix B - Code Functionality

The greater part of this thesis was spent developing a Fortran90 code to perform basic multivariate structure identification and estimation. The code itself is not listed in print but is found together with makefiles and a Microsoft Visual Studio project on the CD-R at the back of this thesis, in the directory `\code\fortran\...`. The visual studio project actually consists of two workspaces for two separate command-line executables; `datagen.exe` and `krige.exe`. These two programs use common modules but have their own main program blocks and perform different tasks. The basic functionality of the programs is described below in terms of external inputs and outputs – the code itself is documented in Appendix E.

## B - 1. Program `krige`

The program `krige.exe` forms the main tool for data comparison and assimilation. The raw data to be compared is input as two complementary files with names differing only in the extension; `*.kr1` and `*.kr2`. These files are resident in the same directory as the executable file `krige.exe`, and they contain sequentially enumerated datasets – that is, sets of spatial data comprising nodal coordinates and corresponding field values. The preparation of these direct access binary files effected by `datagen.exe`, whose operation is described in the next section. It suffices for the moment to note that these files contain all the raw spatial data, and relevant associated information like the size of the sets and how they have been split up. The only other file input to the program is a simple sequential text file called `krigin.txt`, which specifies what spatial models and variables are to be used. This file contains keywords followed by values, that specify various options in the estimation. It does not matter in which order the options appear, and often if the option is not specified a default option is set instead – unless of course, the input is absolutely necessary in which case the program terminates with an error.

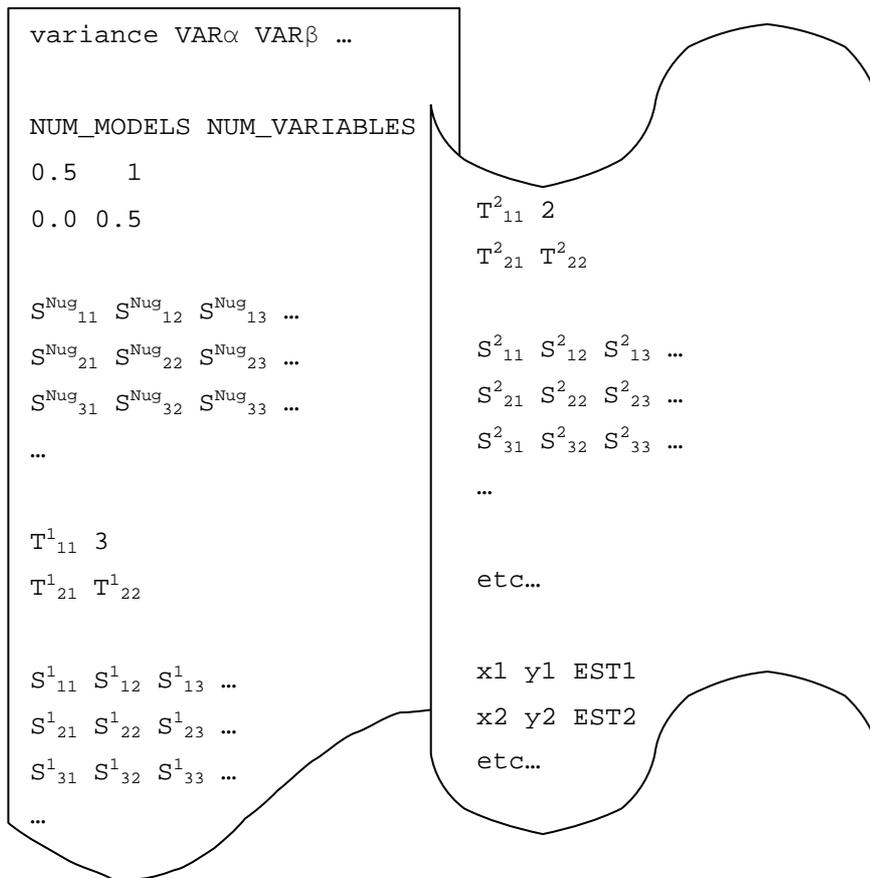
EST1	EST2	EST3	EST4	EST5
EST6	EST7	EST8	EST9	EST10
EST11	EST12	EST13	EST14	EST15
EST16	EST17	EST18	EST19	EST20
VAR1	VAR2	VAR3	VAR4	VAR5
VAR6	VAR7	VAR8	VAR9	VAR10
VAR11	VAR12	VAR13	VAR14	VAR51
VAR16	VAR17	VAR18	VAR19	VAR20
???	???	???	???	???
???	???	???	???	???
???	???	???	???	???
???	???	???	???	???

**Figure B—1: Format for krigout.txt when operating in raster interpolation mode.**

At run-time, the program first prompts the user for the ‘database’ to be used – i.e. ‘smith’ where ‘smith.kr1’ and ‘smith.kr2’ are the raw data binaries, and then the mode of operation. Note that if either raw data file is missing, the program terminates with an error. The second prompt is for the mode of operation. As mentioned in Section IV - 5.1, the three modes of operation are; “raster interpolation”, “structure identification” and “cross-validation”. Entering either of these three phrases (or the keywords `raster` , `continuity` , or `cross` respectively) when prompted sets the program about the tasks described in the next sections.

### **B - 1.1. Raster interpolation**

In this mode the program interpolates the raw data on the regular grid defined following the keyword `raster` in `krigin.txt` . The quantity that is estimated relates to the first dataset specified after the keyword `kriging variables` , and as mentioned in Section IV - 6 the estimation is one of three types; unfiltered, noise filtered or drift estimation – as specified by the keyword `estimation` . The raster of estimates so generated is returned in the text file `krigout.txt` followed by a second



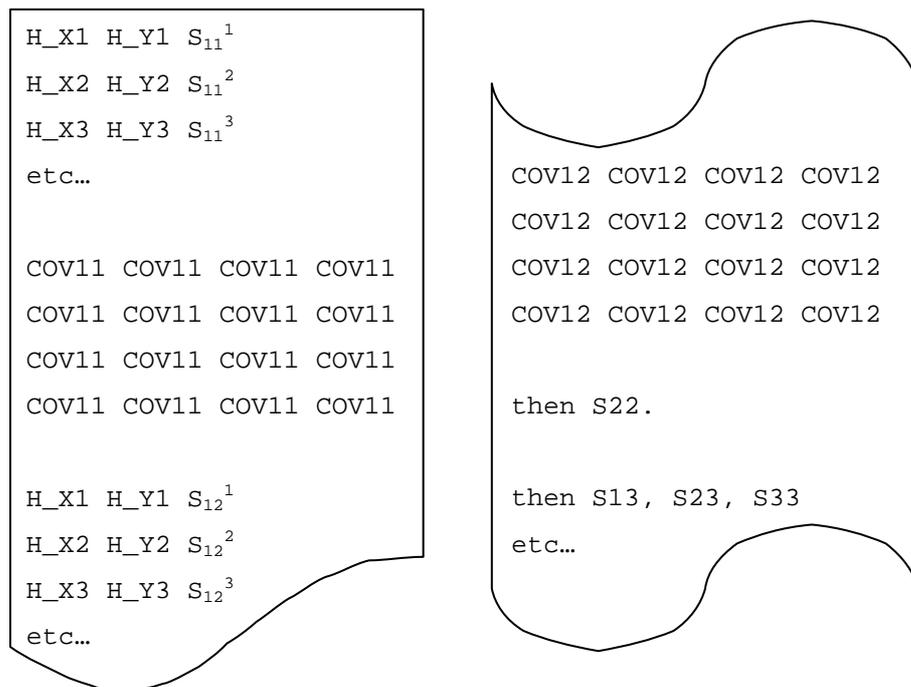
**Figure B—2: Format for krigdia.txt .**

raster of numbers that correspond to the kriging variance. A third raster of numbers follows this – often used for diagnostic purposes or for other special outputs.

Several other output files are produced, all of which are also sequential formatted (ordinary ASCII) text files. In `krigdia.txt`, the model for covariance used for the estimations is summarised along with a listing of coordinates of interpolated points. The first line of this text file reports the total variance for each of the datasets listed after the keyword `kriging variables`. The second line is blank, and the third reports the number of basic models used (those discovered in Section IV - 5.1). What then follows is a listing of the models (anisotropy transform coefficients  $T^i$  and type  $\gamma^j$ ) and their respective contributions to the linear model of coregionalisation  $s^i$ . Then there follows a list of points at which estimates are produced. The basic format is illustrated in Figure B—2. Note that only one half of the transform matrices is represented as they are symmetric, and note also that the item in the first row, second column is the type of model used where; 1 indicates the

spherical model, 2 indicates the exponential model, and 3 indicates the Gaussian model. There is one exception to this – the first model is *always* the nugget effect. The model entries here are spurious and should be ignored – the nugget effect can have no anisotropy, these values are merely default initialisation values that are never used. The only values that are important are the coefficients that follow,  $s^{nugget}$ .

The file `krigrasta.txt` contains a listing of the lag vectors and spatial statistics introduced in Sections IV - 2 and IV - 3 that are generated in the course of structure identification by the splitting procedure outlined in Section IV - 4. These lists are arranged according to the order of the datasets specified after the keyword `kriging variables`. For a sequence of datasets represented by “  $\beta \chi \delta$ ” there will be  $4(4+1)/2$  lists of spatial statistics. These correspond to the spatial statistics  $S^k$ ,  $S^k$ ,  $S^k$ ,  $S^k$ ,  $S^k$ ,  $S^k$ ,  $S^k_{\Delta}$ ,  $S^k_{\Delta}$ ,  $S^k_{\Delta}$  and  $S^k_{\Delta\Delta}$  respectively; different length sequences are analogous although there is a current maximum of four cokriging variables. If there are two interpolating dimensions, these lists are followed by rasters representing the corresponding modelled covariance functions that have been fitted to them. These rasters are produced on a regular 20×20 grid, stretched to cover the maximum extents of the spatial statistics in the foregoing lists. Another schematic of



**Figure B—3: Format for krigrasta.txt**

the format is offered in B—3.

There is another file `krigdr.txt` which is to be left flexible in behaviour. Mostly it is used to stream diagnostics or any output of special interest.

## B - 1.2. Structure identification

In structure identification mode, program `krige` does not make estimates, but instead performs structure identification individually on the datasets listed after the keyword `kriging variables` in `krigin.txt`. No cross-covariant statistics are calculated nor is a linear coregionalisation model fitted, but the algorithms relating to Sections IV - 2, IV - 3, IV - 4 and IV - 5 are executed. Notably, the total variance of the data does not play a part, as no coregionalisation is required and the auto-covariant structures are scaled to one.

The resulting individual fits are output to `krigdia.txt`, which is slightly modified from the format in Figure B—2 in that there is no variance line, and obviously no list of interpolated points. Just the summary of the structures is reported, for which the matrices  $\mathbf{S}^i$  are actually a single scalar value,  $s^i$ .

```

Auto-covariant structures: type X
H_X1 H_Y1 SXX1
H_X2 H_Y2 SXX2
H_X3 H_Y3 SXX3
etc...

X_average X_TotalVariance
0.5 1
0.0 0.5

SNug11 SNug12 SNug13 ...
SNug21 SNug22 SNug23 ...
SNug31 SNug32 SNug33 ...
...

T111 3
T121 T122

S111 S112 S113 ...
S121 S122 S123 ...
S131 S132 S133 ...
...
etc.

Auto-covariant structures: type Y
etc.

```

Figure B—4: Format for `krigdr.txt` (structure identification).

Furthermore similar summaries are reported for all the datasets indicated by `kriging variables`, in the same order as they appear in `krigin.txt`. Likewise, the spatial statistics are output to `krigrasta.txt` as before, but now instead of cycling through auto and cross-covariant structures, lists of spatial statistics are again reported for the datasets indicated by the keyword `kriging variables`. The file `krigout.txt` is not written to when `krige` operates in this mode.

Particular use is made however, of the file `krigrdr.txt`. For each dataset indicated in `krigin.txt`, there will appear in this file information in the format indicated in Figure B—4. This comprises a header line indicating the dataset that is under examination, a list of *raw* spatial statistics (all other output lists of spatial statistics have outliers removed by the subroutine `inlier`), and finally the individual fitted covariance model in much the same format as used before.

### B - 1.3. Cross-validation

This run-mode performs a cross-validation or “leave-one-out” exercise on the data specified in the primary dataset. All other aspects of operation are largely the same, as essentially the same steps must be followed – the output text files `krigrasta.txt` and `krigrdia.txt` are produced in the same manner as for raster interpolation, and the steps and options to generate a covariance model are also identical. The only difference in the output text files is in `krigout.txt` which now lists the coordinates of the primary input data followed by three other values, discussed below. Additionally, two binary files are created `*.kv1` and `*.kv2` which essentially contain a binary copy of the same data written to the formatted file `krigout.txt`.

```
X_1 Y_1 P'_1 σ'_1 (P_1 - P'_1)/σ'_1  
X_2 Y_2 P'_2 σ'_2 (P_2 - P'_2)/σ'_2  
X_3 Y_3 P'_3 σ'_3 (P_3 - P'_3)/σ'_3  
...  
etc.
```

**Figure B—5: Format for `krigout.txt` when operating in cross-validation mode.**

Cross-validation is a way of checking the performance of an estimation algorithm. Essentially, each known datum (or node) is estimated in turn as if one were unaware of its existence. The estimated value is then compared (cross-validated) with the known value. If this is done for all the known values, the performance of the estimation algorithm may then be assessed in terms of the discrepancy at each known point. Cross-validations are performed with reference to any particular set of estimation options. The program then estimates a value at each data point in the primary dataset, using the full set of data. This value, say  $p$  is then compared to an estimate  $p'$  made at the same location, but now with the particular data point missing. What is reported in `krigout.txt` is the new value  $p'$ , the kriging variance of the new estimate  $\hat{\sigma}_k^2$  and the normalised difference  $(p - p') / \hat{\sigma}_k$  – as illustrated in Figure B—5. Further to this, the value  $(p - p') / \hat{\sigma}_k$  forms the basis of a new set of binary files `*.kv1` and `*.kv2`. Apart from the different data values, these files mirror the original data files. These files may be renamed with `*.kr1` and `*.kr2` extensions, and the normalised residuals of cross-validation can be examined as a spatial dataset in its own right.

#### **B - 1.4. Input keywords**

The following is a list of keywords in `krigin.txt` that may affect the operation of `krige`, irrespective of the run-mode. Whenever, in whichever run-mode, the program encounters an algorithm that is sensitive to one of these keywords it checks the input file `krigin.txt` to see what to do. Behaviour with respect to keywords that have a different interpretation in particular run-modes, such as `kriging variables`, has been outlined in the previous sections. Generally, if a keyword is not expected to make any difference to the final result, its effect is deliberately made neutral. These keywords are expected to occur on a single line terminated by a carriage return – otherwise interpreted as the end of sequential formatted record. The input values that go with them are specified on the lines (records) immediately afterwards. Thus the general syntax for `krigin.txt` is;

```
“ keyword1
```

```

10.2 20.3 30.1
4
keyword2
3.14159 2.71828
1 2 3      ”

```

where of course the numbers are also for the purposes of demonstration. A list of all possible keywords follows, together with their usage and effects.

1. `kriging variables` : specifies the datasets in the `*.kr1` and `*.kr2` files to use. Integers refer to the sequential position of the dataset in these files.

For example,

```

kriging variables
3 8 1

```

refers to the 3<sup>rd</sup>, 8<sup>th</sup> and 1<sup>st</sup> datasets in the input binaries. Note that this also sets the output order of the covariance functions and statistics, and whenever there is an estimation, the leading dataset in the list is the primary (estimated) variable. In the example above, the third dataset in the binary file is estimated. Cokriging of up to four datasets may be performed.

2. `estimation` : should an estimation be made in the course of `krige`'s operation, this keyword specifies the type of estimation. As outlined in Section IV - 6, there are three possible estimations that can be made of a particular primary variable; an unfiltered estimate, a noise (nugget) filtered estimate, and an estimate of the mean or drift component. These are signified by 1, 2 or 3 respectively in the next line. For example,

```

estimation type
2

```

indicates that the estimate is to be filtered for the nugget effect, or noise.

3. `support` : indicates the block support of the estimate. Estimates and modelled variance relate to a given spatial support, currently a square

(cube or hyper-cube) block of dimensions indicated in the line following the keyword. For example,

```
support
1.25
```

would estimate the average value (and corresponding variance) in three dimensions of a cube of dimensions  $1.25 \times 1.25 \times 1.25$  (dimensions correspond whatever is used in the dataset). Specifying zero here amounts to point support.

4. `drift` : This option specifies the order of the polynomial drift functions to be used with universal kriging estimation. Enter integers in the subsequent line to specify the polynomial order for each variable in the cokriging. If the line is then terminated with `-1`, these drifts are modelled as dependent drifts (Section III - 6.2) but otherwise, non-independent drifts are used (Section III - 6.1). For example,

```
drift function orders
2 2 2 -1
```

would model the three variables in a cokriging as sharing the same quadratic drift. If nothing is specified or the line is incomplete or in error, ordinary kriging with independent means is used by default.

5. `mirror` : this is another universal kriging option that interpolates an underlying drift function from a field of point data (a secondary dataset) that is also resident in the input binary files `*.kr1` and `*.kr2`. This *sub*-interpolation step is effected by univariate ordinary kriging. The datasets that are to be used as interpolated drift functions are specified by their order in the input binaries, up to a maximum of four such interpolated drift functions. For example,

```
mirror interpolates
3 5 7
```

interpolates drift functions using the 3<sup>rd</sup>, 5<sup>th</sup> and 7<sup>th</sup> datasets in the input binaries. These interpolated drifts are modelled for all variables in the cokriging, and are treated as independent or dependent drifts on the basis of what is defined by the drift keyword. Note that including a dataset

twice or interpolating a drift function from a dataset that is itself a variable in the cokriging will result in singular matrices.

6. `raster` : defines a regular grid at which estimates are to be produced in raster interpolation mode. This mode produces a two-dimensional grid of estimates, regardless of the dimensionality of the kriged space. The four lines subsequent to this keyword are interpreted respectively as; an  $x$ -axis vector, a  $y$ -axis vector, an origin, and a desired integer resolution. For example in three dimensions,

```
raster window
1.0 2.0 -1.0
-1.0 1.0 1.0
2.0 3.0 4.0
10
```

produces estimates on a two dimensional grid centred on  $(2,3,4)$ , with an  $x$ -axis extending from  $(2,3,4)-(1,2,-1)$  to  $(2,3,4)+(1,2,-1)$ , and a  $y$ -axis extending from  $(2,3,4)-(-1,1,1)$  to  $(2,3,4)+(-1,1,1)$ . There are  $10 \times 2 + 1 = 21$  estimates along the shortest axis, and a corresponding grid spacing on the other axis. Furthermore there is always an estimate at the origin, around which the raster is generated.

7. `unstructured` : specifies the way in which the nugget effect is fitted in the linear model of coregionalisation. The nugget effect pertains to autocovariant structures, so this amounts to a series of values on the next line – as many as there are cokriging variables; relating to the datasets listed under `kriging variables` and in the same order. If the value is exactly zero, the nugget effect is forced to be zero by the two-sided constraint (4.50) in Section IV - 5.3. If the value is any positive number, likewise, the unstructured variance is set at that level using the same penalty function. However, when the value is negative the one-sided constraint (4.49) is used instead, allowing the total variance to ‘float’ to whatever value suits the spatial statistics best. For example,

```
unstructured variance
0.01 -1.0
```

specifies that the unstructured component of variance is to be set to 0.01

for the primary variable and left to float for the secondary variable in a two variable cokriging.

8. `structure order` : specifies whether to use variograms (Equation (4.7)) or spatial correlation coefficient (Equation (4.5)) to estimate the stationary structures. In the next line, `-1` indicates the correlogram whereas `0` indicates the variogram. For example,

```
structure order
0
```

indicates that variograms are to be used in structure identification (corresponding to intrinsic random functions of order zero, or IRF-0).

It should be noted that the file `krigin.txt` is only parsed for the keywords listed above, and anything else appearing on the same line is ignored – as is demonstrated in some of the examples above.

## B - 2. Program `datagen`

Program `datagen.exe` is used to prepare the input `*.kr1` and `*.kr2` files from `*.dat` text files containing data in the Tecplot ASCII format. The Tecplot `*.dat` files must be prepared in “FEPOINT” format or the executable will terminate with an error. This is a very simple program with sequential command prompt interaction with the user only.

As a brief introduction, Tecplot data files are arranged in zones and in each zone, there are columns containing the spatial coordinates and corresponding nodal data. This is illustrated in Figure B—6. There are a number of formats for the data in these zones, but `datagen` has so far only been programmed to handle Tecplot data in FEPOINT format. Upon execution, `datagen` firstly prompts the user for a database to which it will add datasets. This database is just the two binary files `*.kr1` and `*.kr2`, and they must be resident in the same directory as `datagen` is running in to be found. If no such database is available an empty one is created, with no datasets in it. As the dimensionality of the spatial data is an integral argument to the database, in this case the dimensionality will also be prompted for.

Having successfully opened the existing or new database, the program then prompts the user to add to it. Entering anything but x will set the program about adding a new dataset to the (possibly empty) database. When this happens, the user is first prompted for a Tecplot \*.dat file and a corresponding zone in this file from which to fetch the data. If no such file exists, there is an error message and the user is prompted to add another dataset. If the file does exist, datagen opens it and then prompts the user for which columns in the \*.dat file hold the spatial ordinates and which columns hold the corresponding nodal data. The program then parses the \*.dat file for this data, spatially clusters it and writes it to the \*.kr1 and \*.kr2 files.

When the user does not wish to add any more datasets, or the maximum limit

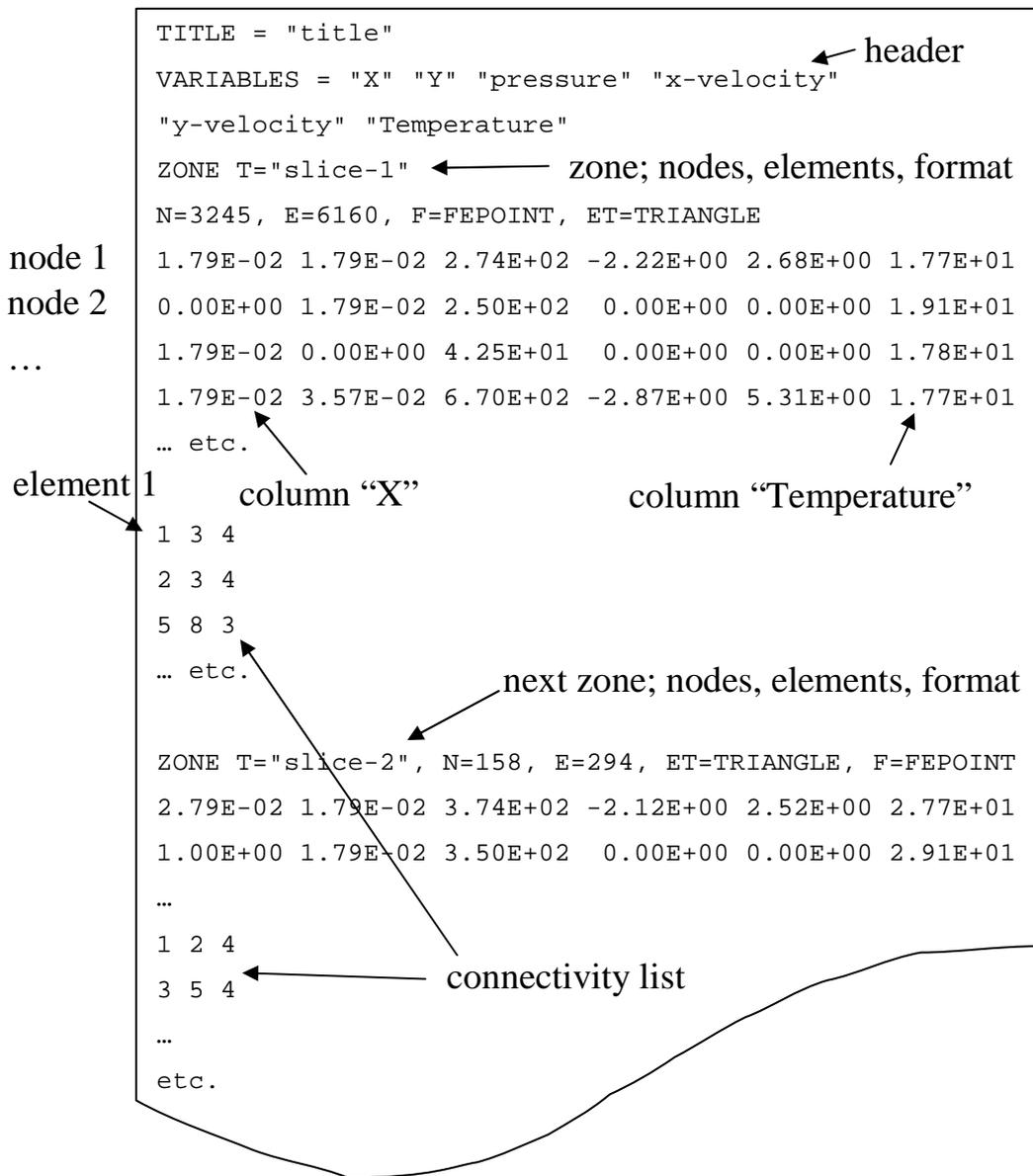


Figure B—6: Tecplot files in FEPOINT format.

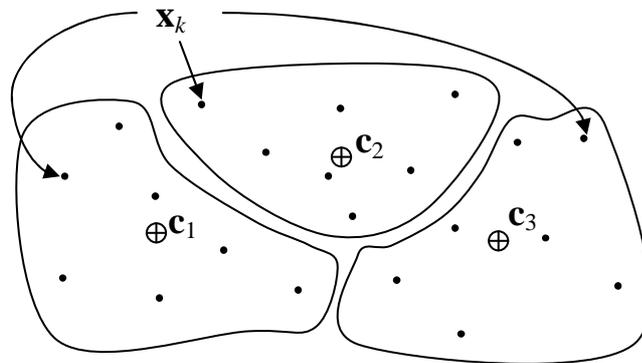
of eight such datasets is reached – `datagen` detrends the raw data now listed in the `*.kr1` file and stores it in the `*.kr2` file. Upon termination, these files are ready to be used by the program `krige`.

# Appendix C - Algorithm Details

In `datagen.exe`, use is made of the k-means clustering algorithm to split up and organise the data in a useful manner, and in `krige.exe`, use is made of the Levenberg-Marquardt algorithm for the solution of the non-linear least squares problem. Whilst these are both standard numerical algorithms, there are modifications to cater for specific code requirements and some arbitrary choices to be made. The specific implementation of these algorithms is described in the following appendix.

## C - 1. k-means Clustering Algorithm

The k-means clustering algorithm [121] is used to split up raw nodal data into spatially local groups to facilitate searches for proximal or ‘relevant’ data. It has been remarked that spatial datasets on fluid behaviour can be very large, so this step is largely to ensure that datasets can at least always be post-processed on modest desktop systems. This is particularly a problem encountered with CFD data, which can be generated on the order of millions of nodes. Whilst experimental surveys are not so formidable, they can still contain thousands of node locations – more if a particularly detailed study has been performed. These data are split up into mutually exclusive local clusters of nodes which are then stored on disk in a direct-access binary file. By storing one such cluster in each record of the direct-access file, even



**Figure C—1: Spatial clustering schematic.**

very large datasets can be efficiently searched and swapped out to disk – where storage is practically unlimited. The k-means algorithm has been used to generate the data clusters, with some modifications to ensure that the clusters are of similar size possessing roughly equal numbers of nodes. This is preferred, as the file records are all equal length for faster input/output, and it is sub-optimal to swap out very small or almost empty records.

Properly, k-means is an intra-cluster variance reduction problem. The algorithm described in the flowchart C—2 is Lloyd’s algorithm, and is typically used to solve the problem – thus the two names are often used interchangeably. Random centroids are picked from the data, and clusters are determined by grouping together points closest to a given centroid. The cluster centroids are then re-calculated and the clusters are re-determined iteratively until there is no further change. We require that the clusters are of reasonably equal size. If a cluster exceeds a given record length on disc it is split into smaller clusters. Also as the iterations proceed, the euclidean distances to each cluster centroid  $i$  are calculated and then further factored by,

$$f_i = \min(g_{ave}, g_i) \tag{C.1}$$

where  $g_{ave}$  is the average number of points per group, and  $g_i$  is the number of points

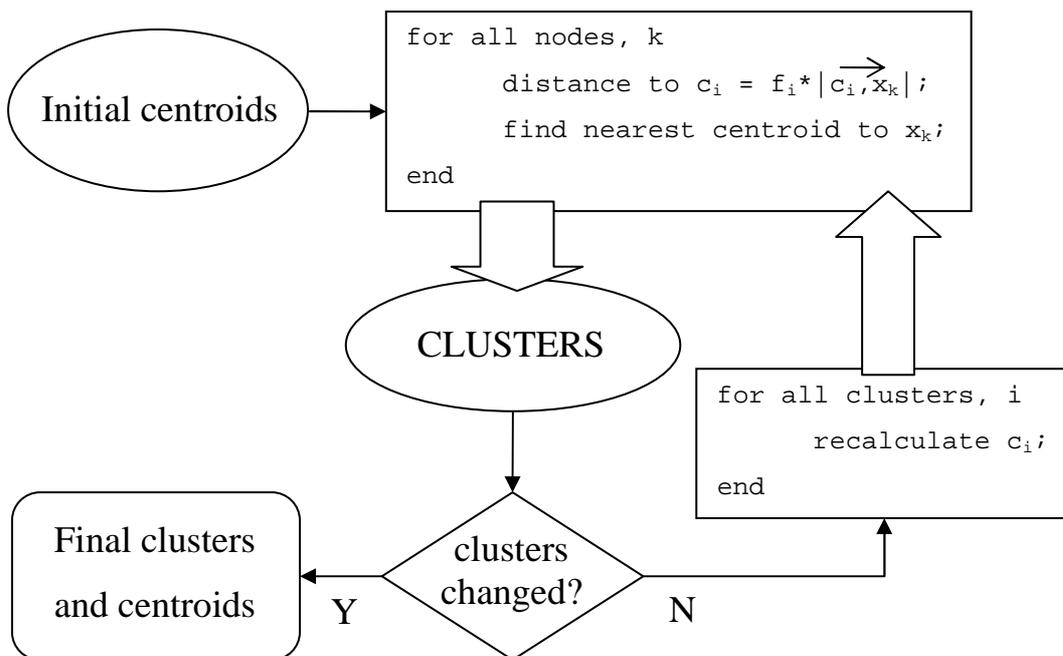


Figure C—2: Flowchart describing Lloyd’s algorithm

in a cluster  $i$ . This obviously distorts the statistical properties of the clustering, but these are not crucial – all that is required is a flexible means of splitting the data, intra-cluster variance reduction was viewed as possibly useful but not absolutely necessary.

## C - 2. Levenberg-Marquardt Algorithm

In Section IV - 5, parametric surfaces are fitted to various sets of points, by tuning their parameters. We have made use of the Levenberg-Marquardt algorithm to perform this fitting in a least squares sense. Levenberg-Marquardt is a non-linear least squares algorithm that iteratively solves the error minimisation problem to determine a solution set of parameters for the response surface [65, 66]. Its attraction lies in both its speed and its robustness, as it effectively interpolates between the Gauss-Newton method and simple gradient descent, using a constant  $\kappa$ . A brief explanation is provided below, to complement the explanations given in Section IV - 5.

Using the same notation as introduced in Section IV - 5, the non-linear least squares problem is:

$$\mathbf{r}' = \min_{\mathbf{r} \in \mathbb{R}^N} \left\{ \sum_k [G_k - g(\mathbf{x}_k; \mathbf{r})]^2 \right\}$$

where the function  $g(\mathbf{x}, \mathbf{r})$  is to be fitted to the points  $\{\mathbf{x}_i; G_i\}$  by adjusting the parameters  $\mathbf{r}$ . The iterative relation in (4.23) can be derived by approximating the sum-of-squares as;

$$\begin{aligned} E_{rr}(\mathbf{r}) &= \sum_k [g(\mathbf{x}_k; \mathbf{r}) - G_k]^2 \\ &\approx \sum_k \left[ g(\mathbf{x}_k; \mathbf{r}_i) + \sum_j \left. \frac{\partial g_k}{\partial r_j} \right|_{\mathbf{r}_i} \Delta r_j - G_k \right]^2 \\ &= [\mathbf{g}_i + \mathbf{J}_i \Delta \mathbf{r}_i - \mathbf{G}]^T [\mathbf{g}_i + \mathbf{J}_i \Delta \mathbf{r}_i - \mathbf{G}] \end{aligned} \tag{C.2}$$

where  $\mathbf{J}_i$  is the Jacobian matrix evaluated at  $\mathbf{r}_i$ . and  $\mathbf{g}_i$  is the function  $g(\mathbf{x}, \mathbf{r}_i)$  evaluated at the known points  $\mathbf{x}_k$ . This is a truncated Taylor series approximation,

which we may multiply out to obtain a new minimisation problem, noting that the transpose of a scalar is itself and constant terms do not affect the minimisation problem;

$$\begin{aligned}
& \min_{\mathbf{r} \in \mathbb{R}^N} E_{rr}(\mathbf{r}) \approx \\
& = \min_{\Delta \mathbf{r}_i \in \mathbb{R}^N} \left\{ \mathbf{g}_i^T \mathbf{g}_i + \mathbf{g}_i^T \mathbf{J}_i \Delta \mathbf{r}_i - \mathbf{g}_i^T \mathbf{G} + (\mathbf{J}_i \Delta \mathbf{r}_i)^T \mathbf{g}_i + \dots \right. \\
& \quad \left. \dots + (\mathbf{J}_i \Delta \mathbf{r}_i)^T \mathbf{J}_i \Delta \mathbf{r}_i - (\mathbf{J}_i \Delta \mathbf{r}_i)^T \mathbf{G} - \mathbf{G}^T \mathbf{g}_i - \mathbf{G}^T \mathbf{J}_i \Delta \mathbf{r}_i + \mathbf{G}^T \mathbf{G} \right\} \\
& = \min_{\Delta \mathbf{r}_i \in \mathbb{R}^N} \left\{ \Delta \mathbf{r}_i^T \mathbf{J}_i^T \mathbf{J}_i \Delta \mathbf{r}_i + 2 \mathbf{g}_i^T \mathbf{J}_i \Delta \mathbf{r}_i - 2 \mathbf{G}^T \mathbf{J}_i \Delta \mathbf{r}_i \right\} \\
& = \min_{\Delta \mathbf{r}_i \in \mathbb{R}^N} \left\{ \Delta \mathbf{r}_i^T \mathbf{J}_i^T \mathbf{J}_i \Delta \mathbf{r}_i + 2 \mathbf{J}_i^T [\mathbf{g}_i - \mathbf{G}] \Delta \mathbf{r}_i \right\}
\end{aligned}$$

This is the minimisation of a quadratic form over  $\delta \mathbf{r}_i$  which is solved by setting the gradient to zero.

$$\begin{aligned}
\nabla_{\Delta \mathbf{r}_i} \left\{ \Delta \mathbf{r}_i^T \mathbf{J}_i^T \mathbf{J}_i \Delta \mathbf{r}_i + 2 \mathbf{J}_i^T [\mathbf{g}_i - \mathbf{G}] \Delta \mathbf{r}_i \right\} &= \mathbf{0} \\
\mathbf{J}_i^T \mathbf{J}_i \Delta \mathbf{r}_i + \mathbf{J}_i^T [\mathbf{g}_i - \mathbf{G}] &= \mathbf{0} \\
\mathbf{J}_i^T \mathbf{J}_i \Delta \mathbf{r}_i &= \mathbf{J}_i^T [\mathbf{G} - \mathbf{g}_i]
\end{aligned} \tag{C.3}$$

This set of equations may be solved for  $\delta \mathbf{r}_i$  to find the Gauss-Newton iteration step.

The Levenberg-Marquardt algorithm modifies the relation in (C.3) by adding a unit matrix damping term to the Hessian matrix  $\mathbf{J}_i^T \mathbf{J}_i$ , replacing it with  $(\mathbf{J}_i^T \mathbf{J}_i + \mathbf{I})$ , which gives us the relation in (4.23),

$$\left[ \mathbf{J}_i^T \mathbf{J}_i + \mathbf{I} \right] \Delta \mathbf{r}_i = \mathbf{J}_i^T [\mathbf{G} - \mathbf{g}_i] .$$

The damping coefficient  $\kappa$  is adjusted as the iterations progress to yield better convergence and speed. For small  $\kappa$  we see that the algorithm approximates the aggressive Gauss-Newton step, and for large  $\kappa$  the algorithm approximates the conservative gradient descent step.

The scheme utilised in the program involves adjusting  $\kappa$  by multiplicative constants according to the behaviour of the log of the sum of squares,  $E_{rr}(\mathbf{r})$ . In heuristic testing, it was noticed that the change in this quantity each iteration would typically show one of three behaviours if  $\kappa$  was kept constant, and the algorithm was

convergent. The damping coefficient was adjusted on the basis of the observed behaviour, considering that a more aggressive, underdamped scheme will converge faster.

At each iteration, the quantity  $\ln(E_{rr}(\mathbf{r}_k))$  is formed – the log function was chosen for convenience, and as we may expect the convergence to be in some sense comparable to an exponential decay. Such a normalisation permits the comparison of a sequence of ‘self-similar’ decreasing values.

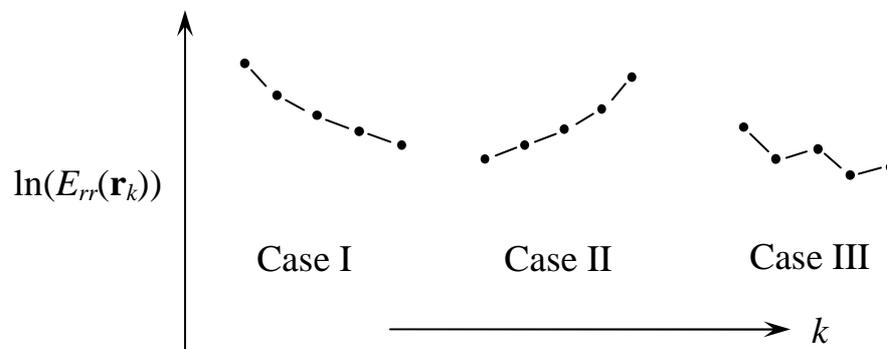
$$\dots \ln[E_{rr}^{k-3}], \ln[E_{rr}^{k-2}], \ln[E_{rr}^{k-1}], \ln[E_{rr}^k] \quad (\text{C.4})$$

We examine the difference in these values per iteration  $\Delta_k$ .

$$\Delta_k = \ln[E_{rr}^k] - \ln[E_{rr}^{k-1}] \quad (\text{C.5})$$

If the algorithm is converging on a minimum, the values  $\Delta_k$  must be negative. To examine the rate of convergence of the algorithm, we plotted the values  $\Delta_k$  against iteration number. Three general behaviours were discovered once the algorithm had established a convergent sequence – these are illustrated in Figure C—3. The values would either increase in negativity (become smaller), decrease in negativity (become larger) or alternately increase and decrease in negativity (oscillate). Heuristically, we interpreted that these cases corresponded to optimally, over and under-damped iterations. Positive  $\Delta_k$  (divergence) is treated later as a separate case.

To control the convergent behaviours, the following strategy has been employed. Whilst crude, it does seem to work quite well – provided that  $\kappa$  is not permitted to get too large, and the Hessian matrix at convergence is not too close to



**Figure C—3: Cases for damping.**

singularity. Firstly, the damping coefficient may not be changed within four iterations of a prior change. This is to allow the iterations to settle into a pattern after applying a new damping constant, and also to observe at least three values of  $\ln(E_{rr}(\mathbf{r}))$  under the new damping regime by which criterion  $\kappa$  is modified. If  $\Delta$  monotonically increases (Case I, Figure C—3) the damping value is not changed as it is considered optimal. If  $\Delta$  monotonically decreases (Case II, Figure C—3) the damping value is reduced to offer faster convergence, and if oscillatory behaviour is observed (Case III, Figure C—3) it is increased to stabilise the iterations. The coefficient  $\kappa$  is increased and decreased by the multiplicative factors  $c_+$  and  $c_-$  respectively.

If a positive  $\Delta$  is returned, it is assumed that the truncated Taylor series in (C.2) is misrepresenting the actual function and so the iterations are heavily damped by a factor  $c_{div}$  to prevent divergent behaviour. From experience, the following values were found to offer reasonable performance and good robustness for problems typical to their application in our code.

$$c_+ = 1.06, \quad c_- = 0.5, \quad c_{div} = 2.05 \quad (\text{C.6})$$

These values were chosen heuristically, but note that to avoid cycles of damping and successive undamping, they are not integer multiples of each other. This algorithm is used in two parts of module `krige`; in subroutines `trane` and `koreg`. The particular objective functions, squared residual vectors, Jacobians and relevant constraints applied as penalty functions, have been described in the main body of the thesis in Section IV - 5.

# Appendix D - Derivations

## D - 1. Derivation of Ordinary Kriging Variance

Equation (3.15) in Section III - 2 presents a brief derivation of kriging variance;  $\text{var}\{\hat{p}_0 - p_0\}$ , the full derivation is shown below. A general statistical identity [62] shall be used in the subsequent discourse;

$$\text{var}(aA + bB) = a^2 \text{var}(A) + b^2 \text{var}(B) + 2ab \text{cov}(A, B) \quad (\text{D.1})$$

where  $A$  and  $B$  are random variables multiplied respectively by scalar coefficients  $a$  and  $b$ . Considering that

$$\text{var}(A) = \text{cov}(A, A) ,$$

for multiple random variables  $A_1, A_2, A_3 \dots$  multiplied respectively by scalar coefficients  $a_1, a_2, a_3 \dots$  the relation in Equation (D.1) becomes generally

$$\text{var}\left(\sum_i a_i A_i\right) = \sum_{i,j} a_i a_j \text{cov}(A_i, A_j) . \quad (\text{D.2})$$

A second statistical identity defining the covariance function in terms of the expectation function is also used;

$$\text{cov}(A, B) = E(AB) - E(A)E(B) , \quad (\text{D.3})$$

as essentially presented earlier in Equation (3.2).

The kriging variance is the variance of the estimation error on the basis of the covariance model, so the first step is to substitute the estimator (Equation (3.14)) for  $\hat{p}_0$ ;

$$\text{var}\{\hat{p}_0 - p_0\} = \text{var}\left\{\sum_i w_i P(\mathbf{x}_i) - P(\mathbf{x}_0)\right\} . \quad (\text{D.4})$$

Noting that  $P(\mathbf{x})$  is second order stationary with mean  $\mu$  and variance of  $\frac{2}{p}$  this can be rewritten using the first Equation (D.1) as

$$\text{var}\{\hat{p}_0 - p_0\} = \text{var}\left\{\sum_i w_i P(\mathbf{x}_i)\right\} + \text{var}\{P(\mathbf{x}_0)\} - 2\text{cov}\left\{\sum_i w_i P(\mathbf{x}_i), P(\mathbf{x}_0)\right\}$$

which, using the relation in Equation (D.2) and noting that  $\text{var}(P(\mathbf{x})) = \frac{\sigma^2}{P}$ ,

$$= \sum_{i,j} w_i w_j \text{cov}\{P(\mathbf{x}_i), P(\mathbf{x}_j)\} + \frac{\sigma^2}{P} - 2\text{cov}\left\{\sum_i w_i P(\mathbf{x}_i), P(\mathbf{x}_0)\right\}.$$

The last term in this is expanded in terms of the expectation function in Equation (D.3) so that

$$\begin{aligned} &= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \frac{\sigma^2}{P} + \dots \\ &\dots - 2\left[\text{E}\left\{\sum_i w_i P(\mathbf{x}_i) P(\mathbf{x}_0)\right\} - \text{E}\left\{\sum_i w_i P(\mathbf{x}_i)\right\} \text{E}\{P(\mathbf{x}_0)\}\right]. \end{aligned}$$

As the expectation function is linear, the term in the square brackets above may be rewritten with all such functions inside the summation, and noting that due to stationarity  $\text{E}\{P(\mathbf{x}_i)\} = \bar{P}$ ,

$$\begin{aligned} &\text{var}\{\hat{p}_0 - p_0\} \dots \\ &= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \frac{\sigma^2}{P} - 2\left[\sum_i w_i \text{E}(P(\mathbf{x}_i) P(\mathbf{x}_0)) - \sum_i w_i \bar{P}^2\right] \\ &= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \frac{\sigma^2}{P} - 2\sum_i w_i \left[\text{E}(P(\mathbf{x}_i) P(\mathbf{x}_0)) - \bar{P}^2\right] \end{aligned}$$

which, given the definition of the covariance function, yields

$$\text{var}\{\hat{p}_0 - p_0\} = \frac{\sigma^2}{P} + \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) - 2\sum_i w_i \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_0)), \quad (\text{D.5})$$

finally proving the relation in Equation (3.15).

## D - 2. Unbiasedness for Ordinary Kriging

The full derivation for the constraint on the weights presented in Equation (3.20), Section III - 2, is presented here. Starting with the first line of this equation, where the estimator in Equation (3.14) has been substituted for  $\hat{p}_0$ ;

$$E\{\hat{p}_0 - p_0\} = E\left\{\sum_i w_i P(\mathbf{x}_i) - P(\mathbf{x}_0)\right\}, \quad (\text{D.6})$$

then as the expectation function is linear, and  $E\{P(\mathbf{x})\} = \mu$  where  $P(\mathbf{x})$  is stationary,

$$\begin{aligned} E\{\hat{p}_0 - p_0\} &= \sum_i w_i E\{P(\mathbf{x}_i)\} - E\{P(\mathbf{x}_0)\} \\ &= \sum_i w_i \mu - \mu \end{aligned}$$

and because the estimator is to be unbiased (Equation (3.18))

$$\begin{aligned} \sum_i w_i \mu - \mu &= 0 \\ \left(\sum_i w_i - 1\right) \mu &= 0. \end{aligned} \quad (\text{D.7})$$

In general  $\mu \neq 0$ , therefore

$$\begin{aligned} \sum_i w_i - 1 &= 0 \\ \sum_i w_i &= 1 \end{aligned} \quad (\text{D.8})$$

as required for Equation (3.20).

### D - 3. Derivation of Universal Kriging Variance

In Section III - 4 an outline derivation of kriging variance in the presence of a drift was presented in Equation (3.36). The full derivation is provided below, with reference to the notation in this section, wherein the second order stationary function  $Q(\mathbf{x})$  now comprises a drift  $\mu(\mathbf{x})$  and a zero-mean second order stationary random function component  $P(\mathbf{x})$  (Equation (3.30)).

$$Q(\mathbf{x}) = P(\mathbf{x}) + \mu(\mathbf{x}) \quad (\text{D.9})$$

As before the variance of the estimation error is expressed in terms of the estimator in Equation (3.31), incorporating the above relation,

$$\begin{aligned}
\text{var}\{\hat{q}_0 - q_0\} &= \text{var}\left\{\sum_i w_i Q(\mathbf{x}_i) - Q(\mathbf{x}_0)\right\} \\
&= \text{var}\left\{\sum_i w_i [P(\mathbf{x}_i) - (\mathbf{x}_i)] - [P(\mathbf{x}_0) - (\mathbf{x}_0)]\right\}
\end{aligned} \tag{D.10}$$

whereupon the non-random mean is separated and removed,

$$\begin{aligned}
\text{var}(\hat{q}_0 - q_0) &= \text{var}\left\{\sum_i w_i P(\mathbf{x}_i) - P(\mathbf{x}_0) - \sum_i w_i (\mathbf{x}_i) + (\mathbf{x}_0)\right\} \\
&= \text{var}\left\{\sum_i w_i P(\mathbf{x}_i) - P(\mathbf{x}_0)\right\}
\end{aligned}$$

because the non-random components do not affect the variance. The last result is of course identically Equation (D.4), and so the kriging variance may finally be expressed as

$$\text{var}(\hat{q}_0 - q_0) = \frac{2}{p} + \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) - 2 \sum_i w_i \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_0)) \tag{D.11}$$

as established originally in Equation (3.36).

## D - 4. Unbiasedness for Universal Kriging

In Equation (3.33) of Section III - 4, a brief outline of how to achieve the unbiasedness constraints in the case of an unknown drift was presented. This proof is re-iterated in detail below. The notation is inherited from this section, as was noted in the previous section; Appendix D - 3. Note that the mean is a function over the domain of estimation;

$$(\mathbf{x}) = \sum_i a_i f^i(\mathbf{x}) \tag{D.12}$$

as reproduced from Equation (3.29).

The expectation of the residual error is first expressed in terms of the estimator in Equation (3.31), and then the stationary random function  $P(\mathbf{x})$ .

$$\begin{aligned}
E\{\hat{q}_0 - q_0\} &= E\left\{\sum_i w_i Q(\mathbf{x}_i) - Q(\mathbf{x}_0)\right\} \\
&= E\left\{\sum_i w_i [P(\mathbf{x}_i) + \epsilon(\mathbf{x}_i)] - [P(\mathbf{x}_0) + \epsilon(\mathbf{x}_0)]\right\}
\end{aligned} \tag{D.13}$$

Because the expectation function is linear and can enter the brackets and summed terms, the above can be re-expressed as

$$E\{\hat{q}_0 - q_0\} = \sum_i w_i E\{P(\mathbf{x}_i) + \epsilon(\mathbf{x}_i)\} - E\{P(\mathbf{x}_0) + \epsilon(\mathbf{x}_0)\}$$

Given that the random function  $P(\mathbf{x})$  is zero-mean, the expectation function only picks up the drift component  $\mu(\mathbf{x})$  therefore,

$$\begin{aligned}
E\{\hat{q}_0 - q_0\} &= \sum_i w_i (\mu(\mathbf{x}_i) - \mu(\mathbf{x}_0)) \\
&= \sum_{i,j} w_i a_j f^j(\mathbf{x}_i) - \sum_i a_j f^j(\mathbf{x}_0) \\
&= \sum_j a_j \left[ \sum_i w_i f^j(\mathbf{x}_i) - f^j(\mathbf{x}_0) \right]
\end{aligned}$$

after substituting Equation (D.12), and rearranging. For an unbiased estimate the expectation of the estimation error must be zero (Equation (3.18)) so setting the last expression to zero,

$$\begin{aligned}
\sum_j a_j \left[ \sum_i w_i f^j(\mathbf{x}_i) - f^j(\mathbf{x}_0) \right] &= 0 \quad \therefore \\
\sum_i w_i f^j(\mathbf{x}_i) - f^j(\mathbf{x}_0) &= 0 \\
\sum_i w_i f^j(\mathbf{x}_i) &= f^j(\mathbf{x}_0)
\end{aligned} \tag{D.14}$$

as required for Equation (3.33).

## D - 5. Derivation of Cokriging Variance

In Section III - 6, Equation (3.48), a brief derivation of the cokriging variance was presented. The full derivation is presented below. Notation is taken from this chapter, greek letters are used to index datasets, and roman letters are used to index

the nodes within these datasets, as described in Section III - 6 by Equation (3.45). Note that the first dataset  $P^1$  is the estimated random function.

Starting with the familiar substitution for the estimator in Equation (3.46);

$$\begin{aligned} \sigma_{CK}^2 &= \text{var} \left\{ \hat{p}_0^1 - p_0^1 \right\} \\ &= \text{var} \left\{ \sum_i w_i P(\mathbf{x}_i) - P^1(\mathbf{x}_0) \right\}. \end{aligned} \quad (\text{D.15})$$

the variance operator is split up using the relation in Equation (D.1) so that

$$= \text{var} \left\{ \sum_i w_i P(\mathbf{x}_i) \right\} + \text{var} \left\{ P^1(\mathbf{x}_0) \right\} - 2 \text{cov} \left\{ \sum_i w_i P(\mathbf{x}_i), P^1(\mathbf{x}_0) \right\}$$

whereupon, using the relation in Equation (D.2) and noting that  $\text{var}(P^1(\mathbf{x})) = \sigma_1^2$ ,

$$= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \sigma_1^2 - 2 \text{cov} \left\{ \sum_i w_i P(\mathbf{x}_i), P^1(\mathbf{x}_0) \right\}.$$

The last term in this is expanded in terms of the expectation function in Equation (D.3);

$$\begin{aligned} &= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \sigma_1^2 + \dots \\ &\dots - 2 \left\{ \text{E} \left[ \sum_i w_i P(\mathbf{x}_i) P^1(\mathbf{x}_0) \right] - \text{E} \left[ \sum_i w_i P(\mathbf{x}_i) \right] \text{E} \left[ P^1(\mathbf{x}_0) \right] \right\}. \end{aligned}$$

As the expectation function is linear, the term in the square brackets above may be rewritten with all such functions inside the summation, and noting that  $\text{E}\{P(\mathbf{x})\} = \mu(\mathbf{x})$ ,

$$\begin{aligned} &= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \sigma_1^2 + \dots \\ &\dots - 2 \left\{ \sum_i w_i \text{E} \left[ P(\mathbf{x}_i) P^1(\mathbf{x}_0) \right] - \sum_i w_i \mu(\mathbf{x}_i) P^1(\mathbf{x}_0) \right\} \\ &= \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) + \sigma_1^2 + \dots \\ &\dots - 2 \sum_i \left[ w_i \left\{ \text{E} \left[ P(\mathbf{x}_i) P^1(\mathbf{x}_0) \right] - \mu(\mathbf{x}_i) P^1(\mathbf{x}_0) \right\} \right] \end{aligned}$$

the last term of which, is by definition, the covariance function so that finally

$$\begin{aligned}
& \overset{2}{C_K} \dots \\
& = \overset{2}{1} + \sum_{i,j} w_i w_j \text{cov}(P(\mathbf{x}_i), P(\mathbf{x}_j)) - 2 \sum_i w_i \text{cov}(P(\mathbf{x}_i), P^1(\mathbf{x}_0)) \quad (\text{D.16})
\end{aligned}$$

as required. The abbreviated notation for the stationary covariance function  $C(\cdot)$  may be employed so that, equally

$$\overset{2}{C_K} = \overset{2}{1} + \sum_{i,j} w_i w_j C(\mathbf{x}_i, \mathbf{x}_j) - 2 \sum_i w_i C_1(\mathbf{x}_i, \mathbf{x}_0),$$

as actually presented in Equation (3.48).

## D - 6. Jacobian for Basic Variogram Model Fit

In Section IV - 5.1 a result for the Jacobian of the model covariance function was presented, firstly with respect to the variables  $v_{ab}^c$  (Equation (4.38)), and then with respect to the variables  $\theta_d$  (Equation (4.39)). The full derivation of this result is presented in this appendix, wherein notation and terminology shall be continued from Section IV - 5.1, where it was originally introduced. Note that the modelled covariance function is expressed as

$$\mathbf{C}(\mathbf{h}) = \sum_i \left\{ \frac{\exp(t_i)}{1 + \sum_j \exp(t_j)} \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}} \right) \right] \right\} \quad (\text{D.17})$$

reproducing Equation (4.35), and the leading term here is more compactly expressed as

$$t_i = \frac{\exp(t_i)}{1 + \sum_j \exp(t_j)} \quad (\text{D.18})$$

reproducing Equation (4.33).

For the Jacobian  $\mathbf{J}$  in the Levenberg-Marquardt least squares optimisation scheme the derivative with respect to the coefficient  $v_{ab}^c$  is required. This is

$$\frac{\partial \mathbf{C}(\mathbf{h}^e)}{\partial v_{ab}^c} = \frac{\partial}{\partial v_{ab}^c} \left\{ \sum_i \left\{ t_i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}^e} \right) \right] \right\} \right\} \quad (\text{D.19})$$

which yields, noting that the derivative of constant terms with respect to  $v_{ab}^c$  is zero;

$$\frac{\partial C(\mathbf{h}^e)}{\partial v_{ab}^c} = -t_c \frac{\partial}{\partial v_{ab}^c} \Gamma^e \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \right)$$

Using the chain rule firstly on the  $\gamma(\cdot)$  function and then on the square root;

$$\begin{aligned} &= -t_c \Gamma^{*c} \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \right) \frac{\partial}{\partial v_{ab}^c} \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \\ &= \frac{-t_c \Gamma^{*c} \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \right)}{2 \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e}} \frac{\partial}{\partial v_{ab}^c} \left[ \mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e \right] \end{aligned}$$

where  $\gamma'(h)$  is the derivative of  $\gamma(h)$  with respect to  $h$ . Then, given that only the  $c^{\text{th}}$  term in the differentiated summation is not constant with respect to  $v_{ab}^c$ ,

$$\frac{\partial C(\mathbf{h}^e)}{\partial v_{ab}^c} = \frac{-t_c \Gamma^{*c} \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \right)}{2 \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e}} \frac{\partial}{\partial v_{ab}^c} \left[ \mathbf{h}^{eT} (\mathbf{v}_b^c \mathbf{v}_b^{cT}) \mathbf{h}^e \right]. \quad (\text{D.20})$$

Now, expressing the vectors in the final inner product indicially – where the leftmost subscript indicates the position within the vector;

$$\frac{\partial}{\partial v_{ab}^c} \left[ \mathbf{h}^{eT} (\mathbf{v}_b^c \mathbf{v}_b^{cT}) \mathbf{h}^e \right] = \frac{\partial}{\partial v_{ab}^c} \left[ \sum_{j,k} h_j^e v_{jb}^c v_{kb}^c h_k^e \right] \quad (\text{D.21})$$

which may be split into four separate sums, based on whether or not  $j = a$  or  $k = a$ ;

$$= \frac{\partial}{\partial v_{ab}^c} \left[ h_a^e v_{ab}^c h_a^e + \sum_{\substack{j=a \\ k \neq a}} h_a^e v_{ab}^c v_{kb}^c h_k^e + \sum_{\substack{j \neq a \\ k=a}} h_j^e v_{jb}^c v_{ab}^c h_a^e + \sum_{\substack{j \neq a \\ k \neq a}} h_j^e v_{jb}^c v_{kb}^c h_a^e \right].$$

Clearly when  $j \neq a$  and  $k \neq a$ , the terms in the summation are constant with respect to  $v_{ab}^c$  and the differential is zero – the differentiation of the other summations is equally straightforward;

$$= 2h_a^e v_{ab}^c h_a^e + \sum_{k:k \neq a} h_a^e v_{kb}^c h_k^e + \sum_{j:j \neq a} h_j^e v_{jb}^c h_a^e$$

which is easily regrouped, so that

$$\begin{aligned}
&= \sum_k h_a^e v_{kb}^c h_k^e + \sum_j h_j^e v_{jb}^c h_a^e \\
&= 2 \sum_k h_a^e v_{kb}^c h_k^e \\
&= 2 h_a^e \mathbf{v}_b^{cT} \mathbf{h}^e
\end{aligned}$$

in vector notation. In summary then,

$$\frac{\partial}{\partial v_{ab}^c} \left[ \mathbf{h}^{eT} (\mathbf{v}_b^c \mathbf{v}_b^{cT}) \mathbf{h}^e \right] = 2 h_a^e \mathbf{v}_b^{cT} \mathbf{h}^e \quad (\text{D.22})$$

The relation in Equation (D.22) is substituted for the final term in Equation (D.20) to arrive at

$$\frac{\partial C(\mathbf{h}^e)}{\partial v_{ab}^c} = \frac{-t_c \Gamma^{ac} \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e} \right) h_a^e \mathbf{v}_b^{cT} \mathbf{h}^e}{\sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^c \mathbf{v}_k^{cT}) \mathbf{h}^e}}, \quad (\text{D.23})$$

the final relation presented in Equation (4.38). This forms the greater part of the Jacobian matrix for the basic variogram model, however the derivatives with respect to the variables  $x_i$  in Equation (D.18) are also required.

Differentiating Equation (D.17) with respect now to  $\theta_d$ ,

$$\frac{\partial C(\mathbf{h}^e)}{\partial \theta_d} = \frac{\partial}{\partial \theta_d} \sum_i \left\{ \frac{\exp(\theta_i)}{1 + \sum_j \exp(\theta_j)} \left[ 1 - \Gamma^i \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}^e} \right) \right] \right\}, \quad (\text{D.24})$$

in which it is noted that the term in the square brackets is constant with respect to  $\theta_d$ . Also, all of the terms in the summation over  $i$  in Equation (D.24) involve  $\theta_d$ , thus none of them are zero after differentiation. Differentiating just the leading term in  $\boldsymbol{\theta}$  (effectively  $t_i$ , in Equation (D.18)), it is easily shown that;

$$\frac{\partial}{\partial \theta_d} \left[ \frac{\exp(\theta_i)}{1 + \sum_j \exp(\theta_j)} \right] = \begin{cases} \frac{-\exp(\theta_i) \exp(\theta_d)}{\left[ 1 + \sum_j \exp(\theta_j) \right]^2}, & \forall i \neq d \\ \frac{\exp(\theta_d)}{1 + \sum_j \exp(\theta_j)} - \frac{\exp(2\theta_d)}{\left[ 1 + \sum_j \exp(\theta_j) \right]^2}, & i = d \end{cases} \quad (\text{D.25})$$

Note that the first case  $i \neq d$  is much like the second  $i = d$ , save for the addition of a term – actually the coefficient  $t_d$ , thus the differentiation in Equation (D.24) proceeds to

$$\begin{aligned} \frac{\partial C(\mathbf{h}^e)}{\partial t_d} = & \frac{\exp(t_d)}{1 + \sum_j \exp(t_j)} \left[ 1 - I^d \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^d \mathbf{v}_k^{dT}) \mathbf{h}^e} \right) \right] + \dots \\ & - \sum_i \frac{\exp(t_i) \exp(t_d)}{\left( 1 + \sum_j \exp(t_j) \right)^2} \left[ 1 - I^i \left( \sqrt{\mathbf{h}^{eT} \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}^e} \right) \right] \end{aligned} \quad (\text{D.26})$$

Because in Equation (D.25) the term comprising the  $i \neq d$  case also appears in the  $i = d$  case, it is seen above that this term multiplies all of the terms in square brackets in Equation (D.24) over the entire summation. Additionally, the extra term arising in the  $i = d$  case appears in the first line of Equation (D.26).

The result in Equation (D.26) is presented in Section IV - 5.1 as Equation (4.39), and completes the derivation of the Jacobian for this section.

## D - 7. Jacobian for Coregionalisation Model Fit

In Section IV - 5.2, the Jacobian of the coregionalisation and thus complete covariance model, that is used finally in estimation, was presented. This Jacobian is formed with respect to the coefficients  $s^i$  in this model's definition, where the model

$$C(\mathbf{h}) = \sum_i s^i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right] \quad (\text{D.27})$$

is reproduced above from Equation (3.69). This model is reparameterised by

$$s^i = \sum_k u_k^i u_k^i$$

so that effectively the model might be expressed as

$$C(\mathbf{h}) = \sum_i \left( \sum_k u_k^i u_k^i \left[ 1 - I^i \left( \sqrt{\mathbf{h}^T \mathbf{T}^i \mathbf{h}} \right) \right] \right) \quad (\text{D.28})$$

The Jacobian of (D.28) with respect to the variables  $u_k^i$  is formed by differentiating

$$\frac{\partial C(\mathbf{h}^e)}{\partial u_{ab}^c} = \frac{\partial}{\partial u_{ab}^c} \left\{ \sum_i \left( \sum_k u_k^i u_k^i \left[ 1 - I^* \left( \sqrt{\mathbf{h}^e \text{ }^T \mathbf{T} \mathbf{h}^e} \right) \right] \right) \right\}$$

and noting that where  $i \neq c$  the summation terms are constant with respect to  $u_k^i$  as are the terms in the square brackets.

$$\begin{aligned} &= \frac{\partial}{\partial u_{ab}^c} \sum_k u_k^c u_k^c \left[ 1 - I^* \left( \sqrt{\mathbf{h}^e \text{ }^T \mathbf{T} \mathbf{h}^e} \right) \right] \\ &= \frac{\left[ 1 - I^* \left( \sqrt{\mathbf{h}^e \text{ }^T \mathbf{T} \mathbf{h}^e} \right) \right]}{\partial u_{ab}^c} u_b^c u_b^c \end{aligned} \quad (\text{D.29})$$

Examining the last term of Equation (D.29), it arrives again that there are four cases to consider (seen before in Equation (D.21)):

$$\frac{\partial}{\partial u_{ab}^c} u_b^c u_b^c = \begin{cases} 2u_{ab}^c, & = = a \\ u_b^c, & = a, \neq a \\ u_b^c, & \neq a, = a \\ 0, & \text{otherwise} \end{cases} \quad (\text{D.30})$$

or,

$$\frac{\partial}{\partial u_{ab}^c} u_b^c u_b^c = \begin{bmatrix} & & & u_{1b}^c & & \\ & \mathbf{0} & & u_{1b}^c & \mathbf{0} & \\ & & & \vdots & & \\ u_{1b}^c & u_{2b}^c & \cdots & 2u_{ab}^c & \cdots & \\ & \mathbf{0} & & \vdots & & \mathbf{0} \end{bmatrix}.$$

and substituting the above Equation (D.30) into (D.29) for this last term, the final expression for the Jacobian is obtained;

$$\frac{\partial C(\mathbf{h}^e)}{\partial u_{ab}^c} = \begin{cases} \frac{2 \left[ 1 - \Gamma^{\sigma} \left( \sqrt{\mathbf{h}_{aa}^{e \text{ T}} \mathbf{T}^c \mathbf{h}_{aa}^e} \right) \right] u_{ab}^c}{a}, & = a \\ \frac{\left[ 1 - \Gamma^{\sigma} \left( \sqrt{\mathbf{h}_a^{e \text{ T}} \mathbf{T}^c \mathbf{h}_a^e} \right) \right] u_b^c}{a}, & = a, \neq a \\ \frac{\left[ 1 - \Gamma^{\sigma} \left( \sqrt{\mathbf{h}_a^{e \text{ T}} \mathbf{T}^c \mathbf{h}_a^e} \right) \right] u_b^c}{a}, & \neq a, = a \\ 0, & \text{otherwise} \end{cases} \quad (\text{D.31})$$

as was originally offered in Equation (4.44).

## D - 8. Jacobian for Penalty Functions

In Section IV - 5.3 a penalty function was introduced to constrain the optimisation of (4.51) and force certain modelling conditions. This function, reproduced below

$$pen = \left[ f_{con} \left\{ \frac{1}{\left[ \sum_{i,n} u_n^i u_n^i - V \right]} \right\}^2 \right]$$

is to be incorporated into the optimisation, thus its Jacobian with respect to the variables  $u_{ab}^c$  in Equation (D.28) was required. The derivation of this Jacobian is presented in this section.

The penalty function is differentiated in the normal way

$$\frac{\partial pen}{\partial u_{ab}^c} = \frac{\partial}{\partial u_{ab}^c} \left[ f_{con} \left\{ \frac{1}{\left[ \sum_{i,n} u_n^i u_n^i - V \right]} \right\}^2 \right]$$

then using the chain rule on the square term and noting the constant terms

$$\begin{aligned} &= \frac{2f_{con}}{\left[ \sum_{i,n} u_n^i u_n^i - V \right]} \frac{\partial}{\partial u_{ab}^c} \left\{ \frac{\sum_{i,n} u_n^i u_n^i}{\left[ \sum_{i,n} u_n^i u_n^i - V \right]} \right\} \\ &= \frac{2f_{con}}{2} \frac{\partial (u_b^c u_b^c)}{\partial u_{ab}^c}, \end{aligned} \quad (\text{D.32})$$

noting that unless  $i = c$ , the terms in the summation above are constant with respect to  $u_{ab}^c$ . Examining the final differential term in Equation (D.32), the relation in Equation (D.30) is substituted directly to obtain

$$\frac{\partial^{pen}}{\partial u_{ab}^c} = \begin{cases} \frac{2f_{con}}{2} \frac{2f_{con}}{2} \left[ \sum_i s_{aa}^i - V_{aa} \right] 2u_{ab}^c, & = a \\ \frac{2f_{con}}{2} \frac{2f_{con}}{2} \left[ \sum_i s_a^i - V_a \right] u_b^c, & = a, \neq a \\ \frac{2f_{con}}{2} \frac{2f_{con}}{2} \left[ \sum_i s_a^i - V_a \right] u_b^c, & \neq a, = a \\ 0, & \text{otherwise} \end{cases} \quad (D.33)$$

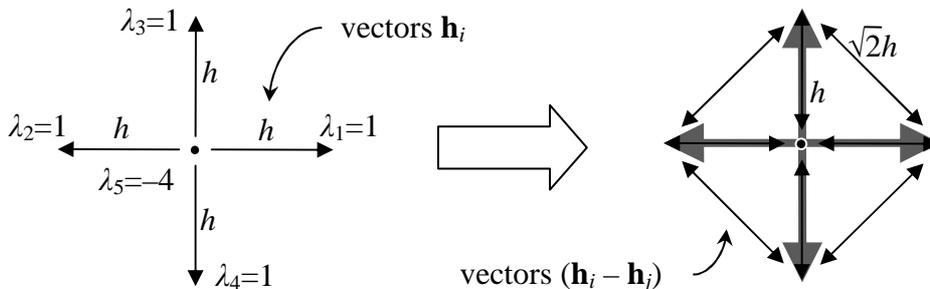
the final result, presented originally in Equation (4.53).

## D - 9. Laplacian Variance in Terms of Generalised Covariance

In Sections V - 2 and V - 3, the variance of the Laplacian template at a scale of  $h$  was re-expressed in terms of the generalised covariance, defined in Equation (5.11). The Laplacian is an Allowable Linear Combination of order one (ALC-1), and may be expressed in the form

$$\begin{aligned} Z(\mathbf{x}) &= \sum_i \lambda_i Z(\mathbf{h}_i + \mathbf{x}) \\ &= Z(\mathbf{x} + h\hat{\mathbf{i}}) + Z(\mathbf{x} - h\hat{\mathbf{i}}) + Z(\mathbf{x} + h\hat{\mathbf{j}}) + Z(\mathbf{x} - h\hat{\mathbf{j}}) - 4Z(\mathbf{x}) \end{aligned} \quad (D.34)$$

where  $Z(\lambda)$  is the Laplacian ALC-1,  $h$  is the size of the template and  $\mathbf{i}$  and  $\mathbf{j}$  are the



**Figure D—1: The Laplacian template (ALC-1) and lag generation for application of the generalised covariance function.**

$j \backslash i$	$\mathbf{h}_1=(h,0)$ $\lambda_1=1$	$\mathbf{h}_2=(-h,0)$ $\lambda_2=1$	$\mathbf{h}_3=(0,h)$ $\lambda_3=1$	$\mathbf{h}_4=(0,-h)$ $\lambda_4=1$	$\mathbf{h}_5=(0,0)$ $\lambda_5=-4$
$\mathbf{h}_1=(h,0)$ $\lambda_1=1$	$K(0,0)=$ $K(0)$	$K(2h,0)=$ $K(2h)$	$K(h,-h)=$ $K(\sqrt{2}h)$	$K(h,h)=$ $K(\sqrt{2}h)$	$-4K(h,0)=$ $-4K(h)$
$\mathbf{h}_2=(-h,0)$ $\lambda_2=1$	$K(-2h,0)=$ $K(2h)$	$K(0,0)=$ $K(0)$	$K(-h,-h)=$ $K(\sqrt{2}h)$	$K(-h,h)=$ $K(\sqrt{2}h)$	$-4K(-h,0)=$ $-4K(h)$
$\mathbf{h}_3=(0,h)$ $\lambda_3=1$	$K(-h,h)=$ $K(\sqrt{2}h)$	$K(h,h)=$ $K(\sqrt{2}h)$	$K(0,0)=$ $K(0)$	$K(0,2h)=$ $K(2h)$	$-4K(0,h)=$ $-4K(h)$
$\mathbf{h}_4=(0,-h)$ $\lambda_4=1$	$K(-h,-h)=$ $K(\sqrt{2}h)$	$K(h,-h)=$ $K(\sqrt{2}h)$	$K(0,-2h)=$ $K(2h)$	$K(0,0)=$ $K(0)$	$-4K(0,-h)=$ $-4K(h)$
$\mathbf{h}_5=(0,0)$ $\lambda_5=-4$	$-4K(-h,0)=$ $-4K(h)$	$-4K(h,0)=$ $-4K(h)$	$-4K(0,-h)=$ $-4K(h)$	$-4K(0,h)=$ $-4K(h)$	$16K(0,0)=$ $16K(0)$

**Table D–1: Table of lag vectors (separations) hence terms in the expansion of  $\text{Var}(Z(\lambda))$  in Equation (D.35) and simplification to the isotropic case.**

unit vectors in two dimensions. This was presented earlier in Equations (5.8) and (5.16), and is represented graphically in Figure D—1, where each coefficient  $\lambda_i$  is attached to one of the points on the template. Given that the random function  $Z$  on which this template is applied is an Intrinsic Random Function of order one (IRF-1), the expected value of the template  $E(Z(\lambda))$  is zero and its variance is given by the generalised covariance  $K(\mathbf{h})$ , defined earlier in Equation (5.11) as

$$\text{Var}[Z(\lambda)] = E[Z(\lambda)^2] = \sum_{i,j} \lambda_i \lambda_j K(\mathbf{h}_j - \mathbf{h}_i) . \quad (\text{D.35})$$

Seeing that the covariance function above operates on all possible vectors  $\mathbf{h}_j - \mathbf{h}_i$  connecting points  $i$  to  $j$  in the template in Figure D—1, a table of such vectors is prepared, with a corresponding table of multiplied coefficients  $\lambda_i \lambda_j$ . Given that the covariance function is isotropic so that

$$K(\mathbf{h}) = K(\|\mathbf{h}\|) = K(h) , \quad (\text{D.36})$$

the same table may be re-expressed in terms of the magnitudes of the vector arguments only, resulting in Table D–1. Observing the symmetries in this table, it is then possible to write the sum of terms in equation (D.35) as

$$\begin{aligned}\text{Var}[Z(\cdot)] &= 20K(0) + 8[-4K(h)] + 8[K(\sqrt{2}h)] + 4[K(2h)] \\ &= 20K(0) - 32K(h) + 8K(\sqrt{2}h) + 4K(2h)\end{aligned}\quad (\text{D.37})$$

which result was earlier presented in Equation (5.21).

Chilès and Delfiner [84] (p. 264) present a canonical form for a generalised isotropic covariance function, to wit

$$K(h) = C_0\Delta(h) - b_1h + b_2h^2 \log h + b_3h^3, \quad (\text{D.38})$$

earlier presented in Equation (5.13), for which the coefficients are constrained to

$$C_0 \geq 0, \quad b_1 \geq 0, \quad b_3 \geq 0, \quad b_2 \geq -\frac{3}{2}\sqrt{b_1b_3}. \quad (\text{D.39})$$

Using Equation (D.38), the functions  $K(0)$ ,  $K(\sqrt{2}h)$  and  $K(2h)$  are evaluated – bearing in mind that the nugget effect  $\delta(h)$  is unity at  $h = 0$  and zero for  $h \neq 0$ ;

$$\begin{aligned}K(0) &= C_0\Delta(0) = C_0 \\ K(\sqrt{2}h) &= C_0\Delta(h) - \sqrt{2}b_1h + 2b_2h^2 \log(\sqrt{2}h) + 2\sqrt{2}b_3h^3 \\ K(2h) &= C_0\Delta(h) - 2b_1h + 4b_2h^2 \log(2h) + 8b_3h^3.\end{aligned}\quad (\text{D.40})$$

Rewriting Equation (D.37) as

$$\frac{\text{Var}[Z(\cdot)]}{20} = K(0) - \frac{8}{5}K(h) + \frac{2}{5}K(\sqrt{2}h) + \frac{1}{5}K(2h),$$

Equations (D.38) and (D.40) are substituted into the right hand side, and expanded so that

$$\begin{aligned}\frac{\text{Var}[Z(\cdot)]}{20} &= C_0 - \frac{8}{5}C_0\Delta(h) + \frac{8}{5}b_1h - \frac{8}{5}b_2h^2 \log h - \frac{8}{5}b_3h^3 + \dots \\ &\quad + \frac{2}{5}C_0\Delta(h) - \frac{2\sqrt{2}}{5}b_1h + \frac{4}{5}b_2h^2 \log(\sqrt{2}h) + \frac{4\sqrt{2}}{5}b_3h^3 + \dots \\ &\quad + \frac{1}{5}C_0\Delta(h) - \frac{2}{5}b_1h + \frac{4}{5}b_2h^2 \log(2h) + \frac{8}{5}b_3h^3\end{aligned}$$

$$\begin{aligned}
&= C_0 - C_0 \Delta(h) + \frac{6-2\sqrt{2}}{5} b_1 h + \frac{4\sqrt{2}}{5} b_3 h^3 - \frac{8}{5} b_2 h^2 \log h \dots \\
&\quad + \frac{4}{5} b_2 h^2 \log(\sqrt{2}) + \frac{4}{5} b_2 h^2 \log(h) + \frac{4}{5} b_2 h^2 \log(2) + \frac{4}{5} b_2 h^2 \log(h)
\end{aligned} \tag{D.41}$$

wherein the  $b_2 h^2 \log h$  terms cancel, leaving

$$\frac{\text{Var}[Z(\ )]}{20} = C_0 - C_0 \Delta(h) + \frac{6-2\sqrt{2}}{5} b_1 h + \frac{4\sqrt{2}}{5} b_3 h^3 + \frac{4}{5} b_2 h^2 \log(2\sqrt{2}) \tag{D.42}$$

the required relation in Equation (5.22).

# Appendix E - Program Notes

The code to support these notes can be found on the CD-R at the back of this thesis, in the directory `\code\fortran\...`. Along with the `*.f90` source files, there is also a MS visual studio workspace `krige.dsw` that defines makefiles and compilation configurations for the two projects, `krige` and `datagen`. These two projects are kept in the sub-directories `\code\fortran\datagen\...` and `\code\fortran\krige\...`, which in turn compile “Release” and “Debug” configurations of the executable in their respective `...\Release` and `...\Debug` sub-directories. Both projects make use of some or all of the modules defined by the source code in the parent directory `\code\fortran\...`. Modules and important subroutines and functions within them are described in the following subsections.

## E - 1. Style

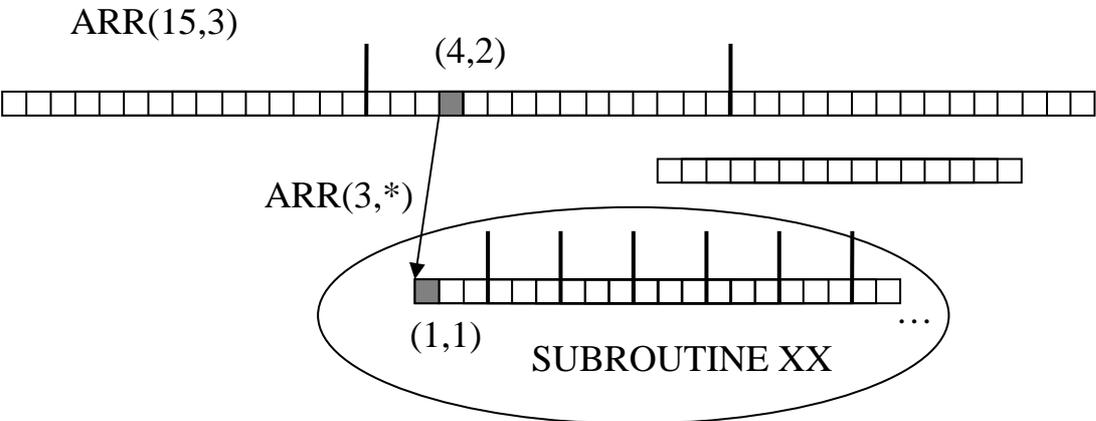
The code has been written and modified continuously over a number of years, in which my ideas about how to program have changed immensely – mostly as a result of my early messy attempts. As a result, the style is somewhat haphazard, and comprises many work-arounds and clean-ups. Certainly if I were to start again, it would be done differently! There are however some practices and conventions which are relatively constant throughout the code’s development and they are described below.

Full advantage is made of Fortran implicit typing – as is conventional variables starting with `A-H` or `O-Z` are `real*8` and variables starting with `I-N` are `integer*4`, except where a variable is specified otherwise. Whilst this is unfashionable, it is useful as it immediately identifies what variables are doing what in a code that largely employs `real*8` and `integer*4` types. In addition to this, array variables are exclusively written in uppercase, whereas single variables are always lowercase. This again highlights the variable’s functionality and saves confusion when there are functions next to arrays. Furthermore but more loosely, integer

addresses (pointers) usually start with the letters *i*, *j*, or *k*, whereas an integer counting a number of objects often starts with the letter *n*. Generally, loop counters in particular are denoted *i<sub>a</sub>* for the outermost loop, *i<sub>b</sub>* for the next one in, *i<sub>c</sub>* for the next after that and so on. Sometimes *i<sub>z</sub>* is used when an existing loop is nested inside another retrospectively.

Full and extensive use is made of another unfashionable feature of Fortran – the ambiguity of addressing whereby only first-byte address information is passed. Often a sub-element of an array or object *ARR*, say of size *ARR(15, 3)* will be passed into a subroutine by a first byte location in it, say *ARR(4, 2)*. The called subroutine then simply operates on the data as if it were an array starting at this address – for example *ARR* could be reshaped and internally represented as *ARR(2, \*)*, as illustrated in Figure E—1. The leftmost subscript varies fastest. This is very useful as it essentially allows Fortran routines to be objectified, without a formal framework for object oriented programming.

In general, rather limited use is made of allocatable arrays, especially in the early code. However, when space is allocated it is always deallocated upon exiting the subroutine. The only exception to this is when modules are initialised by some public initialisation routine, in which case space may be allocated in the initialisation, but is deallocated later when the module is closed by its exit routine. An extension to this rule applies to files, which are always closed by the subroutines that open them, or closed by the exit routine of a module where they are opened in the initialisation routine.



**Figure E—1: Passing a first-byte address and reshaping an array internal to a subroutine in Fortran.**

As the code grew more complex and sophisticated, the usefulness of general purpose routines to do common tasks was realised. These routines are kept in the module `speed`, which is used by practically all other modules and routines in the code. By and large they are self-explanatory with a little explanation included in a comment line for each. For example, `pcopy(A,B,ndim)` copies `ndim real*8` from the location starting at the first-byte address `A(*)` to the location at the address `B(*)`. Note that it does not matter here what shape `A` and `B` are in the calling routine, the arrays are reshaped on entry to the `speed` routines. These routines are all pure routines – they have no side effects and can be used anywhere.

## E - 2. Module dataset

There are a number of private variables in this module that are accessed by the subroutines in it. These variables define the arrangement of the data and the data clusters. The following maximum sizes are defined as parameters; `ntypes` defines the maximum number of datasets; `maxcat` defines the maximum number of clusters for each dataset; `memblok` defines the length in bytes of each data cluster. There are also two parameters defined in the module `universal` that are scoped via `use universal`. These parameters are `kdimax=10`; the maximum number of interpolating dimensions of the kriging algorithm, and `komax=4`; the maximum number of parallel datasets in a cokriging. These variables are used to define the parameter `maxdbuf` which sets the size of buffers holding raw nodal (coordinates and data) information.

Further common data structures are input in the initialisation subroutines `datini`. This subroutine opens the necessary binary files `*.kr1` and `*.kr2` specified by the argument `datfile`. If this argument consists of whitespace, is empty, or the files do not exist the program terminates with an error. Future provision has been made for default files `data.kr1` and `data.kr2` to be created and filled in interactively although this has not been programmed at this stage. Subroutine `datini` must be run before any other calls are made to module `dataset`, as it sets a number of private variables that are used throughout the module. These include `ndimd`, the dimensionality of the nodal data; `numrec` the number of records currently written to

in the \*.kr1 file; memax , the maximum number of nodes in the \*.kr1 file; memac , a reduced number of nodes that is used for better performance; kindsr , the number of datasets in the binary files; iniran , the initialising number of a pseudo-random number sequence.

The \*.kr1 and \*.kr2 files that are opened in datini and later closed in datexit are organised in the following way. The \*.kr1 file contains nodal data split into mutually exclusive, spatially local clusters stored one-to-a-record. Notably only the first four integer\*4 of the first record are used, to indicate 1) the dimensionality, 2) the number of datasets, 3) the record from where the cluster data is written sequentially and 4) the total number of records in the file. The raw spatial data is stored from record two and the the cluster data (which will be detailed shortly) is a kind of ‘postscript’. The \*.kr2 file contains essentially the same information with some modification. The spatial data is linearly detrended for each dataset and the corresponding detrended data is stored in the exact same locations as used in the \*.kr1 file, starting from the second record. The details of this detrending – the coefficients of the removed linear components – are stored in a ‘postscript’ starting necessarily at the same place as it did in the other file. These coefficients are stored in the private array COELINE . The only difference between the two files then, is what is stored and the total number of records. Thus the first three integers of the \*.kr2 file are the same, but the fourth may differ from the \*.kr1 file.

Another useful reference set in datini is the array LENREC , which contains the number of nodes in each record (as measured in bytes, integer\*4, real\*8, and nodes) and the corresponding sequential dataset number and cluster number. The array NELCAT contains the reverse table, recording that for a particular dataset and cluster in that dataset, there is a corresponding record number. These arrays are constructed in the subroutine records on the basis of what is read from the \*.kr1 file – where the basic cluster information LCAT and SEED is stored in a contiguous data structure containing real\*8 and integer\*4 base types. The two-dimensional integer array LCAT( : , : ) indexes the dataset in the second (slowest) index. The first (fastest) index lists the number of data-clusters (i.e. binary data records) in the dataset, followed by a list of numbers indicating how many nodes (coordinates and data) in each of those data clusters. Their record location in the \*.kr1 file is in the

same order as they are specified here, starting from the second record. A three-dimensional real\*8 array `SEED(:, :, :)` specifies the centroids of each of the data clusters. This information is used to speed searches for spatially local data. The centroid coordinates are stored in the first (fastest) subscript, the second subscript indexes the cluster and the third subscript indexes the data-type, in the same order as they appear in the subscripts of `LCAT`, hence the `*.kr1` and `*.kr2` files. The array `NUMPTOT` is also set in subroutine `records`, and contains the total number of nodes in each dataset in the raw data. There is also a small, and little used scratch space array `DATSMML` that is used to reserve space in the common block `DOOM`.

The subroutine `datexit` is invoked to close down usage of module `dataset` (a destructor in effect). It updates the current data structures in `LCAT` and `SEED` should they have changed (currently unnecessary, as they do not change), and then closes the files. It is assumed that every time the `*.kr1` file is modified somehow, the detrended file `*.kr2` is also updated. The principal routine to perform the detrending is called `detrend`, which detrends one binary `*.kr1` file to another on the basis of its file number. It is supported by the subroutines `regen`, `retrend` – which restores the detrended data to its prior state, and `lincom`, which computes the linear component at a point.

There are a number of clustering routines in module `dataset`. These are never used in the course of estimation (program `krige`), but they are used by the program `datagen`. In this case the point of access for the module `datagen` is subroutine `bunch` which accepts raw nodal data and then splits it up using the k-means algorithm as implemented in the subroutine `kmeans`. Significantly, it is assumed that the data can at least initially fit in memory – although it would not take too much modification to change this. As some of the downstream operations scale rather worse with size, this does make sense for the moment. This subroutine is called from `chunk`, which first checks if there is so little data that it may all fit in one cluster, and if it does not then initialises the k-means algorithm described previously. The initial groups for the clustering are decided by a simple octree algorithm `dom` and `dominit` called from `kseeds`, and the algorithm is implemented in `kmeans`. This is the iterative algorithm described in C - 1, but with some modifications for faster

operation. When it is called in `chunk`, `kmeans` reorders the points in `PTS` into their group orders and sets the corresponding descriptors for this ordering in `LCAT` and `SEED` (described previously). The array `STP` in `kmeans`' parameter list is merely passed as swap space for the node reordering.

The iterations occur in loop 11 of `kmeans`: firstly, an array `CDIS(:, :)` of values is calculated of the distance from each cluster centroid to every other centroid (it is zero-diagonal and symmetric). A corresponding index array `IND(:, :)` is also used to keep track of the order of these values (and hence the clusters they relate). The lists of distances to each cluster in `CDIS(:, i)` are sorted along with their cluster numbers `IND(:, i)` so that the lists `IND(:, i)` now refer to a ranking of the clusters in terms of proximity to cluster `i`. Each point is originally denoted as being in a particular cluster – `ITS(:)` is a list of the clusters that agglomerate each point in the original list `PTS(:)`. What then happens is that all the points are checked to see which of the `ntmp` closest clusters to the cluster they are already in is in fact now closest. The distance here is renormalised by the factors in `CAT(:)` (Equation (C.1)). To save operations, `ntmp` is crudely initialised by a random number generator and normalised to point usually to closer clusters. When the nearest cluster is ascertained, the point is agglomerated by perhaps a new cluster, this is recorded in `ITS(:)` and the list `JCAT` and the centres `CNTR(:, i)` are updated using the array of running averages `AVE(:, i)`. The last activities are performed by `recat` which also updates the factors `CAT(:)` and calculates `del` – a convergence norm. This convergence norm effectively measures the overall change in position of the recombined clusters: if it is small the iterations terminate, if not they continue. Subroutine `recat` also reports any cluster that has exceeded maximum length by passing out the cluster number `incat`. Should `incat` not be equal to zero, this cluster is split into two, and the arrays `JCAT`, `ITS`, `CAT` and `CNTR` are modified accordingly in subroutine `makecat`.

The points and nodal data in `PTS` are then reordered so that they reflect the clusters formed by their proximity to the converged centres finally in `CNTR`. Points are reordered such that the first `JCAT(2)` points belong to the first cluster around `CNTR(:, 1)`, the next `JCAT(3)` points belong to the second cluster around `CNTR(:, 2)`, and so on. The total number of clusters is stored in `JCAT(1)`. Each node's place in the list `PTS` prior to re-ordering is stored in `NADS`.

Module `dataset` also contains some routines to allow access to data and data structures. Subroutine `qdataset` returns the dimensionality of the datasets `ndim`, the number of datasets `kinds` and the number of clusters in each dataset in `NCAT(:)`. Subroutine `qcats` queries dataset about the `kinds`-th dataset. This routine returns the number of clusters `ncat` in this dataset, an array of the number of node points in each cluster `JCAT(:)`, and an array of the cluster centroids `CENT(:, :)` – in which spatial dimension is the fastest subscript. Subroutine `qvar` returns general statistics `av` and `va` on the `kin`-th dataset, as represented in the binary file with the file number `nfile`. The variable `av` is the average nodal value and the variable `va` is the total variance of this nodal value. There are also some general purpose routines that also allow more detailed access. Subroutine `recnum` takes `num` cluster numbers listed in `ICAT(:)` for a given dataset `ky` and over-writes the cluster numbers in `ICAT(:)` with their actual file record numbers. Subroutine `rawdata` returns nodal data from the file `nfile`. It runs through the list of `nrec` record numbers listed in `IREC(:)` from which to fetch information, and returns a close-packed list of coordinates with nodal values in `DAT`, consisting of all the nodal data in these records. If the records skip through a number of contiguous datasets (without returning to any of them) the subroutine returns `KINDAT(:)`; a list of `num` node-counts – that is, how many nodes of each type of dataset that are written to `DAT`. Crucially, `KINDAT` is not initialised to zero in this routine. This is to allow direct incrementing of the node-count if the routine is called multiple times, for example in a loop. When records are not type-contiguous subroutine `kindord` can be used to reconcatenate them in contiguous blocks.

Because module `dataset` has ready proximity to useful public variables such as `LCAT`, `LENREC` and of course the data, it makes sense to use it as a staging post for particular operations in the kriging algorithm. One such operation is the generation of spatial statistics in subroutine `dimple`; the part of structure identification described in Section IV - 4. This is mostly implemented in another module `sample`, but this module is accessible from here by the statement `use sample`, and the data preparation for this module is performed here. Subroutine `dimple` generates up to `max` spatial statistics between the entire spatial datasets `KIN(1)` and `KIN(2)`,

employing module `sample` for the splitting procedure outlined in Section IV - 4.1. The resulting list of lag vectors and statistics is returned in a close-packed list of coordinates and statistic values in `EXM(:, :)`, wherein the lag vector and bundled statistic are stored in the fastest subscript. The argument `max` is also used as an output to indicate how many such statistics are listed.

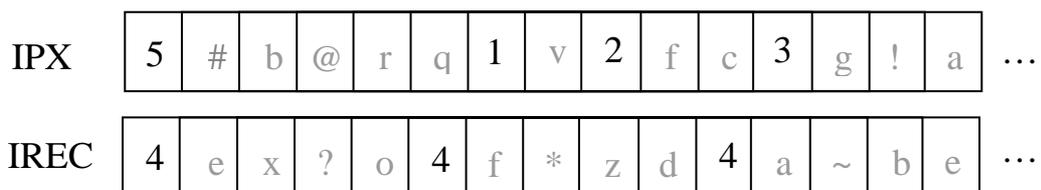
Subroutine `dimple` firstly checks for the existence of the datasets `KIN(1)` and `KIN(2)`. If such datasets exist, the routine `listcat` generates a pair of lists `LIST(:)` and `KIST(:)`, that identify pairs of record numbers to pull in – one pair `LIST(1)` and `KIST(1)` at a time – to generate the list of lag vectors and values specified by Equation (4.13). The subroutine `listcat` outputs `LIST`, `KIST` and the number of pairs `num`. It first creates a list `DIST(:)` of all distances between cluster centroids in dataset `kt1` and cluster centroids in `kt2`. This list is indexed by `IND(:)` which stores a unique integer for each entry, and when `DIST` is rearranged in order of increasing separation distance, so is `IND`. Loop 15 in `listcat` then goes through the lists `IND` and `DIST`, and works out how many (more) points in the list (4.13) will ultimately arise by using each successive pair of data-records. When this loop hits a maximum `npmax` or exhausts all items in `IND`, it breaks and `LIST` and `KIST` are populated up to their maximum size. What then follows is a randomisation of the record number order, and then the lists are ready.

In the foregoing discussion a maximum size of the lag vector list (4.13) is set. This argument is passed in from `dimple` when `listcat` is called, and it is determined using the public parameter `maxfile` in module `sample`. It was mentioned that the lag vector list is swapped out to disk because of its size, and this parameter determines the maximum size of the file. In loop 13 of `dimple` the raw data is fetched for the pair of records `IRA(ia)` and `IRB(ia)`, and it is input to the subroutine `xgen` which writes the lag vector list to disk. Subroutine `xgen` is resident in module `sample`, as is subroutine `flbuf` which simply empties a temporary buffer in this module. Subroutine `modtree`, also a `sample` routine, performs the splitting of the lag vector list as detailed in IV - 4.1 and calculates the appropriate statistics. Finally, the subroutine `sampexi` cleans up all files and values in module `sample`, so that it may possibly be used again. Note that in loop 13, the call to `rawdata` specifies that the

detrended data in the \*.kr2 file should be fetched – indeed it is a simple matter to change this to any other detrended file, or even use the original \*.kr1 file.

The other important set of routines in module `dataset` organise estimation of multiple points over multiple local data windows. The principal routine to organise this is subroutine `master`. This routine takes a list `PX(:, :)` of `npts` points at which estimations are desired and a list of `nkin` datasets listed in `KIN`, which are to support the estimation of the primary variable `KIN(1)`. Results are returned in `RES(:, :)` where `RES(1, :)` is the estimated quantity, `RES(2, :)` is the variance of the estimated quantity and `RES(3, :)` is the flexible output value mentioned of the `krigout.txt` file described in B - 1.1. Subroutine `master` fetches data on the scale of clusters, and writes the data to a buffer `DATP(:)` where it is used to support estimation in module `antekrige`.

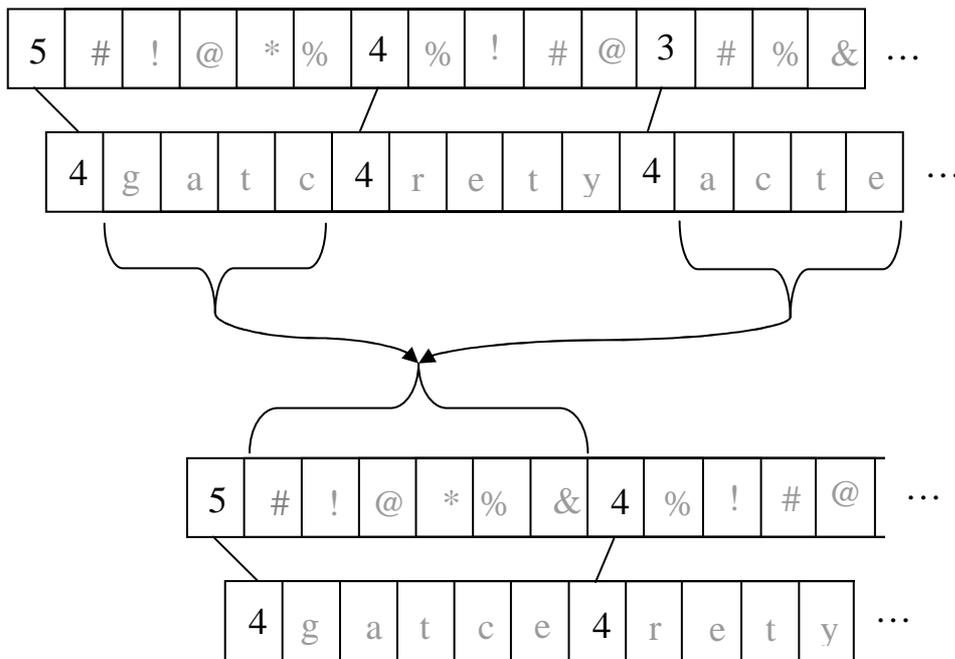
As described in IV - 6, a search is made for the nearest  $2^{\text{nddimd}}$  clusters to each estimation point in `PX`. This is performed by subroutine `meercat`. This routine scans through the list of `npts` nodes `PTS`, and finds the nearest  $2^{\text{nddimd}}$  clusters (or less if there are not that many clusters anyway) for each dataset in the cokriging. It stores their record numbers in the array `IND`, which it passes to subroutine `padup` which is resident in module `sample`. Subroutine `padup` then updates the irregularly shaped arrays, `IREC` and `IPX`. These arrays convey unstructured lists of point or record addresses (the point address is its position in the original list of nodes). Sets of interpolation points in `IPX` correspond sequentially to record numbers in `IREC` needed to estimate them, but the sets are not necessarily the same length all the time. Figure E—2 demonstrates the organisation of these arrays. It should be noted however that the stride for `IREC` is always the same, which is why in `padup` it is dimension



**Figure E—2: Unstructured lists in subroutine `meercat` – note that `IREC` has an even stride.**

IREC(nbit+1,\*). Other input parameters for padup include max, which indicates a maximum number points in an individual set of points; nbit the number of addresses in IND and ip the address of the one point that will require all of the pointed items in IND for its estimation.

After meercat is called in master, the basic lists IPX and IREC are established – there are ngp sets of points indicated in IPX requiring corresponding sets of records indicated in IREC. These lists are then merged as was outlined in Section IV - 6, using the subroutine mergsom (resident in module sample). This takes two corresponding unstructured lists (Figure E—2) IND and MEM (previously IPX and IREC), and examines the sets of addresses in MEM for similarity (Figure IV—13). The most similar sets are merged as demonstrated in Figure E—3, with the proviso that the final set in MEM does not exceed max members, and the parallel merge operation in IND does not result in a set exceeding mox members. Note that back in master, these maxima are set as maxr and maxp; the maximum number of records that can fit in the buffer DATP, and the maximum number of points that will fit into integer pointer buffers later on – maxptb. Note that maxptb is a public parameter that resides in module antekrige, and like the routines padup and mergsom, it is made accessible



**Figure E—3: Merging groups in a pair of unstructured lists on the basis of similarity of groups in the bottom list.**

by the use `antekrige` statement in `master`.

Once `mergsom` has been called in `master`, the `ngp` sets in the unstructured lists `IPX` and `IREC` are ready. These lists are parsed, one set at a time in loop 11 of `master`. Firstly the list of record numbers is ordered by `kindord` so that data is of contiguous type (indeed, it should be anyway at this stage), as specified by the parameters `KIN` and `nkin`. Then the data is fetched to the buffer `DATP` by subroutine `catdat`. Subroutine `catdat` takes a list of `nrec` records `IREC` and loads the data to the dummy array `DATC(ndimd+intrmod+1,*)`. This data is close-packed nodal data; the first `ndimd real*8` are the nodal coordinates, the next `real*8` is the nodal data value at this point, and the next `intrmod real*8` are the nodal values of the interpolated universal kriging drift functions. Subroutine `catdat` collects the nodal data for estimation from the `*.kr1` file (unit 51), and collects the interpolated modes from the `*.km` file (unit 80). As shall be explained later, this file is of identical structure to the `*.kr1` file except that the file record length is changed to accommodate the absence of nodal coordinates, but now also the presence of `intrmod` interpolated (or even just non-polynomial) drift function bases at those nodes. When it fetches the data from the `*.kr1` file, the subroutine `rawdata` also returns an array `NDAT(:)` which is a list of how many nodes of each dataset are actually in the array `DATC`. This array is subsequently passed into the subroutine `antinit` (resident in module `antekrige`) which initialises module `antekrige` for the purposes of performing an estimation.

Now that module `antekrige` is initialised, and the local data for the estimation of points indicated in `PX` has been fetched, we proceed to estimation by calling subroutine `enqueue` (resident in module `antekrige`). This routine takes the list of points `PX` and the addresses in it at which to interpolate `IPX(iat)`, and finally performs the estimations, filling in the corresponding addresses in `RES`. It is given access to the known data in the buffer `DATP`, but it still needs to know what the interpolated drift function values are at each of the interpolated points. This information is passed in by `PRE(:)`. Although `PRE` is officially of dimension `PRE(*)`, it is effectively of dimension `PRE(intrmod,*)` but merely passes through `master` without reference. When the estimates are produced, module `antekrige` is de-

initialised by `antexit`, the pointers in `IREC` and `IPX` are moved, and estimation moves to the next group in the lists, until all such groups have been covered.

If interpolated drift functions are used in the estimation, interpolated values at each of the nodal data and at the estimation points will be required. As previously mentioned the interpolated drift function for each of the nodal data is communicated via the `*.km` file. However, the interpolated function at the estimation points is passed into `master` with those points in `PRE`. This array is always constructed beforehand by the subroutine `preintr`. This subroutine takes a list of `npts` nodal coordinates in `PX` and interpolates each of the `nkin` datasets listed in `KIN` in turn, at these coordinates. This is performed after first retrieving the covariance model resident in module `krige`, which is stored in `VAGM`, `SILS` and `OFFS`. The number of interpolated modes `intrmod` in module `interpuk` is then temporarily set to 0, and the variables `kuesst` and `kruntyp` in `datmisc` are temporarily set to 2 and 1 respectively. In loop 11, the estimations for each of the `nkin` interpolated drift functions are performed and the relevant results in `BUF(1,:)` are repacked into the dummy argument `RES`. The value `RES(1,i)` is the interpolated drift function constructed from `NKIN(1)` at the coordinates `PX(:,i)`, the value `RES(2,i)` is the interpolated drift function constructed from `NKIN(2)` at the coordinates `PX(:,i)`, and so on. After loop 11, the values of `kruntyp`, `kuesst` and `intrmod` are restored to their former values and the former covariance model, now in `VAGM`, `SILS` and `OFFS` is loaded back into module `krige`.

Other routines in `dataset` facilitate other specific functionality. Subroutine `mirror` constructs the `*.km` file that implements the option to include interpolated drift functions in the drift model, as specified by the keyword `mirror` in `krigin.txt`. This file has been opened in subroutine `modini` – called by `datini`, and its contents are set in this subroutine, `mirror`. This routine takes a parameter `NKIN(:)` which contains the `kinds` datasets involved in the cokriging at whose nodes interpolated drift functions (modes) will be required. The other parameter `MODES(:)` contains the `kint` datasets that will have to be interpolated – these are the datasets

listed after the keyword `mirror` in `krigin.txt`. Subroutine `mirror` firstly compiles a list `IREC` of the records in the `*.kr1` containing the nodes at which interpolated modes are to be calculated. What this routine does is to write the interpolated modes to these same record numbers, but in the `*.km` file instead. Thus when it comes to estimation – as has been explained previously, the relevant interpolated data will be fetched from the same record locations as the raw data, but from a different file.

In loop 15, raw nodal data from the records in `IREC` is read into the array `BUFF(:)`, eight records at a time. As we are not interested in the actual nodal values, these are discarded upon repacking the array using the `speed` routine `rawdata`. This step effectively reshapes `BUFF` from `BUFF(ndimd+1,*)` to `BUFF(ndimd,*)` – note that `BUFF` is represented locally as a singly subscripted array, but in reality it is much more flexible. It is assumed that the covariance models have already been determined for the interpolation of the nodal data indicated by `MODES`, and that they reside in the `public` arrays `VARIO` and `COVAR` of module `interpuk`; a data-only module. These models are crude single variogram models, although indeed for the purposes of the exercise even a nominal model could be used, for example a spherical, isotropic covariance with range and sill at unity. In the nested loop 17, these models are loaded for each of the interpolated drift functions in `MODES` in turn, and `master` is called to make estimations at the nodal coordinates loaded in `BUFF`. These estimates are then repacked (to remove the variance and the flexible output) into the array `DAT`. After all estimations in nested loop 19, this array is then written to the same eight records that the estimation nodal coordinates came from, but in the `*.km` file. The data packing in this file and in `DAT` is simply `DAT(intrmod,*)`.

The `cross-validation` mode of operation is implemented in the subroutine `xvalid8`. It has been described already in Section B - 1.3. Subroutine `xvalid8` takes as input the kinds datasets for cokriging in `NKIN` and the ‘modes’ datasets from which drift functions are to be interpolated in `MIR`. This information specifies the estimation problem for the cross-validation exercise. The binary files that contain the original information are copied to new binary files, the leave-one-out exercise is performed at all known data points in them, and finally the results of cross-validation are reported in the new binaries, and also in a list in the file `krigout.txt`.

Firstly, the program makes use of the system command-line function `system` to `copy` the `*.kr1` file to a parallel file `*.kv1`. This file is opened along with a `*.kv2` file, whose purpose will be analogous to the `*.kr2` file. It then reads in the data-clusters of the primary variable `NKIN(1)`, eight records at a time. It repacks the data with `repack` effectively reshaping it from `BUFF(ndimd+1,*)` to `BUFF(ndimd,*)`, discarding the nodal value in the process and retaining only the nodal coordinates. Subroutine `preintr` prepares the array `PRE` from these coordinates, which is passed with the coordinates in `BUFF` and other necessary information into `master`. The behaviour of the actual estimation in module `antekrige` will be different under the circumstance of cross-validation, and the returned matrix `RES` in subroutine `xvalid8` will include the following information. As before the array is of shape `RES(3,*)`, but now the first `real*8` is the estimated value at the nodal coordinates with the nodal data there *left out*; the second `real*8` is the standard deviation corresponding with this estimate; and the third `real*8` is the difference between the *original* nodal data and the estimate as normalised by the previous standard deviation (as discussed in B - 1.3). Once `RES` has been produced, the data in `BUFF` is padded out by an extra space using `repack` and the values `RES(3,:)` are written to `BUFF(ndimd+1,:)` in nested loop 13. Also in loop 13, all of the cross-validation results in `RES` are written out to `krigout.txt` as described previously.

As the number of data-clusters is unlikely to be divisible by eight, the code between `continue` statements 11 and 19 in `xvalid8` performs exactly the same operations as described above, but for the remaining data-clusters. Indeed, this code is a candidate for a clean-up: it can be implemented in a single `if` statement in the main loop 11. Subsequent to all cross-validations, `xvalid8` detrends the `*.kv1` file to the `*.kv2` file. In this way, a regular raster or contour plot may easily be prepared of normalised cross-validation interval over the estimation domain – by simply renaming the `*.kvX` files as `*.krX` files and performing a univariate kriging on the relevant variable. Finally, all files that are opened in subroutine `xvalid8` are closed.

## E - 3. Module antekrige

Some parts of module `antekrige` have been touched upon already – these `public` routines will be detailed first, followed by the `private` routines that support them. Like most modules, `antekrige` has ‘constructor’ and ‘destructor’ type subroutines. Subroutine `antinit` is called before any of the estimation routines are used, to set the `private` integers `ndimd`, `kin`, `maxdata` and `ndstr`: these are respectively the data dimensionality, the number of datasets, the maximum number of nodes in any one of these, and the stride of the data in `DAT`. The array `DAT(ndstr,:)` that resides in module `datmisc`, is also allocated in `antinit`. This array holds all the raw data for the estimation; nodal coordinates, nodal values, and the nodal values of the interpolated drift functions. The `private` integer array `NDAT(:)` contains the number of nodes each type of data, listed contiguously and sequentially in `DAT`; it is also set in `antinit`. Note that `antekrige` need not be initialised to use the subroutines `padup` and `mergsom` – these are ‘pure’ functions. Additionally there is a subroutine `antexit` that deallocates the array `DAT` in module `datmisc`.

As described in the previous section subroutine `padup` takes a list of `nbit` addresses in `IND` and checks if there is an identical list in a set of lists `IREC`. If there is, it adds the associated address `ip` to the corresponding set of associated addresses in `IPX` – which contains lists of associated addresses. These address lists are arranged as was explained in Figures E—2 and E—3. If there is no identical list `IND`, or the set of associated addresses exceeds `max`, a new list `IND` is added to the end of `IREC` and a list consisting of one associated point `ip` is added to `IPX`. In this subroutine the lists of points in `IPX` are parsed by a pointer `iat`, one group at a time in loop 11. In this loop the maximum condition is checked, and if it is not respected control passes to 10 `continue`. If it is respected the similarity of the list at `IREC(ia,:)` with `IND(:)` is checked in loop 17 – and any dissimilarity causes a jump to 10 `continue`. At 10 `continue`, `iat` is incremented before loop 11 cycles again. Should the code before 10 `continue` be executed, the address `ip` is added to the list pointed by `IPX(iat)` – and this list is updated accordingly (half of it must be shuffled forwards to

accommodate the new point). Once this is done, `padup` returns with `goto 99`. If there are no similar lists and loop 11 is never broken in this manner, `IND` is copied to a new list on the end of `IREC` and a list of the one point `ip` is created at the end of `IPX`. Note that for this subroutine `padup`, `IREC` is assumed to be of constant stride (`nbit+1`), but because of the circumstances in which it is called, this is guaranteed. If it were not constant stride, a second pointer would be required.

Similar to subroutine `padup` is subroutine `mergsom`. Subroutine `mergsom` takes `nsoc` unstructured lists in `IND` and `MEM`, and merges the lists in `MEM` on the basis of similarity. When two lists are merged in `MEM`, the sequentially corresponding lists are also merged in `IND` – up to a maximum list-length in `IND` and `MEM` of `max` and `max`, respectively. The merges continue in loop 11 until these maxima are reached and no more merges may be made; then control passes outside the loop to 12 `continue`. The similarity between two lists of addresses is measured by function `simet` – as articulated in Equation (4.54). This function takes `nn` integers in `LIST` and `kk` integers in `KIST`, returns the similarity metric between zero and one described in (4.54) and the size of the set if `LIST` and `KIST` were to be merged in the parameter `new`.

Back in subroutine `mergsom`, the nested loops 13 and 15 compare each list of points in `MEM` to every other list of points in `MEM`, and constructs a square matrix (without diagonals) `PER(:, :)` of similarity metrics using `simet`. Note that in these loops 13 and 15, the  $i_b^{\text{th}}$  unstructured list in `MEM` is pointed by `ip`, the  $i_b^{\text{th}}$  list in `IND` is pointed by `jp`, the  $i_c^{\text{th}}$  list in `MEM` is pointed by `iq`, and the  $i_c^{\text{th}}$  list in `IND` is pointed by `jq`. High similarity in `PER(:, :)` is indicated by values close to one and low similarity by values close to zero. If a particular combination `i-merge-with-j` is voided by the maximum list size requirements, `PER(i, j)` is set to `-1.0`. In nested loops 19 and 17, the two most similar sets `icb` and `ibb` are found. If no legitimate sets are found the subroutine returns. Otherwise, subroutine `remoue` is called on `MEM` to merge the  $i_{bb}^{\text{th}}$  and the  $i_{cb}^{\text{th}}$  lists and reconstruct `MEM` – the subroutine works on `IND` in a similar fashion. The merged sets are always stuck on the end of the unstructured lists and so the pointers to them, `ip` and `jp` are also passed out, although `nsoc` is as yet unchanged. The subroutine `remoo` then updates the matrix `PER` by removing the  $i_{bb}^{\text{th}}$  and  $i_{cb}^{\text{th}}$  columns and rows, and `nsoc` is then decremented. Note

that there is an extra column and row in `PER` that now must be ascertained – the new merged set at the end. This is accommodated by setting the loop 13 limit `ns` to `nsoc`, so that it is evaluated on the next loop.

The primary purpose of module `antekrige` is to act as a staging point for estimation that is removed from file I/O activities. Subroutine `enqueue` is the point of access for this purpose. This routine takes the entire list of interpolation points `PTS(ndimd,*)`, interpolated drift functions at these points `PIN`, pointers to those `npts` points requiring estimation `IPX`, and the raw nodal data from the surrounding data-clusters in `DATP`. It passes these values to `crunch` which produces estimates at the addresses pointed by `IPX` in `RES(3,*)`. Subroutine `enqueue` determines the final data that will be used to construct the covariance matrices – it performs the sub-search mentioned in IV - 6 that selects a group of the `near` closest points in the proximal clusters. It runs through each point in `PTS` pointed by `ip=IPX(ia)` in loop 11, and for each cokriging variable in loop 13; finds the distance `DIS(ic)` between `PTS(:,ip)` and nodal points of sequential type `ib` in `DATP`. The array `IND` stores the locations of these data in `DATP`, and it is extended for each dataset in loop 13. If there are not enough points – `NDAT(ib)` is less than `near`, all of the points are used in estimation. Otherwise, the `speed` subroutine `smallh` is called to find the smallest `near` values in `DIS` and the corresponding entries in `IND`, which again indicates the points to use in the final estimation. After loop 13, `IND` contains the addresses of `it` points in `DATP` that are to be used to estimate the point `ip` (or `IPX(ia)`). Finally in loop 11, the unstructured lists in `NPX` and `IDAT` are appended or modified by `ip` and `IND` to form the final sets of points: `NPX` will contain unstructured lists of addresses in `PTS` that correspond sequentially to `IDAT` containing unstructured lists of addresses in `DATP` to estimate them. There was originally an additional step to call `mergsom` on these lists afterwards, but for problems typical to this thesis this was not found to be advantageous. Once the data windows have been selected above, subroutine `crunch` is called to make the estimates.

Subroutine `crunch` runs through the `ngp` unstructured, paired, lists in `NPX` and `IDAT` in loop 17. In nested loop 19 it copies the estimation coordinates in `PTS(:,ip)`, followed by the `intrmod` interpolated drift functions at these points in `PIN(:,ip)` to

the local buffer `PX`. It then prepares the nodal data upon which to base the estimation in another local buffer `RE`, copying it over in contiguous chunks from the various addresses in `DATP` specified in `IDAT`. At the same time, it counts how many nodes (and associated information) of each dataset have been copied, and enters this in the public array `NTYDAT(:)` resident in module `datmisc`. A computed `goto` then pursues one of two options at `42 continue` or `44 continue`, depending on the public integer `kruntyp` (also resident in `datmisc`). Should `kruntyp=1` or `3`, corresponding to raster interpolation or structure identification, then subroutine `krigery` in module `krige` is called upon to make the interpolations, with the input data now in the required formats in `PX` and `RE`. If `kruntyp=2`, corresponding to cross-validation, the subroutine `xvalid` in `krige` is called instead – again with the appropriate data in `PX` and `RE`. Whichever course is charted, the estimates in `RE` effectively of shape `RE(3,*)` are then copied back to their proper in locations `RES(:,ip)`.

## E - 4. Module `inter`

This module orchestrates many of the functions of the other modules – and coordinates them to achieve a specific functionality. Once again it is initialised by a subroutine `intinit`, but only to set the dimensionality – a private integer `ndimd`.

Subroutine `taste` takes a list of `nkin` datasets `NK` (for example, those that appear after the keyword `kriging variables` in `krigin.txt`) and constructs a licit covariance model to describe the auto- and cross-covariant spatial relationships. The first part of the routine, in loops `11` and `13` implements the identification of basic anisotropies and models, described in Section IV - 5.1. Each possible pairing of the variables in `NK(:)` is identified in these loops, and the dataset routine `dimple` is called to append a list of spatial statistics in the buffer `VA(:)`. No more than `nreq` statistics (heuristically chosen to equal `400*ndimd`, earlier in the routine) are produced. After `dimple` is called, there will be `nv` new spatial statistics in `VA`. This array is effectively of shape `VA(ndimd+1, :)` – the data is close-packed with the first `ndimd` `real*8` of the fastest index describing the lag vector, and the remaining `real*8` describing the spatial statistic, which is normalised between zero and one for auto-

covariant statistics, or one and minus one for cross-covariant statistics. Note that `kat` points to the start of the list of spatial statistics in each pass of the inner loop. If a cross-covariant statistic has been calculated, the statistics are centred around a central peak and the offset `HOF` is calculated by subroutine `centre` (resident in module `sample`). A logical value `lauto` is also set to `.false.` if this is the case – it is `.true.` otherwise. This value is passed with the spatial statistics in `VA` to subroutine `varigen` which finds the basic anisotropies and models described in Section IV - 5.1 and appends them to a list of models and anisotropy matrices in `VUM`. Note that because of the symmetry of the transform matrices  $\mathbf{T}^i$  in (3.28), there is always enough room to pack in information about the model type, along with the coefficients in an `ndimd×ndimd` array. Thus the effective dimensionality of this list is `VUM(ndimd,ndimd,:)`.

Subroutine `varigen` also removes outliers from the list starting at `VA(kat)`, thus it may modify the value `nv`. This pruned list is retained, and the number of statistics is recorded in `NSAM(:)` – in the same order as they are calculated in the loops 11 and 13. It is worthwhile noting that this sort of ordering for auto- and cross-covariant components or statistics, is the norm throughout the program for close-packed lists relating to symmetric structures where only one side is stored; or even symmetric matrices wherein only one side is stored. Usually, the outer loop indexes `ia=1,nsquare` and the inner loop indexes `ib=1,ia`. Furthermore, usually the faster subscript is chosen as the inner loop (when only the upper half of an array need be accessed). Therefore, the close-packed list `HOFF` is stored in this way and contains the offset information for the spatial statistics. The various pointers in these lists; `kat`, `k`, `nmod`, `ntmp`, `nat` and `koff` are all updated at the end of loop 13.

The rest of subroutine `taste` is used to perform the coregionalisation fitting described in Sections IV - 5.2 and IV - 5.3. Subroutine `vrload` is a subroutine resident in module `krige` that loads the `nmod` variogram models in listed in `VUM` into the data structures describing the covariance model in module `krige`. Having established these basic models, the subroutine `koreg` (also in `krige`) is called to determine the coefficients  $s^i$  in Equation (3.69) (which are stored in module `krige`), considering the spatial statistics in `VA`, arranged in blocks of `NSAM(:)` statistics. The number of datasets in the cokriging is also passed in to module `krige`

at this point. There are also three other parameters that are determined in the prior call to `nugmod1`. These are the arrays `VAR`, `UNS` and `MASK`, and they determine the total variance of each auto-covariant covariance model and the way in which the nugget effect is to be fitted to the statistics (see Section IV - 5.3). The array `VAR(:, :)` itself is an array of maximum total variance values for the auto and cross-covariance functions. It is filled in completely in `nugmod1`, but the variances are calculated along the leading diagonal in loop 11 by a call to the routine `qvar` (in module `dataset`).

Subroutine `nugmod1` uses the variances along the leading diagonal of the matrix `VAR` and checks the input file `krigin.txt` to prepare the rest of the matrix `VAR` and the matrices `UNS` and `MASK`. In loop 11 it scans `krigin.txt` for the keyword `unstructured`. If this is found `nugmod1` then reads the desired inunstructured variance for the auto-covariant structures into the local array `UNST(:)`. In loops 15 and 17 it then takes the square root of the leading diagonal of `VAR` and fills in the off-diagonal components with the maximum cross-covariance indicated in Equation (3.72). In loop 19, the leading diagonal `VAR(ia, ia)` is returned to its former state and the array `UNS` is also set. This array is a matrix of maximum structured variance for cross and auto-covariance functions. In the cross-covariant components `UNS(i, j)` it is always equal to `VAR(i, j)`, but it may vary in the auto-covariant components if there exists a positive value `UNST(ia, ia)` (the unstructured variance is to be set) and it does not exceed the total variance in `VAR(ia, ia)`. Also set is `MASK(ia, ia)` which is set to 1 for those covariance models whose unstructured variance is set at a particular level. As it stands, the unstructured variance is never set for the cross-covariant components – `MASK(i, j)` is always 0. This is because it is thought that an estimate of short-scale cross-variability would be almost impossible to find. The cross-covariant parts of these matrices were however retained in module `krige` to allow greatest flexibility in the future.

After `koreg` has been called in `taste`, the coefficients in Equation (3.69) are all set, and there is a complete and licit covariance model in module `krige` for all subsequent multivariate estimation problems. Note that the spatial statistics were all centred earlier. Removing any cross-covariant offset makes the covariance function fitting problem simpler, and the effects of the offset can still be accounted for in the covariance model in `krige`. This is done by calling `cregoffset` (resident in `krige`)

to set the offsets for the cross-covariant structures recorded in `HOFF`. However the array `HOFF` is set entirely to zero beforehand, as it is still difficult to estimate the offset with any great certainty in `centre` – although at least it affords a better model in `koreg`. It may be useful to implement this as a flexible option in `krigin.txt`, in the future.

Finally in `taste`, the covariance model coefficients and variances are output to `krigdia.txt`, and the spatial statistics are written to `krigrasta.txt` for future reference.

Subroutine `varigen` takes a list of `nex` spatial statistics in the parameter `EXM(ndimd+1,*)` and generates a maximum of `max` basic spatial models and transformation matrices in the parameter `VMOD(ndimd,ndimd,*)` – the step described in Section IV - 5.1. The parameter `lauto` is passed in `.true.` if the statistics in `EXM` are auto-covariant, `.false.` otherwise. The statistics in `EXM` are first passed into subroutine `inlier` (in module `sample`) to remove outlying or aberrant statistics – although, the way in which the statistics are generated should guard against these at least at short scales. The main business of this subroutine is – in loops 13, 15, and 17, to fit all possible combinations of the three models (spherical, exponential and Gaussian) to the points in `EXM`. The fitting itself is performed by subroutine `trane` (in module `krige`) and the models for this fitting are specified in the integer array `MODO`. As described in Section IV - 5.1, models for negative covariance are treated separately from models for positive covariance, which is why `lauto` is required. The array is shaped `MODO(2,3)`, and in it are integers indicating how many models of a given sort are required. The slowest subscript indexes the models; respectively spherical, exponential and gaussian, and the fast (first) subscript indexes whether or not the model is positive or negative. Hence if all entries in `MODO` were 1, it would indicate that one of each model, both positive and negative, are to be used – presumably for a cross-covariant structure.

Array `MODO` is initially set to 0, and its entries are cycled in the looping structure in 13, 15 and 17, which covers all combinations of three structures. If a cross-covariant model is required, the components in `MODO(2,:)` mirror those in

`MODO(1, :)` (they will be zero otherwise). Admittedly, the looping structure is not obviously extensible to more than three types of models – for this, a more general piece of code may need to be written. The subroutine `trane` is called with the particular specification in `MODO` and the statistics `EXM`. It passes out the models in `TRAN`, and the multiplicative coefficients relating to these models in this case in `CONT`. The array `CONT` is not used – it is merely a dummy variable. The basic models `TRAN` are then passed into subroutine `checka`, which checks them for suitability. If a model is very ill-conditioned, or not positive-definite it may be modified or rejected outright. Acceptable models are retained in the list `TRAN`, and `nmod` is modified accordingly. Finally, if there is enough room left in the dummy variable `VMOD` (checked by reference to `max`) the models listed in `TRANS` are appended from `VMOD(1, 1, mod)` and the counter `mod` for the total models discovered is incremented. This counter is ahead by one, as FORTRAN indexes arrays from one – so at the end it is decremented by one and assigned to parameter `max`. This parameter now indicates the number of models discovered by `varigen`.

For simplicity, this part of the program is currently set to run through all combinations of one model – in other words it performs three fittings, one for each of the models. The outermost loop limit `nt=modtrane` which is a parameter currently set to 1. This was not originally the case, but it was tried to save some computational effort – and was found to make little appreciable difference on the quality of the fit. Indeed, often the cruder model and looser fit leads to a more robust set of base models.

Subroutine `checka` takes a close packed list of models in `TRAN` and checks that they are indeed positive definite, and are reasonably well-conditioned. This routine is a little heuristic, but seems to ensure decent stability. A little must be known of the structure of the close-packed models in `TRAN`: this is of shape `TRAN(ndimd, ndimd, *)` and the `ndimd×ndimd` blocks contain the transformation coefficients in the ‘top’ diagonal (`TRAN(1, 1, :)`, `TRAN(1, 2, :)`, `TRAN(2, 2, :)`, `TRAN(1, 3, :)`, `TRAN(2, 3, :)`, `TRAN(3, 3, :)` etc.). This means the space `TRAN(2, 1, :)` is always free to store information – currently it is equivalenced as `dopp`, which represents two `integer*4`, so that the first integer is the model type; 1

for spherical, 2 for exponential and 3 for Gaussian. Loop 11 of subroutine `checka` runs through each of the models in the list in turn. The acceptance criteria for a badly conditioned matrix is set by `crit` – a ratio of eigenvalues that is less than 20.0 is considered acceptable, except for Gaussian models which have a stricter limit of 5.0. Subsequently, the negative covariance models are made positive for the purposes of the coregionalisation, which deals with just positive definite models. The model under examination `TRAN(:, :, ia)` is copied to `TEMP` and `TRAN2` – this is because the speed routines to check for positive definiteness and condition number (respectively `ppos` and `pconda`) modify the input matrices. These copies are tested with `ppos` and `pconda`. If `ppos` does not return `ind==1`, the model fails automatically and is not added to the list at `TRAN(1,1,mod)`. If the condition number is greater than the acceptable criterion `crit`, the model `TRAN(:, :, ia)` also fails, unless the logical\*4 `let` is `.true.`. In this case, some attempt is made to make the model more positive definite. This is done by identifying the largest leading diagonal term in `TRAN(:, :, ia)` in loop 21 and adding 10% of this value to all the leading diagonal terms in loop 17. The subroutine `pconda` is called again, and if the condition number `cond` is acceptable, the model is retained. The number of accepted models `mod` updates `nmod`, which is passed out of `checka`.

Subroutine `strucmode` is used to generate each of the covariance models for the interpolated drift functions. The `nkin` datasets from which these models are produced are listed in `KIN(*)`, and appear after the keyword `mirror` in the input file `krigin.txt`. In loop 11 of `strucmode`, subroutine `taste` is called on each dataset in `KIN` to produce `nkin` separate univariate covariance models for them (the second argument is 1). The model is then retrieved from module `krige` by the subroutine `getmod1`, and stored in a close packed list in `VARIO`, `COVAR` and `INDX`. Note that the offset vectors are returned in `HDUM`, which is discarded by virtue of the fact that these are univariate models, and the offset is therefore zero. The arrays `VARIO`, `COVAR` and `INDX` exist as public arrays in module `interpuk` – subroutine `strucmode` can use them because of the statement `use interpuk`. The array `INDX(:)` is a list of the number of basic models in each covariance model. These basic models occupy contiguous blocks in the lists, `VARIO` and `COVAR`. Note that `VARIO(:)` is a close-

packed list – effectively, it is of dimension `VARIO(ndimd,ndimd,:)`. Each `ndimd×ndimd` block consists of a model and a transformation matrix. If for example `INDX(1)=3` and `INDX(2)=1` then the first three such blocks correspond to the model for `KIN(1)` and the next one corresponds to `KIN(2)`. The array `COVAR` is effectively a list of coefficients multiplying each of the models listed in `VARIO`. This linear combination forms the covariance model for the univariate case. In `strucmode`, these lists are all appended by writing to these arrays from the pointer locations `iat` and `jat`.

Subroutine `switch` is called in the main program block `krigedemon`. This routine sets the program about a particular run-mode, given an input character string `atype` – one of the keywords `raster`, `cross-validation`, or `continuity` as described in B - 1. There is also actually another run-mode set by the keyword `detrend`, however its functionality is not used in this thesis and has yet to be conclusively tested. We do not describe it here. Subroutine `switch` searches the character string `atype` for the relevant keywords using the intrinsic function `index`. If the string is not found, `index` returns 0, thus this becomes the trigger for the subsequent execution sequences.

For raster interpolation, the keyword `rast` is searched for. If found, the public variables in module `datmisc`; `kruntyp`, `kuesst` and `intrmod`, are set to 1, 2 and 0 respectively. These variables control various aspects of the estimation. The integer `kruntyp` specifies what sort of estimation problem is being undertaken; 1 for basic estimation, and 2 for cross-validation. The integer `kuesst` specifies the quantity that is being estimated (the three options in IV - 6); it is 1, 2 or 3 for unfiltered, noise-filtered or drift estimation respectively. Finally the integer `intrmod` specifies the number of interpolated drift functions that are to be used. The reason why `intrmod` and `kuesst` are initially set to 0 and 2 respectively, is so that when in subsequent calls to `strucmode` and `mirror`, these functions are interpolated in the first place, the interpolations are standard, noise-filtered estimations. A call to `spec` is made to set `NKI` which consists of the `kkr` variables that form the basis of the cokriging, and also to set `MIR` which consists of the `imo` variables that are to form the basis of the interpolated modes. The estimation type is also retrieved and stored

temporarily in `kuess`. The array `MIR` (and `imo` for its length) are passed to `strucmode`, which performs structure identification for the interpolated drifts, as described in Section E - 2. This allows the subroutine `mirror` to construct the `*.km` file for final estimation using these interpolated modes. The final estimation step follows; the values `kuesst` and `intrmod` in module `datmisc` are finally set at their true values, and `taste` is called on the `kkR` datasets listed in `NKI` to perform the final structure identification. After the call to `taste`, the (possibly multivariate) covariance model is stored in the data-structures of module `krige`, and is ready to use. The next call to `graphout` produces the estimates and the final output file `krigout.txt`.

The second block of code handles the occurrence of the keyword `cross`, indicating that a cross-validation is to be performed. This follows almost exactly the same execution pattern as for raster interpolation, except that instead of a final call to `graphout` to organise the interpolation of a raster, the variable `kruntyp` (in `datmisc`) is set to 2 and the routine in `dataset`, `datmisc` is called instead. Setting `kruntyp=2` here controls the call to `xvalid` in the `antekrige` routine `enqueue`, later. The final block of code recognises the keyword `continuity`, which simply calls `spec` to generate a list of kriging variables `NKI`, and then performs a basic univariate structure identification by calling subroutine `paper`. The array `MIR` is not used in this case. Subroutine `paper` performs structure identification for a single dataset, so it is called `kkR` times in loop 11 – once for each dataset in `NKI(:)`.

In the above routine `switch`, subroutine `spec` performs the task of obtaining the basic specification from the input file `krigin.txt`; the `nkin` datasets to use as the kriging variables in `NKIND(:)`, the `nint` datasets from which drift functions are to be interpolated, and the estimation type `kuesst` (1, 2 or 3). For each, `rewind` is called on the file `krigin.txt`, which is then read line-by-line (in 57, 37 and 13). Each line is read into the temporary character string `hitch`, and searched for the particular keyword. If it is found, the loop (in 57, 37 and 13) breaks to the next line which is scanned for the relevant data (a list of integers), and if the end of the file is reached instead, the code skips to a control label (40, 60 or 46) beyond this `read` statement. At this point, should the relevant keyword not have been found or spurious specifications been read in, default values `nint=0` and `kuesst=1` will be set.

However, as the cokriging variables are absolutely necessary – if these are not found, the user is instead prompted interactively for them.

Subroutine `paper` accepts an integer argument `nkind` specifying a particular dataset in the `*.kr1/*.kr2` files and performs a univariate structure identification on it. This step comprises only the preliminary identification of spatial statistics and variogram models, described in Sections IV - 4 and IV - 5.1. Firstly, subroutine `paper` calls `dimple` – after setting the input array `NK(:)` appropriately, and generates a list of close-packed spatial statistics `VA(ndimd+1,:)`. This list is written to `krigdr.txt`, along with the total variance and average of the nodal data, `ave` and `var` – determined in a prior call to `qvar` in module `dataset`. Outliers in the list `VA` are then removed with a call to `inlier` (in module `sample`), and the subroutine `trane` is called to identify basic models, returned in `VUM`, and their corresponding weights in `STR(:)`. Note that the first element in `STR` relates effectively to the nugget effect, which is not set explicitly – hence the address of the second `real*8 STR(2)` is passed into `trane`. The array `MODO` specifies the models to use – one of spherical, exponential and Gaussian. Every second subscript is zero, as `VA` is always an auto-covariant structure and negative base models are not required. The models are written out to `krigdr.txt` along with their multiplicative constants in `STR`. These models are also written to `krigdia.txt` and `krigrasta.txt` by calls to `graphcorg` and `saveme` (in module `krige`), for which purpose they are loaded into `krige` by calls to `vrload` and `siload`. Note that the nugget effect `STR(1)`, is set to zero because as the models are fitted to statistics normalised between zero and one in `VA` – the absolute magnitude of them is trivial. Furthermore, unlike the list of statistics in `krigdr.txt` (written earlier) the list of statistics in `krigrasta.txt` has the outliers removed.

Another routine that takes care of a specific run-time behaviour is subroutine `graphout`. For reasons largely of convenience, the `nkin` variables in the cokriging `NKIND` are passed in along with the `modes` datasets listed in `MIR` from which interpolated drift functions are to be constructed. Firstly, the `krigin.txt` file is scanned for the keyword `raster`, after which is defined a two-dimensional window in the spatial dimensionality of the interpolated space (as described in Section B -

1.1). The three vectors defining this window are loaded into `v1`, `v2` and `v3` (`v1(:)` and `v2(:)` are the axes defining the extent of the spatial window, whereas `v3(:)` is the origin of these axes). The subroutine also reads in `nres` – the desired resolution of the raster, defined in the line following `v1`, `v2` and `v3` in `krigin.txt`. Subroutine `graphout` attempts to make the resolution of the raster of interpolations the same in each direction by increasing the resolution in the longer direction of `v1` or `v2`. Thus a resolution in the `v1` direction `nb` and a resolution in the `v2` direction `na` is calculated. A close-packed list `GR(ndimd, :)` of `k` estimation points is then calculated in loops 13, 15 and 17, and passed into `preintr` to interpolate values of the drift functions, stored in the matrix `PRE`. This matrix is then passed with `GR`, `NKIND`, `k` and `nkin` into `master`, which performs the interpolations and leaves the results in the close-packed array `RE(3, :)`. The values in this array are then written to the `krigout.txt` file in loops 19, 21 and 23 as previously detailed in Section B - 1.1.

## E - 5. Module `krige`

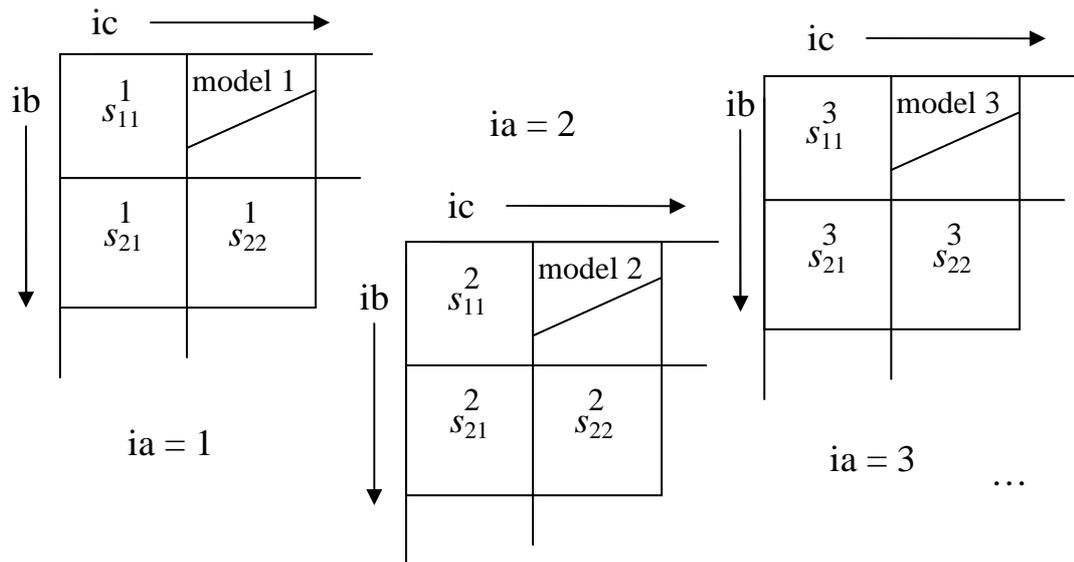
Module `krige` takes care of the actual estimation, and the fitting of the covariance models to a given set of statistics. These two activities are lumped together because they necessitate the existence of similar functions – and furthermore, changes to how these functions operate will tend to affect all of the routines therein. Also kept in module `krige`, are a number of data structures that store the covariance model to be used in estimation and which are updated as the covariance model is fitted.

In the same manner as for the other significant modules, `krige` has an initialisation routine, subroutine `krinit`. This routine sets the internal data dimensionality `krdim` for `krige`, after first checking that it is legitimate, and then parses the file `krigin.txt` for the desired support of the estimation, and also the order of the polynomial drift functions. We assume that these will not affect the estimation of the interpolated drift functions to a great degree although in future, these values could be set elsewhere to circumvent this conflict. Subroutine `krinit` also sets a random number seed `iniran` (this should be initialised from the clock, but

a static seed is useful for the purposes of debugging), and calls `rffinit()` to initialise all of the covariance model data-structures in module `krige`.

There are a significant number of private data-structures in module `krige`. There is also one public integer parameter `ksolmax`, which defines the maximum number of points per dataset in the estimation. The entire module uses the parameters defined in `universal`, `komax` and `kdimax`. The private parameter `maxmod` controls the maximum number of models in the covariance model (the maximum value of  $i$  in Equation (3.69)), the parameter `localsizer` provides a commonly used local sizing parameter, and a `real*8` parameter `eps` defines a “small” `real*8`, considered acceptably large to avoid potential underflow problems when running diminishing numerical algorithms. Also defined are private integers `krdim`, `nvmod`, `kokri` and `iniran`; respectively the spatial dimension, the number of variogram models in the covariance model (3.69), the number of variables in the cokriging, and as previously mentioned, a random number seed. The private array `BLOCK(:)` contains the sidelength of the multidimensional cube which is considered to be the support of the estimate – if the sidelength is zero, point support is assumed. This is to allow block kriging which calculates the average grade of a region in space; a cube in this case. It should be noted that block kriging is not yet fully tested (although semi-implemented). Furthermore, a common block called `KRIBIG` is declared which nominally contains the array `SPACE`. This memory partition is used in `krige` as scratch space for the basic model identification and subsequent coregionalisation fit. It is partitioned rather differently in each case, but as it is a common block, this does not matter.

The other private static arrays in `krige` specify the covariance model. The array `VARM(:, :, :)` contains a list (not close-packed) of variogram models. There are `nvmod` of these, and each of them resides in the first two subscripts of `VARM`, in the subscript ranges `VARM(1:krdim, 1:krdim, :)`. In fact, only half of these addresses are used, as the positive definite transform matrices  $\mathbf{T}^i$  in Equations (3.28) and (3.69) are symmetric. The half that is used is detailed in Figure E—4; for the  $i^{\text{th}}$  model the transform matrix in `VARM(ib, ia, i)` is accessed by `ia=1,krdim` and `ib=1,ia`. This always leaves an entry `VARM(2, 1, i)` that can be used to store the type of model. This `real*8` address is equivalenced to two concatenated `integer*4` addresses, the first of



**Figure E—4: Arrangement of the half-populated array VARM(ic,ib,ia) in memory.**

which indicates the model type; 1 indicates a spherical model, 2 indicates an exponential model and 3 indicates a Gaussian model (as detailed in Section III - 3, Equation (3.23)). In addition to this, the first model  $\text{VARM}(:, :, 1)$  is *always* reserved for the nugget effect, regardless of the entry in  $\text{VARM}(2, 1, 1)$ . **It is very important to understand this construction!**

The array  $\text{SILM}(:, :, :)$  contains the coefficients  $s^i$  introduced in Equation (3.69), in the addresses  $\text{SILM}(i, \beta)$ . Although these coefficients also form symmetric matrices, the extra space is not required and the elements  $\text{SILM}(i, \beta) = \text{SILM}(i, \beta)$ . The private array  $\text{SILT}(:, :)$  is used to store the total variance of the elements in  $\text{SILM}$ , and it is set as a consequence of  $\text{SILM}$ , by subroutine `setsilt`. The final element of the covariance model describes the possible offset in the cross-covariance models in the array  $\text{OFFSET}(:, :, :)$ . The vector containing the  $\text{OFFSET}(:, \beta)$  is stored in the first subscript, and the remaining subscripts index the variable pair to which it belongs. Note that the symmetry of the offset vectors is again not taken advantage of, although whenever these entries are set it is forced that the diagonal offset vectors  $\text{OFFSET}(:, \beta)$  are zero and the off-diagonal entries are related by  $-\text{OFFSET}(i, \beta) = \text{OFFSET}(i, \beta)$ . This preserves the relationship in Equation (3.63). Finally, there is a private array  $\text{GAMINT}(:)$  which is used in calls to the subroutine `trane` to communicate values through several nested calls without passing them through the argument lists.

Module `krige` produces estimates on the basis of the nodal data stored in module `datmisc`. This data is to be set beforehand, and is stored in the array `DAT(:, :)` which is a list of shape `DAT(krdim+intrmod+1, :)`, dynamically allocated in `antinit` of module `antekrige`. The list forms contiguous blocks of data from each or some of the datasets, which are now indexed in the order in which they appear on this list. Hence, the list `NTYDAT(:)` is `kokri` elements long, and in the list `DAT`, the first `NTYDAT(1)` elements in the slow subscript correspond to the first dataset, the next `NTYDAT(2)` elements correspond to the second dataset, and so on, up to the number of variables in the cokriging, `kokri`. Also stored in `datmisc` is `KORDR(:)` which contains the order of the polynomial drift for each of the `kokri` variables, in the same order in which they appear in `DAT`. Note that this order may be totally different from the order in which these variables appear in the original binary files – variable ordering in `krige` is an inner layer of organisation. Note that the `KORDR(:)` stores `kokri+1` integers as the last integer flags whether independent or non-independent drifts are to be used (Section III - 6). If this entry is less than 0, non-independent drifts will be used and only the first integer in `KORDR(:)` indicates the polynomial order for all drifts. Otherwise, a list say of [0, 1, 2, 0] wherein `kokri=3`, will indicate that zero-th order (ordinary kriging), linear and quadratic families of monomial functions are to be used as independent drifts for each variable. As explained before, the integers `kuesst` and `kruntyp` communicate the type of estimation and the type of estimation problem respectively, and the variable `intrmod` signifies the number of interpolated drift functions, whose values at each node are also stored in `DAT`.

There exist mechanisms to set and retrieve the covariance model in `krige`. The subroutine `rfininit` initialises the model transform matrices and types – the arrays `OFFSET`, `SILM` and `SILT` are set entirely to zero, and all of the variogram models are set to be spherical (type 1) with a range of 0.5. Subroutine `vrload` loads `num` models from the close packed list `VARI(krdim, krdim, *)` of models – which are arranged in the format listed previously, to the static arrays `VARM(:, :, :)`. In loop 11 `ia=1, num` it checks the legitimacy of the model types (if the type does not exist it sets it to spherical type 1) and loads it into `VARM(:, :, ia+1)` in nested loop 13. In this way the model `VARM(:, :, 1)` is effectively the nugget effect. Although it is never

used, it makes sense to start the indexing from two to make it compatible with loops that index the elements in `SILM`. The number of variogram models `nvmod` is therefore set to `nvmod=num+1`. The coefficients in `SILM` can be loaded by the subroutine `siload`. This routine sets `kokri=kkri` and makes a straight copy of the elements in `CORR` to the corresponding elements in `SILM` – it does not check for symmetry. Note that the ordering of the subscripts is different in these arrays. The fastest index in the static array `SILM(:,kdimax,kdimax)` indexes the models, and corresponds to the slowest index in the close-packed list `CORR(ndimd,ndimd,*)` that also indexes the models. It was mentioned that the array of total variances `SILT` is calculated from `SILM`; this is usually accomplished using subroutine `setsilt`.

Subroutine `cregoffset` sets the array `OFFSET` by reference to the input `HOFF`. This is a close packed list `HOFF(krdim,*)` of vectors for the offsets relating to each the covariance functions in the following order:  $C_{11}, C_{21}, C_{22}, C_{31}, C_{32}, C_{33}$ ... where the subscripts indicate the place order of the parent dataset in `DAT` (loops 11 and 13 correspond to this ordering). Note that only half of the offsets are stored in `HOFF`, the other half are of opposite sign and are set within loop 15, corresponding to  $C_{12}, C_{13}, C_{23}$ ... As the diagonal offsets must be zero, these elements are ignored in `HOFF`, and they are set directly to zero in `cregoffset`. Finally, subroutine `support` sets the array `BLOCK(:)`, which informs the support of estimation for each of the variables in the cokriging. Arguably all of the elements in `BLOCK` ought to be the equal anyway, but the extra flexibility has been retained because as yet, block kriging has not been fully tested and developed.

The means of retrieving the covariance model from `krige` is either by output parameters in subroutines that perform a particular modelling task; `koreg` and `trane`, or by a call to subroutine `getmod1`. This routine simply returns the variogram models `VARM(:, :, :)` in the close-packed list `VARI(krdim, krdim, *)`, the coefficients in `SILM(:, :, :)` in the close-packed list `CORR(kokri, kokri, *)` (note the subscript reordering here) and the offsets `OFFSET(:, :, :)` in the list `OFFS(krdim, *)` (storing only half of them, as elaborated previously). Note that in `VARI`, it does not return the nugget effect but starts at the second model – by default the first model. It also returns the number of models `nmod=nvmod-1`, the number of kriging variables in `kkri=kokri` and the dimensionality of the domain `ndim=krdim`.

The base subroutines for calculating variogram models for a given lag vector are `vario` and `crelo`. Subroutine `vario` takes an argument `H(*)` containing a lag vector and calculates the normalised variogram value corresponding to the model and the anisotropy transform in the  $n^{\text{th}}$  model. This is effectively the function;

$$I^n \left( \sqrt{\mathbf{h}^T \mathbf{T}^n \mathbf{h}} \right)$$

and it ranges between zero and one. The resulting value is returned via the argument `val`. Subroutine `vario` first checks the value of `n`, and if it equals 1, it evaluates the nugget effect model and returns via `99 continue`: if the euclidean length of `H` is greater than the parameter `eps`, it is deemed close to zero and the nugget effect is one; it is zero otherwise. If `n` is not equal to 1, the the following sequence is evaluated. Firstly, subroutine `modnorm` is called to evaluate the transformed range `ans` corresponding to the transform matrix in `VARM(:, :, n)`, effectively;

$$\sqrt{\mathbf{h}^T \mathbf{T}^n \mathbf{h}}$$

The value `ans` is stored in `GAMINT(1)` – for the purposes of subroutine `trane`. The variogram model type is then retrieved by checking the first four bytes (`integer*4`) of `VARM(2, 1, n)`, and taking the absolute value of it, `nt`. This variable should be either 1, 2 or 3 for spherical, exponential or Gaussian variograms – the value is checked and the code diverts to `10 continue`, `20 continue` or `30 continue` to call the appropriate function, `gammas`, `gammae` and `gammag`, respectively. Control subsequently passes to `98 continue` where `GAMINT(2)` is set as the return value, `val`.

The calculation of `vario` in this manner is somewhat historical, as the code actually deals more usually with covariance functions and not variograms. The normalised covariance function or correlogram is calculated in `crelo`. The normalised covariance function is related to the normalised variogram via the relation (4.17). Subroutine `crelo` therefore calls `vario` and passes into it the inputs `H` and `n`. Subroutine `vario` passes out the value `var`, which is renormalised depending on whether the `integer*4` variogram identifier `NT(1)` in `VARM(2, 1, n)` is positive or negative. This indicates a positive or negative covariance model, the use of which is

explained in Section IV - 5.1 and exemplified in Equation (E.1). This functionality is needed for basic structure identification, performed by subroutine `trane`. If `NT(1)` is a positive integer, the renormalisation merely flips the variogram output `var` from `vario`, and returns `val`;

$$\text{val} = 1 - \text{var}$$

If `NT(1)` is negative, the same flip is performed but the result is multiplied by `-1` so that

$$\text{val} = \text{var} - 1$$

An obvious future improvement would be to use normalised covariance functions for `gammas`, `gammag` and `gammae` instead. Currently these functions merely calculate the normalised variograms for some input `h`, as presented in Equation (3.23), with sill and range of unity.

Subroutine `modnorm` calculates the norm of the vector `x(:)` as expressed by Equation (3.27), and returns the normalised vector in `ans`. It uses the  $n^{\text{th}}$  transformation matrix, stored in `VARM(:, :, n)`. Firstly it calculates the Euclidean norm on `x`, storing the squared components of the vector in `HOP(:)` for later – if this norm less than `eps`, then it sets `ans` to zero and returns. The norm consists of a sum of diagonal and off-diagonal components of the outer product of `x`. The diagonal components of this are already stored in `HOP` and is summed in the total `di`, and the other half is evaluated and summed in loops 19 and 21 in the total `bt`. The total `ans` is checked for positivity – if it is not it is set to zero and an error is flagged, as well as a call to `saveme` to record the covariance model at fault. When (normally) it is positive the square root of it is returned.

What is used more extensively in estimation is the total covariance model – that is a weighted sum of models, as expressed by Equation (3.69). To calculate these, subroutines `crelog` and `variog` are used. Subroutine `crelog` refers to the modelled covariance function (which is used in estimation), and `variog` refers to the modelled variogram function (which is included mostly for completeness). Except for a call to `vario` instead of `crelo`, subroutine `variog` works analogously to `crelog` which is described here. Subroutine `crelog` takes the following inputs; a lag vector `H(:)`, the covariance function `Ciijj` is specified by `ii` and `jj`, and the

integer\*4 flag *iflag*. If *iflag* is 2 then the calculation of the model covariance function does not include the nugget effect model, and if it is 1 then it does: this is useful for producing noise filtered estimates. Firstly, *crelog* subtracts the offset corresponding to  $C_{iijj}$  in *OFFSET(:,ii,jj)* from the lag vector, and stores the resulting vector *HOFF(:)*. It then calls *crelo* for all the models in loop 11 (including the nugget effect as required by *iflag*) returning the temporary normalised value *sub*, which is between 0 and +1. This value is then multiplied by the coefficient corresponding to  $C_{iijj}$  and the model *ia*, *SILM(ia,ii,jj)*, and added to the total *tot*. This total is finally returned by *va*.

The principal purpose of module *krige* is to finally construct the covariance matrices in Equation (4.55) (general form of (3.37), (3.55) or (3.58)) and solve them for the weights, from which an estimate and an associated kriging variance may be calculated by Equations (3.46) and (4.58). The principal routine for estimation is subroutine *krigery* – its function is extended by subroutine *xvalid* to perform cross-validation exercises.

Subroutine *krigery* takes a close-packed list of *npts* points in *PTS(krdim+intrmod,\*)*, and produces a list of estimates at these points in *POUT(3,\*)*. The coordinates of each point in the list *PTS* are stored in the first *krdim* *real\*8*, and the remaining *intrmod* *real\*8* are used to store the values of the interpolated drift functions at these points. For a point *i* in *PTS(:,i)* the three *real\*8* in *POUT(:,i)* will contain respectively the required estimate (of the primary variable's value, with or without noise, or of its drift component, as required by *kuesst* in *datmisc*), followed by the kriging variance of this estimate and a 'soft' value, which is used for special purposes. The integer\*4 parameter *ierrf* is used to store an error flag, and returns 0 in error-free operation. Subroutine *krigery* first checks that the estimation flag *kuesst* in *datmisc* is legal – if it is not, it is reset to 1 (unfiltered estimation of the primary variable). It then initialises some useful strides, *npstr*, *ndstr* and *nbit*, and initialises the error return *ierrf*=0. The initialisation routine *krigini* is called to initialise the operational array *LIMS(:)*, and calculate the total number of data-points *krpts* and equations *neq* that will be required in the kriging. The strides *ncov* and *neqq* can then be set, and space for the

system of equations can be allocated in the dynamic array  $\text{COV}(:, :)$ , which is subsequently entirely set to zero with a call to the speed routine `pzero`.

The array `COV` is used to store the left and right-hand sides of the equations in (4.55), and later their solution vectors. The left-hand side is constructed from the data resident in module `datmisc`, and it is the same for all estimation points in `PTS`. To avoid reducing the same matrix multiple times, the reduction is performed on multiple right-hand side vectors at a time, as the right-hand side is by contrast, different for each point in `PTS`. The structure of the array  $\text{COV}(:, :)$  is described in Figure , which shows how blocks of it are reserved for particular variables and how the equations for the Lagrange multipliers are arranged, under the two cokriging drift paradigms (independent and non-independent) described in Sections III - 6.1 and III - 6.2. The coefficients in these will consist of either point values of polynomial drift functions, or the interpolated drift point values originally in the `*.km` file. The column index in the covariance matrices corresponds to the fastest varying subscript

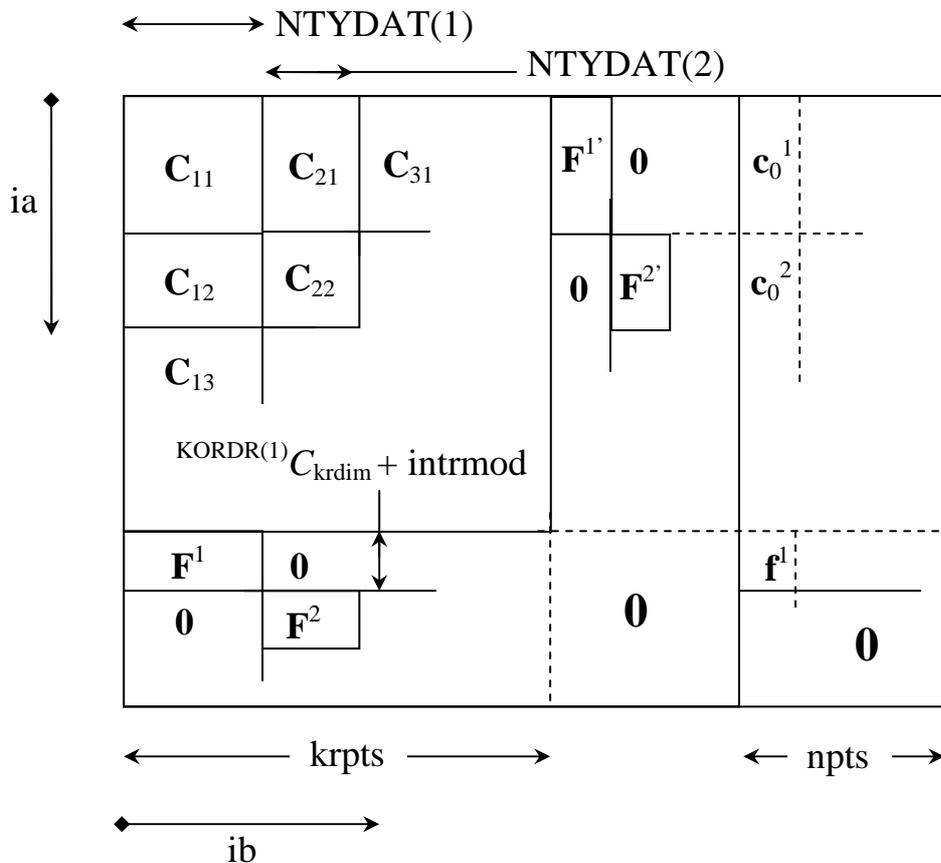


Figure E—5(a) Arrangement of  $\text{COV}(\text{ib}, \text{ia})$  in krigery for independent drifts.

in COV.

The subroutine `krigini` uses the data in `datmisc` to produce the values `LIMS(*)`, `krpts` and `neq`. These are essentially operational values in `krigery` – but it makes sense to initialise them together to compact the code. The array `LIMS(*)` stores pointer limits for the covariance matrices, which when exceeded signify that the next cross-covariance function must be used to inter-relate data from different datasets. The first limit is therefore `NTYDAT(1)`, the second is `NTYDAT(1)+NTYDAT(2)`, the third is `NTYDAT(1)+NTYDAT(2)+NTYDAT(3)`, and so on, as calculated in loop 9. The number of data points in the kriging `krpts` is therefore the last element, `LIMS(kokri)`. The total number of equations is also returned in `neq`, and this will encompass equations to remove the drift in the system of equations. The number of monomial drift functions is calculated by `ntot` in loop 11 – there are  ${}^{krdim+KORDR(ia)}C_{KORDR(ia)}$  of them for the  $ia^{th}$  variable. To the total of monomial drift functions `ntot`, there are also `krpts` equations from each nodal data point in the estimation, and there are `kokri*intrmod` interpolated drift functions. Note that should non-independent drifts be specified by the last integer of `KORDR(kokri+1)`

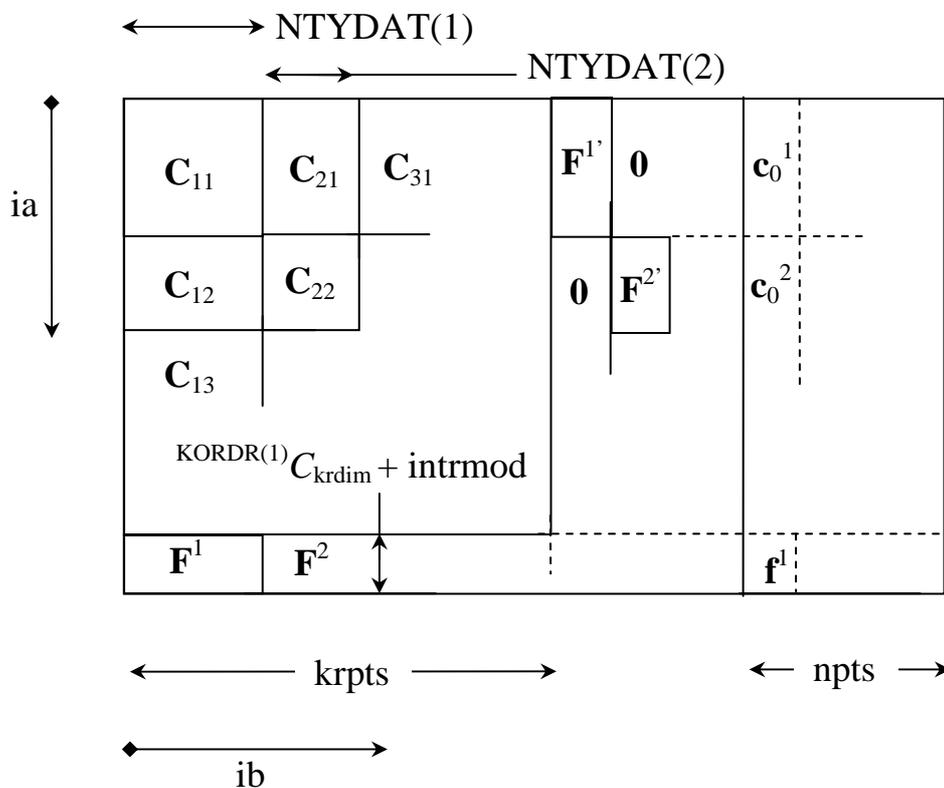


Figure E—5(b) Arrangement of COV(ib,ia) in krigery for common drift.

(which is negative in this case), there are only `intrmod` interpolated drift functions and loop 11 terminates after one pass only.

After values have been initialised by `krigini`, subroutine `krigery` constructs the covariance matrix `COV` in loops 11, 13 and 17, where loop 11 indexes the rows and loops 13 and 17 index the columns. Loop 13 fills in all of the left-hand side covariances in Figure E—5; it works through one half of the symmetric matrix, and fills in the symmetric entries as they are calculated. The indexes `kb` and `ka` indicate which model covariance function to use in the call to `crelog`, and they are incremented in each of the loops if the loop counters `ib` or `ia` exceed the limits `LIMS(kb)` or `LIMS(ka)` respectively. Note that the last argument in the first call to `crelog` is 1, thus the nugget effect is always included in the covariance model on the left-hand side. The remainder of loop 11 sets the right-hand sides of the covariance matrices. If `kuesst=3` then drift estimation is required and these entries are set entirely to zero. Otherwise, the row of `npts` estimation point covariances are set relative to the node `ia` of dataset `ka`. If a block estimate is required i.e. `BLOCK(1).ne.0.0`, a call is made instead to `bloksup` which performs the appropriate spatial averaging of the covariance over a regular cube centred at the lag `H(:)`.

After calculating the covariances in the solution matrices, the next step is to calculate the drift model coefficients. The entries here will depend on whether or not the cokriging drifts are independent or non-independent, as indicated by `kxx`, the trailing entry in `KORDR`. For independent drifts a call is made to subroutine `trends`, which fills in the drift coefficients in `COV` as appropriate. Otherwise, for non-independent drifts subroutine `ukmode` is called to fill in the entire block of drift model coefficients located underneath the covariance entries – as indicated in Figure E—5(b). This routine operates relative to the address `COV(1,krpts+1)` and calculates the monomials appropriate to the order `KORDR(1)` and the nodal data locations in `DAT`, and also adds under these the `intrmod` interpolated drift function values, also in `DAT`. In loops 91 and 89, these values are copied to their respective symmetric counterparts in the covariance matrix. Note that because `COV` was entirely initialised to zero, any entries that are not set here are zero, by default. This is important in the next call to `ukmode`, which fills in the block of drift coefficients on the right-hand side relative to the entry `COV(neqq,krpts+1)`, many of which will be

zero for independent drifts. The final step before solution is to copy the right-hand side of the set of equations in `COV` to `RHS`. This is to allow the kriging variance to be easily calculated in the manner outlined in Section IV - 6, as the entries in `COV` are all used in matrix reduction and the results are output to the former right-hand side.

Subroutine `pcsol` is then called by `krigery` to perform a forward reduction on the left-hand side and back-substitute each of the right-hand side vectors. This routine also requires some temporary space which is passed in as the extra row in `COV` starting at `COV(1,neqq)`. Arrays containing space to index row swaps `JOR` and `KOR` are also passed in – these are allocated previously. Subroutine `pcsol` accepts an `neq×neq` matrix and solves it for `npts` right-hand side vectors, in the format used by `COV`. The results are output to the right-hand side vector locations, replacing them. If a singular matrix is encountered the flag `ierrf` is set to 1 and control is passed to 98 `continue` where the allocatable matrices in subroutine `krigery` are deallocated.

In loop 31 of `krigery` the estimates and kriging variances are finally produced for each of the estimation points `ia=1,npts`. A running weighted sum is calculated in loop 33 for the estimate `est` at point `ia`, and the variance `var` at point `ia` – note that the weight for the nodal data at `DAT(nbit,ib)` for the estimate at `PTS(:,ia)` is `COV(ia+neq,ib)`, assigned to `tmp`. By Equation (4.58), the calculation of kriging variance is also affected by the drift model, which serves to reduce kriging variance. This extra component is summed in a further loop 35. The estimate `POUT(1,ia)`, its kriging variance `POUT(2,ia)` and the flexible value, `POUT(3,ia)` are then assigned. At the moment the flexible variable is used to indicate the percentage of the structured variance that is effectively removed by the consideration of secondary data – this is of course, meaningless for univariate kriging problems.

The subroutine `ukmode` calculates blocks of non-zero drift coefficients relating to both the monomial drift functions and the interpolated drift functions – as indicated in Figure E—5. It fills in the  $({}^n C_{r+intrmod}) \times npts$  sub-matrix in `SUBM(nstr,*)` with stride `nstr` for which the columns are indexed by the fastest subscript. Full advantage is made of the flexibility/ambiguity of Fortran90 addressing – the argument `SUBM` is passed in as a first byte address in the array `COV` in `krigery` and local `ukmode` addressing starts from this address – thus a stride of `ncov` passed in from `krigery` will address the next rows in `COV(ncov,:)`. The monomial functions

are calculated at the coordinates (the first `krdim real*8`) in `PTS(nbit,*)` of stride `nbit`. These basis functions occupy the first row in `SUBM(1:npts,1)`, up to the  $nCr^{\text{th}}$  row where `SUBM(1:npts,nCr)` where  $nCr$  is the number of such basis functions, given `krdim` and the maximum monomial order `kuk`. These coefficients are calculated in the nested loops 11, 13, 15 and 17 – note that this section is *limited to the implementation of up to cubic polynomial drifts*, and simple kriging can be effected by means of a negative polynomial order `kuk`. The pointer `k` addresses the relative row in `SUBM(:,k)` – it indexes the drift functions. After the monomial drifts, the interpolated drifts are copied from their locations (listed after the coordinates) in `PTS`, into the rows in `SUBM` in loop 19 using the `speed` routine `repack`.

The subroutine `trends` uses `ukmode` to fill in a number of blocks of drift coefficients when the whole covariance matrix `COV` is passed in with its stride `nstr` from `krigery` and independent drifts are required, for which the maximum monomial orders are indicated in `KUK(*)`. Subroutine `ukmode` is called for each of the cokriging variables for which a block of coefficients is required, in loop 11. The first byte address of the sub-matrix that is written each time is kept in the pointers `ka` and `kb`, which are incremented to the next start point at the end of the loop. The new pointers are temporarily stored in `ma` and `mb`, as the old first byte addresses are useful in loops 13 and 15 where the corresponding symmetric block of coefficients must also be written.

A basic provision for block kriging is made in the code. This is performed by subroutine `bloksup`. This subroutine modifies the right hand vectors in `krigery` so that they reflect an average value over a regular cube whose characteristic dimension is passed in through `supp`. Because all covariance functions relate data in the first dataset to other datasets on the right-hand side, only the other dataset `j` needs to be nominated, and the covariance function to use is specified by passing `1` and `j` to the third and fourth arguments of `crelog` in loop 13. Firstly the resolution `resp` of the spatial averaging is calculated from the resolution parameter `nres`, and the size of the block required in `supp`. An array of addresses `IP` is then initialised to all equal `-nresp` and a vector `A(:)` corresponding to this array multiplied by the scalar `resp` is constructed. The vector `A` is subtracted from the centre of the block indicated by `H(*)` and a covariance is calculated and added to the running average `ave`, and the number

of averaged elements `num` is incremented. Then `IP(1)` is incremented by 1. The elements in `IP(1:krdim)` are to be integers in the range `[-nres:nres]`. If the first index is exceeded when incremented it is reset to `-nres` and the next index `IP(2)` is incremented, which again if exceeds `+nres`, is set to `-nres` and the next `IP(3)` is incremented, and so on. In this way, all possible combinations of addresses are generated in loop 15, and from these all possible locations `A(:)` can be calculated by multiplication by `resp`. The extra value `IP(krdim+1)` is set to `-1`, and its eventual modification to zero terminates loop 13. The average covariance is returned in `vav`. There is another subroutine `prolog`, which performs double averaging over two different blocks and variable types – but its usefulness for the left-hand side in `COV` is speculative at this stage.

The other principal run-type in program `krige` is that of cross-validation. This is implemented in module `krige` by the subroutine `xvalid`. This subroutine takes a list of `npts` estimation points (and interpolated drift functions) `PTS(krdim+intrmod,*)` and returns a list of cross-validations in `RES(3,*)` by removing the data point closest to the requested estimation point. The new value, its kriging standard deviation, and the interval of cross-validation (described previously) are listed in the first index of `RES`. Where data and estimation points are coincident the closest data point is the node itself – if this is the case the logical flag `lrun` is set to `.true.`. The address `JAT(ia)` is a pointer to the closest data point in `DAT` of `datmisc`, – if `lrun` is `.true.`, the original values at these points are copied directly to the `RES(1,ia)`, otherwise a call is made to `krigery` to estimate them (note that `RES(2,ia)` and `RES(3,ia)` are not actually required). In loop 11 then, for each of the points on `PTS(:,ia)`, the closest data point indicated by `JAT(ia)` is removed from the matrix `DAT` in `datmisc` and temporarily stored in `REM`, and the value of `NTYDAT(1)` is temporarily decremented. A call to `krigery` is now made to estimate the value at `PTS(:,ia)` without its closest primary data-point and the result is returned in `RESU(:)`. This then allows the three elements in `RES(:,ia)` to be calculated. The missing data point is then replaced for the next cross-validation, and finally the original value of `NTYDAT(1)` is restored.

Module `krige` also takes care of variogram modelling and fitting the covariance model to the spatial statistics. The first part of this is basic structure identification, as outlined in Section IV - 5.1 and implemented by subroutine `trane`. This subroutine takes the `neg` spatial statistics listed in `STR(krdim+1,*)` and fits the basic variogram model types specified by the array `MODE` concurrently by adjusting their anisotropy matrices, and ranges in different directions. The covariance structures and functions in `krige` are used for this purpose, and the resulting variogram models in `VARM` are then copied to the close-packed output list `VOUT(krdim,krdim,*)` along with their relative magnitudes in `STR(*)`. Note that there are always `isum(MODE,6)` such models – whilst a model’s corresponding linear coefficient in `STR` can become very small, it never disappears.

Before loop 11 `trane` performs a number of initialisations, including a check that there are not too many points in `EGS`, and a call to `trainit`. Subroutine `trainit` initialises the models specified in `VARM(2,1,:)` by interpreting the array `MODE`. This array of size `MODE(2,3)` specifies how many of each model are required in each of its entries; the last subscript indexes the model type `ia` in loop 9, and the first subscript – `ib` in loop 15, indexes whether the model is taken to be positive or negative (the coefficient multiplying it is always taken to be positive, as previously explained in Section IV - 5.1). The total number of models is set in `nvmod`. Subroutine `trainit` also sets the largest `gra` and the smallest `pet` extents of the vectors in `EGS`, for use in initialising the anisotropy matrices in `VARM` for the optimisation algorithm.

The unconstrained optimisation described in IV - 5.1 is performed in loop 11 of `trane`. This loop begins with an `if endif` block, wherein the convergence criteria `siz` is checked (and also an error flag `ierr`). The first time around this loop, this is tripped artificially by the precedent initialisation of `siz` to zero. In this first loop or when convergence is reached, the counter `nconv` is incremented and if the current sum-of-squares in `sos` is better than the best solution so far, stored in `bst`, a call to `nubst` is made to store the current solution in `VOUT` and `STR`, and `bst=sos`. Note that `sos` and `bst` are both initialised to an equally large number so this is not tripped in the first loop. Subroutine `eiginit` is then called on the array `EVEC` to generate a random initialisation that captures the points in `EGS` in the anisotropy model, and the array `CONST` is initialised to zero. These arrays are the re-parameterised variables

forming the optimisation problem, and they define the actual coefficients in `VARM` and their multiplying coefficients via a call to `calvarm`. Once the basic initial model has been set up, the Jacobian matrix `COB` and the residual vector `RES` can be calculated by a call to `jacres`, which are used to initialise the optimisation routines by calling `lvinit` (note that `COB` is destroyed in this process). The algorithm may then (re-)commence iterations towards a solution.

In Section IV - 5.1, it was explained how a direct constrained optimisation over the matrices  $\mathbf{T}^i$  and their coefficients  $t_i$  could be reparameterised as an unconstrained optimisation over  $v_{jk}^i$  and  $\theta_i$ . In subroutine `trane` (and related) these new variables are kept in a common block, `KRIBIG` in the arrays `CONST` and `EVEC`. The actual values  $\mathbf{T}^i$  in `VARM` are calculated from `EVEC` by `calvarm`, and the linear coefficients  $t_i$  are kept in `SILM(:,1,1)` which again must be calculated by `calvarm`. Note that for these routines only, the element `SILM(1,1,1)` does not signify the nugget effect, but corresponds to `VARM(:, :, 2)`; the coefficient `SILM(1,1,ia)` multiplies `VARM(1,1,ia+1)`. As the `crelog` routines are never used here, this does not present a conflict.

Subroutine `calvarm` uses Equation (4.32) to calculate the coefficients in `VARM` from the vectors `VECT` in loop 11. Note that `VECT(krdim,krdim,*)` (effectively, `EVEC` in `trane` is of this shape too) contains the elements  $v_{jk}^i$  in `VECT(j,k,i)` – the spatial dimension is the fastest subscript and the models are the slowest subscript. Noting that some terms appear a lot in the partial derivative calculations in Equations (4.38) and (4.39), in loop 25 subroutine `calvarm` also calculates and stores the exponential of each element in `CONS(ia)` in `SILM(ia,1,2)`. Then noting whether or not the coefficient multiplies a positive or negative model it keeps a running total of the ‘positive’ denominator in Equation (4.39) in `SILM(1,2,1)` and the negative denominator in `SILM(1,2,2)`. Their reciprocals are kept in `SILM(2,2,1)` and `SILM(2,2,2)` respectively for future reference. Note that `SILM` is merely used in these routines as a convenient publicly accessible storage space.

Subroutine `eiginit` takes a close-packed array of `num` transform models in `VECT(ndim,ndim,*)` containing the vectors  $v_{jk}^i$ , and initialises them randomly so that their eventual ranges should still lie within the ranges set by `big` and `tiny`.

These random initialisations are performed in loop 17 until an acceptable one is found – note that it is the  $\mathbf{v}_k^i$  vectors are thus scaled so that the reciprocal of their Euclidean length is between `big` and `tiny`, thus transform matrices should have a corresponding range between `big` and `tiny`. This procedure does not guarantee the required scaling but even an approximate scaling between these values is acceptable. This is performed for all models in loop 11.

The other initialisation subroutine `lvinit` specifically initialises the optimisation routines in module `optimisation`. For this, the number of residuals `nex` and the number of parameters in the optimisation `nro` must be passed in, with the initial Jacobian matrix `COB`. The magnitude of the columns of `COB` is then calculated (after reshaping by `ptrans`, whereupon it is properly of shape `COB(nex,*)`), and the average value is used to initialise the damping factor `damp` for the optimisation routines. These are initialised by a call to subroutine `linit` in `optimisation`.

We shall now continue with the description of subroutine `trane`, starting where we left off in loop 11. After the initialisations in the leading `if endif` block, the Jacobian matrix `COB` and residual vector `RES` are calculated by a call to `jacres`. The ordering of the entries in these arrays corresponds to the parameter ordering, and the ordering of the spatial statistics in `EGS` – as shall be discussed vis-à-vis subroutine `jacres`. A call is then made to subroutine `lemarq` in `optimisation` to calculate the change in parameters  $\delta\mathbf{r}$  in Equation (4.23), which is returned in `DIR(:)`. In this parameter vector, the first `mod real*8` correspond to the parameters  $\theta_i$ , in `CONST(2)` through `CONST(nvmod)`, and the next `nv` coefficients correspond to the coefficients  $v_{jk}^i$  in the same order as they appear in the array `EVEC`. Also passed out of `lemarq` is the current sum-of-squares `sos`, used for comparison with the previous ‘best’ solution stored by `nubst`. The convergence criterion `siz` is calculated as a ratio of the step-size  $\|\delta\mathbf{r}\|$  and the actual state vector  $\|\mathbf{r}\|$  – when this becomes small, the solution is assumed not to be moving anywhere, and so the set of iterations are terminated by the leading `if endif` statement. Finally in loop 11, the vector `DIR` is added onto the existing parameters in `CONST` and `EVEC`, from which the updated model is re-calculated by `calvarm`. The new `RES` and `COB` arrays can be calculated in the next loop and the next Levenberg-Marquadt step made.

Subroutine `jacres` takes the coefficients  $v_{jk}^i$  in `EVEC(krdim,kdrim,*)`, and the spatial statistics in `EGS(krdim+1,*)` and where `nro` is the number of parameters in the optimisation, it fills in the coefficients in the Jacobian matrix `YAK(nro,*)` as calculated by Equations (4.38) and (4.39), and the residuals vector `RES(*)` as calculated by (4.37). The first loop `9` calculates the entries in the Jacobian relating to the parameters  $\theta_i$  in Equation (4.39). The array `DCDR(:, :)` is a symmetric array used to store the elements that multiply the terms in the square brackets, i.e. the multipliers;

$$\frac{-\exp(\theta_i)\exp(\theta_d)}{\left(1 + \sum_j \exp(\theta_j)\right)^2} \quad \text{where } i \neq d$$

or

$$\frac{\exp(\theta_d)}{1 + \sum_j \exp(\theta_j)} - \frac{\exp(\theta_i)\exp(\theta_d)}{\left(1 + \sum_j \exp(\theta_j)\right)^2} \quad \text{where } i = d$$

where `DCDR(d, i)` stores the multiplier for the term;

$$\left[1 - I^r \left( \sqrt{\mathbf{h}_e^T \sum_k (\mathbf{v}_k^i \mathbf{v}_k^{iT}) \mathbf{h}_e} \right)\right]$$

required to evaluate the derivative of  $C(\mathbf{h}^e)$  with respect to  $\theta_d$  in Equation (4.39). In calculating `DCDR`, use is made of the  $\exp(\theta_i)$  terms in `SILM(:, 1, 2)` as well as the  $t_i$  terms in `SILM(:, 1, 1)` and the denominator terms in `SILM(2, 2, 1)` and `SILM(2, 2, 2)` to save recalculation on numerically intensive exponential functions. Note also that the parameters  $\theta_i$  are split into two groups -  $\theta_i^+$  and  $\theta_i^-$  in Equation (4.35). Effectively, this presents the equation below.

$$C(\mathbf{h}) = \sum_i \left\{ \begin{array}{l} \frac{\exp(\theta_i^+)}{1 + \sum_j \exp(\theta_j^+)} \left[ 1 - I^r \left( \sqrt{\mathbf{h}^T \sum_k (\mathbf{v}_k^{i+} \mathbf{v}_k^{i+T}) \mathbf{h}} \right) \right] + \dots \\ \dots + \frac{\exp(\theta_i^-)}{1 + \sum_j \exp(\theta_j^-)} \left[ I^r \left( \sqrt{\mathbf{h}^T \sum_k (\mathbf{v}_k^{i-} \mathbf{v}_k^{i-T}) \mathbf{h}} \right) - 1 \right] \end{array} \right\} \quad (\text{E.1})$$

but the derivatives for each of these sets of coefficients are handled *separately* as they do not interact. Thus, when `VARM(2,1,iaa)` is of different sign model to `VARM(2,1,ibb)` then the corresponding entry in `DCDR(ib,ia)` is set to zero. Also, an extra minus sign is factored in if the model in `VARM(2,1,iaa)` is a negative covariance model.

Loop 11 fills in the Jacobian matrix row by row (`YAK(:,ia)`) corresponding to each spatial statistic in `EGS(:,ia)` – its first operation is to zero this row. The factor in (4.39) is calculable by a call to `crelo` for model  $i$  evaluated at  $\mathbf{h}_e$ , at the beginning of loop 13. This is evaluated for all the models `ib=2,nvmod` and the return value `dc` is used to factor the rows of `DCDR` to produce the terms in the derivatives (4.39), which are summed progressively to the `mod` entries from the start of `YAK(1,ia)`. Also in loop 11 is the evaluation of the derivatives in Equation (4.38). These derivatives with respect to  $v_{ab}^c$  are of course stored in the same order as the values  $v_{ab}^c$  appear in `EVEC(:, :, :)`, and for each  $\mathbf{h}_e$  indexed by `ia`, the models  $c$  are indexed by `ib` and the vector-groups  $b$  within a model are indexed by `ic`. In loop 13 `crelo` is called on each of the models, for the vector `EGS(:,ia)` and the model `ib`. The result `dc` is factored by the coefficient  $t_i$  in `SILM(ib-1,1,1)` and a running total `sig` is maintained in place of a call to `crelog`. Note that each time `crelo` is called, the normed  $\mathbf{h}_e$  vector (under the square root sign in Equation (4.38)) is stored in `GAMINT(1)`. It is retrieved in loop 13 to evaluate the denominator in (4.38) and it is also input to the function `dgamma` which calculates  $I^c$  for model `ib`. In this way everything except the final term  $h_a^e \mathbf{v}_b^c \cdot \mathbf{h}^e$  in (4.38) is evaluated and stored in `fa`. Loop 15 then calculates  $h_a^e \mathbf{v}_b^c \cdot \mathbf{h}^e$  using the `speed` dot-product function `pinner` and the vectorial factor function `pfacta`, and fills in the `krdim` values indexed by the subscript  $a$ . Finally as a last step in loop 11, the residual vector is calculated for  $\mathbf{h}_e$ , by its subtraction with the running total `sig`.

The function `dgamma` evaluates the derivative of the normalised variogram model in `VARM(:, :, n)` at  $\mathbf{h}$ . It first checks the type of the model in `VARM(2,1,n)` and redirects via a computed `goto` to `10 continue`, `20 continue` or `30 continue` where functions `dgammaas`, `dgammaae` and `dgammaag` compute the normalised derivative at  $\mathbf{h}$  for the spherical, exponential and Gaussian models respectively – as new models are

added, this routine must also be modified and corresponding derivative functions also added. Finally, a check of whether the variogram model is positive or negative covariance modifies the sign of the output, `var`.

The other part of the covariance model is to fit the linear model of coregionalisation whilst implementing options on the partitioning of variance, described in Sections IV - 5.2 and IV - 5.3. This is implemented in a group of routines headed by subroutine `koreg`. Subroutine `koreg` takes a list of spatial statistics `EGS(krdim+1,*)` and performs a coregionalisation for `kkcr` kriging variables. These statistics are normalised between +1 and -1, so the total variance is supplied in the array `VART(kkcr,*)` for each auto- and cross-covariant structure. A similar array `GOLD(kkcr,*)` is an array of possible limits on the total structured variance – if the entry in `MASK(i,j)` is 1 the total structured variance is set to the limit specified in `GOLD(i,j)`, if it is 0 it is permitted to ‘float’ up to a maximum `VART(i,j)`. Subroutine `koreg` performs the necessary coregionalisation fitting and stores the resulting covariance model coefficients  $s^i$  of Equation (3.69) in the matrix `SILM(:, :, :)`. The list `EGS(krdim+1,*)` comprises a list of auto- and cross-covariant statistics, which are arranged in contiguous blocks in the usual manner; the first such block consists of `NEG(1)` statistics relating to the covariance function  $C_{11}(\mathbf{h})$ , the second such block consists of `NEG(2)` statistics on  $C_{21}(\mathbf{h})$ , the third is `NEG(3)` statistics on  $C_{22}(\mathbf{h})$ , the fourth consists of `NEG(4)` statistics on  $C_{31}(\mathbf{h})$ , and so on. Importantly, the variogram models in `VARM` and their number `nvmod`, need to have been set or loaded before `koreg` can run – these entities are not modified by `koreg`.

Up to the beginning of loop 17, subroutine `koreg` checks for problem capacity and initialises a number of strides, pointers and factors, some of them in a call to subroutine `crinit`. As described in Section IV - 5.2, the coefficients  $s^i$  in `SILM(i, ,  $\beta$ )` are re-parameterised by the coefficients  $u^i_k$  in Equation (4.41), which are stored in the array `EVEC` in common block `KRIBIG`. Subroutine `crinit` initialises the array `EVEC` and the values in `SILM` and `SILT` that result from it, as well as some other aspects of the covariance model. It also re-expresses the structured variance in `STUN(i,i)` (initially copied from `GOLD`) as a fraction of the total variance in

$VART(i,i)$  and checks that the array  $VART$  is symmetric. Furthermore, it sets the values  $ndim$  – the number of parameters in  $EVEC$  and the optimisation,  $nex$  – the total number of spatial statistics,  $ncon$  – the total number of constraints (whether one or two-sided) and  $fa$  – the initial factor  $f_{con}$  in Equations (4.49) through (4.53). The parameter  $iend$  is usually 0 but an error is flagged and action taken should it equal 1 (problem size is exceeded), or 2 (there are insufficient models to perform a coregionalisation).

Subroutine  $koreg$  then calculates the residual vector  $REZ$  arising from the spatial statistics (Equation (4.43)) and the constraint ‘residuals’ (4.51) as factored by  $fa$ . The array  $REZ$  must be calculated before the iterative solution, as must the array  $UNSQ$  which is passed out of  $conres$  and later used in a call to the subroutine  $conjob$ . The part of the Jacobian described by Equation (4.44) is then calculated to estimate a damping factor  $damp$  with which the optimisation routines are initialised by a call to  $linit$ . This last step is very similar to the routine  $lvinit$  used in subroutine  $trane$ . The damping and constraint factors  $dampf$  and  $constrf$  are set according to the problem type and the routine is ready to perform the iterative solution. Note that at this stage, the entries in  $EVEC$  and consequently the coefficients in  $SILM$  are normalised to unit covariance, as are the spatial statistics in  $EGS(krdim+1, :)$ . In  $crinit$ , this is initially achieved by setting all the values in  $EVEC$  to a value  $step$  (variable in  $crinit$ ), which ensures that all coefficients in  $SILM$  will equal  $step*step*kokri$ . To scale the models so that they lie between zero and one,  $step$  is set to  $\sqrt{0.8/(nvmod*kokri)}$ .

As has been described in Sections IV - 5.2 and IV - 5.3, the constraints are implemented as penalty functions, or in effect the penalty residuals offered in Equation (4.51). These residuals are concatenated on the end of the list of residuals produced for each spatial statistic in  $EGS$  – thus they are the  $ncon$  residuals listed from  $REZ(nex+1)$ . The Jacobian matrix also gains a number of rows as a result of the residuals (4.51) – the array  $YAK$  in  $koreg$  is effectively of shape  $YAK(kkr*kkr*(nvmod-1), *)$  wherein we shall nominate the faster subscript as denoting the columns, hence the entries  $YAK(1, :)$  form the first row. The first  $nex$  rows of  $YAK$  consist of the partial derivatives calculated by Equation (4.44), and the  $ncon$  subsequent rows are the constraint derivatives calculated by (4.53). The pointer

`ipj` points to the first byte address in `YAK` of the constraint partial derivatives as addressed by `koreg - YAK(ipj)`.

The iterative solution in `koreg` is performed in loop 17. Firstly, the Jacobian matrix is calculated by calls to `jacob` and `conjjob`. Note that the call to `conjjob` passes in `YAK(ipj)` to calculate the derivatives of the constraint residuals. This is then passed with `REZ` into `lemarq`, which returns the sum-of-squares on `REZ` the gradient `grad` at this point, and the iteration step  $\delta\mathbf{r}$  in `DIR`, which is ordered in the same fashion as the coefficients  $v^i$  in `EVEC`. If there are no errors in `lemarq`, the increment `DIR` is added to `EVEC` in an element-wise fashion by `pplus` and the new `SILM` and `SILT` arrays are calculated by `calsilm` and `setsilt`. A new residual vector is then formed in `REZ` using `rzidul` and `conres` and the algorithm is checked for convergence. As for `trane`, this is indicated when  $\|\delta\mathbf{r}\|$  becomes small with respect to  $\|\mathbf{r}\|$ , but now also if the gradient `grad` becomes too small. Furthermore the counter `iter` must be greater than 7, to allow the iterations to settle before changing the constraint factor `fa`, which occurs whenever iterations converge – which situation is dealt with by the closing `if endif` block.

Should iterations converge in loop 17, the `if endif` block at its end finds the largest violation of the constraint in `soc` – as the covariance models are all still normalised to unit covariance, this is not biased towards a particular model. This is done by using `pninf`, to find the value and then removing the factor `fa` and the square operation on the elements of the subarray from `REZ(nexx)`. The counter `iter` is set to 0 at this point. If the largest constraint violation is less than one part in a hundred or 0.01, the algorithm is considered converged in both the iterations and the constraint factor `fa`, and control breaks to 18 `continue`. If not, `fa` is factored (increased) by `constrf` and the damping coefficient `dry` (returned in each call to `lemarq`) is modified by `dampf`. The optimisation routines are re-initialised with this new damping factor, the constraint residuals are re-calculated for the new factor `fa` by `conres`, and iterations are allowed to proceed again for the modified optimisation problem. When either `koreg` runs out of iterations `nit`, or the result is converged in all respects, a call is made to `setvar` to factor the covariance model by the total variances in `VART` and the subroutine returns.

The subroutine `calsilm` is used throughout `koreg` to calculate the coefficients in `SILM` from a close-packed array of coefficients  $u^i$  in `EVEC(krdim,krdim,*)`. The relevant entries in `SILM` are first set to zero in loops 19 and 21, and then totals are accrued in them in loops 13, 15 and 17. These latter loops implement the summation found in Equation (4.41). Another sundry point in subroutine `koreg` is the storage of the last-known-good solution in the array `EVLNG` whenever the iterations converge. If the call to `lemarq` returns with an error, this solution is loaded back into `EVEC`, hence `SILM` and `SILT` by `calsilm` and `setsilt` – and control terminates with a warning.

Subroutine `setvar` renormalises the covariance model in `SILM`, which by the end of loop 17 in `koreg` is scaled to unity, to a model that is scaled by the variances in `VART`. Furthermore, it also adjusts the matrices `SILM(i, :, :)` such that they meet a basic minimum level of positive definiteness by adding a fraction of the total variance matrix in `SILT`. To each matrix `SILM(ic, :, :)` in nested loops 31, 33 and 35, the component `POS(ic)*SILT(:, :)` is added. The list `POS(:)` is calculated earlier in loop 17. Adding a small fraction of `SILT` to the arrays in `SILM(ic, :, :)` will augment the total variance by the amount `psum(POS(2),nv)` – this is corrected for by factoring all coefficients in `SILM` by `div` in loop 35. The entries in `SILM` are also factored by the total variance `VART(ib,ia)`, so that the covariance model is no longer scaled to unit covariance.

The determination of `POS(:)` in `setvar` is performed in loop 17. This loop cycles through the coefficient matrices  $S^i$  and `SILM(ia, :, :)` and compiles a close-packed matrix `TEST` of the matrix `SILM(ia, :, :)` augmented by `POS(ia)*SILT(:, :)`. The condition number of `TEST` is calculated by `pconda` and if it is less than `crit`, or the entries in `SILM(ia, :, :)` are relatively small, then the corresponding flag in `MASK(ia)` is set to 1. Otherwise `POS(ia)` is augmented by `scat*big` an amount proportional to the ‘size’ of the matrix in `SILM(ia, :, :)`. This continues until all flags in `MASK` are set to 1, which eventually causes a break from loop 17 by its `while` mechanism. Note that before this loop, two values are initialised – the limiting condition number `crit` is set to `50*crit` – relative to the condition number of the total variance in `SILT`, and the increment `scat` is chosen to be the a

small fraction of the total variance which is unity, and proportional to the ‘average’ variance expected of all the models.

The nugget effect is also set by `setvar`. This model in `SILM(1, :, :)` is not explicitly modelled, and must be set at whatever variance remains after the covariance model fitting. This is performed on the unit covariance model in loop 27 of `setvar`: if the entry in `NGST(ia, ia)` is 1 the nugget effect is to the difference between 1.0 and the desired level of structured variance in `STRU(ia, ia)`, if `NGST(ia, ia)` is 0, the nugget effect is simply set to the difference between 1.0 and the value in `SILT(ia, ia)`. The nugget effect is then factored by total variance `VART` in loop 33. Note that the nugget effect is only set for the auto-covariant parts of the covariance model! It would be possible to do away with the cross-covariant parts in `MASK` and `STUN` of subroutine `koreg`. These were only ever included as historically, the code did implement cross-covariant constraints on the nugget effect. Although this functionality has been removed, the framework remains for future developments.

The subroutines `rzidul` and `jacob` calculate the residual (Equation (4.43)) and the Jacobian entries (Equation (4.44)) for subroutine `koreg`. Subroutine `rzidul` takes a list of spatial statistics `EGS(krdim+1, *)`, which index the covariance functions in blocks described by `NEG` (as described for `koreg`), and calls `crelog` for the coordinates starting at `EGS(1, ic)` relating to the covariance function  $C_{ia, ib}$  in loop 15. The residual `RES(ic)` can then be calculated with respect to the spatial statistic in `EGS(krdim+1, ic)`. Subroutine `jacob` is rather more complicated, but returns the Jacobian in `DYDX` through calls to `jarow`. This array, of shape `DYDX(kokri*kokri*(nvmod-1), *)` indexes the subscripts  $a$ ,  $b$  and  $c$  for (4.44) in the columns (the fastest subscript) for the parameter ordering in `EVEC`, and the spatial statistics  $e$  in the rows. It is initialised to zero before values are totalled in it by calls to `jarow`. Subscripts  $\alpha$  and  $\beta$  in (4.44) are cycled by `ia` and `ib`, which are passed into `jarow` along with the lag vector `EGS(1:krdim, ic)` to calculate the row of the Jacobian relating to `ic`. Array `EVEC` is also passed in so that it may be addressed differently. In subroutine `jarow`, Equation (4.44) is evaluated by a call to `crelo` that produces `ttmp`, which is then factored by the appropriate entry in `EVEC`. Note that this routine only accumulates values – this means that when  $j$  is equal to  $k$  a double summation is performed which corresponds to the first case in Equation (4.44).

Performing an analogous function for the constraint residuals listed in Equation (4.51) are `conres` and `conjob`. The constraint residuals are listed after the residuals from (4.43) and in the usual order relating to symmetric pairs of variables,  $C_{11}, C_{21}, C_{22}, C_{31}, \dots$ . Subroutine `conres` calculates the residuals in (4.51) and stores them in `RES(:)` along with the term  $2f_{con} \llbracket \cdot \rrbracket$  in `RTS(:)` which is used later by Equation (4.53), where  $f_a$  is the factor  $f_{con}$ . The array `MAS` conveys whether or not the one or two-sided constraint in Equations (4.49) through (4.53) is to be used – the constraint residual is *not* set to zero if the structured variance is to be set (`MAS(ib,ia)` equals 1) or the total variance exceeds the maximum permissible in `UNS(ib,a)`. Subroutine `conjob` calculates the sub-matrix of Jacobian elements using Equation (4.53) and the  $2f_{con} \llbracket \cdot \rrbracket$  factors in `RTS(:)` and outputs the close packed matrix in `YAK`. By using `RTS`, the nugget effect modelling options are preserved – the zero terms in (4.51) carry through. As was the case for `jacob`, each row of `YAK` is set to zero before a call to another subroutine, `conjob` in loop 13. The subscripts  $\alpha$  and  $\beta$  in Equation (4.53) are communicated to `conrow` via `ia` and `ib` and the elements of `RTS` are multiplied by the appropriate elements in `EVEC` in `conrow`. Again, the values are accumulated so that the extra factor of two where  $\alpha = \beta$  is included.

Finally in module `krige`, are some routines for output of two-dimensional rasters of the modelled covariance function; `graphvar`, `graphcorg` and `graphvag`. Subroutine `graphvar` simply plots the  $n$ th variogram model in `krigrasta.txt`, over a square raster defined by the vectors `v1` (for the  $x$ -axis) and `v2` (for the  $y$ -axis). Subroutine `graphvag` is similar, but plots the total covariance for  $C_{ij}$  instead. Subroutine `graphcorg` takes a close-packed list of points in `PTS` consisting of contiguous blocks of spatial statistics in `NPT` (as was passed into `koreg`), and writes to `krigrasta.txt` a list of the spatial statistics and if `krdim==2`, plots the corresponding model covariance rasters using `graphvag`. The extents of these rasters correspond to the maximum and minimum extents of the coordinate lists in `PTS`. Another subroutine `saveme` writes the current covariance model to `krigdia.txt` if its input argument is 0, and reads a covariance model from `krigdia.txt` if its argument is 1.

## E - 6. Module optimisation

Module `optimisation` is used extensively by subroutines in module `krige` to perform least-squares optimisations using the Levenberg-Marquardt algorithm. The precise implementation of this algorithm is detailed in Section C - 2, here we shall describe the routines that effect it, using the same nomenclature introduced in C - 2 and in Section IV - 5.

Subroutine `lemarq` performs a single iteration of the Levenberg-Marquardt algorithm on the basis of the Jacobian passed in by the array `DYDX` and the residual vector `RES`. It passes out  $\delta \mathbf{r}$  in the vector `GO` with a number of other operational parameters. Subroutine `lemarq` has access to the values in the common block `OPTSCR`, where it stores values necessary for its execution. These values are initialised by subroutine `linit`, which must be called before any calls to `lemarq` are made. Subroutine `linit` takes three input arguments; `ndim`, the number of parameters (the length of  $\mathbf{r}$ ) in the optimisation; `nex`, the number of residuals in the sum of squares (the length of  $\mathbf{G}$ ); and `dampin`, the initial damping coefficient  $\kappa$ . It assigns these values to `npa`, `nex` and `damp` respectively in `OPTSCR` for future reference. It then initialises `real*8 sold` to zero. When it is used by `lemarq`, `sold` is the natural log of the total sum of squares one iteration ago,  $\ln(E_{rr}(\mathbf{r}_{k-1}))$ . The variable `sos` in `OPTSCR` is also set to zero – this is the natural log of the total sum of squares at the current iteration,  $\ln(E_{rr}(\mathbf{r}_k))$ , and it is reset at the beginning of each call to `lemarq`. The values `dia` and `dib` are initialised as `-1E150` and `-1E100`, two large negative numbers; note that `dia < dib`. When used by `lemarq`, `dia` is the difference between the log sum of squares, effectively  $\Delta_k$ . The values `dib` and `dic` are then respectively the differences  $\Delta_{k-1}$  and  $\Delta_{k-2}$ . Subroutine `linit` also sets the `logical*4` flags `ifa` and `ifb` to `.false.` and `.true.` respectively in accordance with the other values it has initialised. When used in `lemarq` the flag `ifa` is `.true.` if the reduction  $\Delta_k > \Delta_{k-1}$ , and the flag `ifb` is `.true.` if  $\Delta_{k-1} > \Delta_{k-2}$ . The iteration counter `iset` recording how many iterations ago the last change to the damping parameter `damp` occurred, is also set to zero.

As previously mentioned, subroutine `lemarq` takes the residual vector `RES(*)`, and the Jacobian matrix `DYDX(npa,*)` at the  $k^{\text{th}}$  iteration and returns the

solution step  $\delta \mathbf{r}_k$  in the vector GO. It also returns the values; `sumsq`, the sum of squares *before* the step in GO is taken; `damp`, the factor used at the  $k^{\text{th}}$  iteration  $\kappa_k$ ; `grd` the Euclidean norm of the gradient of the sum of squares function *before* the step GO is taken; and `ierr`, an error flag for the matrix solution. Subroutine `lemarq` firstly initialises some internal values and allocates space for the matrix solution in `SOL(np+1,np+1)`, a temporary storage space for the transpose of the Jacobian in `YACT(neg,np)` and swap space required later by `pcsol` in `JKOR(np,2)`. It then calculates the new value `sos` ( $\Delta_k$ ) from the input residual vector `RES`, works out the new value `dia` and flag `ifa` that result, and copies the previous value of `sos` to `sold`, `dia` to `dib`, `ifa` to `ifb` and so on. Having done this, the damping coefficient to use on the current iteration can be calculated. If the sum of squares has not improved between iterations, `dia` is greater than zero and the damping factor is greatly increased by the parameter `rin` ( $c_{div}$ ). If `ifa` and `ifb` are both true then the algorithm is performing optimally – corresponding to Case I Figure C—3, and the damping factor is not changed. However if the algorithm is slowing up or oscillating – corresponding to Case II Figure C—3 and Case III Figure C—3 respectively, the damping factor is reduced by factoring with the parameter `red` ( $c_-$ ) or increased by the factor `der` ( $c_+$ ) respectively. In either of these scenarios the counter `iset` is reset to zero, and for any change by these mechanisms `iset` must be greater than four iterations in the first place. Whatever happens, `iset` is then incremented by one after the damping coefficient `dam` is decided.

From this point in `lemarq` the Jacobian in `DYDX(np,*)` is rearranged into `YACT(neg,*)` for which the subscripts are reversed in loops 9 and 10. The matrix `SOL(1:np,1:np)` is populated by the result of  $[\mathbf{J}_i^T \mathbf{J}_i - \mathbf{I}]$  which is calculated in nested loops 13 and 15 of the larger loop 11. Note the damping factor `dam` is added after `continue 13` to the diagonal component `SOL(ia,ia)`. The gradient vector  $\mathbf{J}_i^T [\mathbf{G} - \mathbf{g}_i]$  on the right hand side of Equation (4.23) is calculated in loop 17 and assigned to `SOL(np+1,ia)`. Also in loop 11, the square of the Euclidean norm of the gradient is evaluated with reference to  $\mathbf{J}_i^T [\mathbf{G} - \mathbf{g}_i]$  and subsequently assigned to `grd`. The array `SOL` is then passed into the Gaussian elimination solver `pcsol` and the solution vector, resident in `SOL(np+1,1:np)` after this call is copied to GO. Note

that the space in `SOL(:,npa+1)` is only required internally by `pcsol`. If this routine returns an error (singular matrix) or there was an earlier allocation error, the flag `ierr` is set to one (zero otherwise).

## E - 7. Module `sample`

Module `sample` generates the spatial statistics introduced in Sections IV - 2, IV - 3 and IV - 4. It is somewhat labyrinthine as a result of many extensions and is a prime target for a rewrite, using a faster splitting algorithm. There are a great many online splitting algorithms – for example fuzzy sorts, which could do a much better job here, saving both computer time and coding complexity.

The module contains initialisation and destructor routines, subroutine `sampini` and `sampexi`. Subroutine `sampini` initialises the local spatial dimension in private integer `ndimd`, and also interrogates the file `krigin.txt` for the desired order of the spatial statistics which is stored in private integer `norder`. The integer `norder` can either be -1 for the correlation coefficient in Equation (4.5) or 0 for the variogram in Equation (4.7). It also opens the binary direct access file `kr02.bin` under the identifier 55. This is the file in which the list expressed by Equation (4.13) is stored, in records of length `mblok` bytes, a private parameter to module `sample`. It also initialises the common block `XINF`, setting the first two integers respectively to the maximum number of `real*8`, and the maximum number of items on the list in (4.13) allowed per record in `kr02.bin`. Note each item on this list will require `ndimd+2 real*8` to store it. The next four integers are set to zero. The common block `PRO` is also initialised to zero. Note that the values `rsos`, `rmean` and `nrtot` are only used by the routine `xgen` in `sample`, and in later execution the space in `PRO` is used differently. Subroutine `sampexi` is called when module `sample` is finished with to close the `kr02.bin` file and then delete it.

As described with regard to subroutines in module `dataset`, subroutines `xgen` and `flbuf` write the list in Equation (4.13) to the file `kr02.bin`, filling it up in a sequential manner. Subroutine `xgen` takes a list of `nu` nodes and data-points (items)

in  $U(\text{ndimd}+1, *)$  and  $nv$  items in  $V(\text{ndimd}+1, *)$  and then in loops 11 and 13 generates the list of lag vectors from  $U$  to  $V$ , and writes this with the values pertaining to each sequentially to the buffer  $BUF$  in  $XINF$ , incrementing the counter  $k$  which is also stored in  $XINF$ . When adding a new item to the list in  $BUF$  will exceed the available space per record  $mbok$  (a private parameter),  $BUF$  is written to the next record in  $kr02.bin$  by a call to `flbuf`. Note that the lists  $U$  and  $V$  are randomly reordered by the indexing scheme in  $IND$ , which is generated in loops 19 and 17 from a list of random numbers in  $ORD$ . For historical reasons only, the integer to initialise the random sequence  $iniran$  is passed into `xgen`. The flag `iflag` is set to zero when auto-covariant statistics are required so that later in loop 13, coincident nodes will not be added to the list. Note that also for auto-covariant structures, every positive lag vector is created with a negative one. The file  $kr02.bin$  holds the list in Equation (4.13) stored such that each record from the beginning holds as many complete  $(\text{ndimd}+2)$   $real*8$  segments (lag vector and two nodal values) as possible given a fixed record size, with the last record being only partially full. Additionally, as the list in  $kr02.bin$  is accrued, a running average and covariance is maintained in `rmean` and the private  $real*8$  `uvari`. Running average information is stored in `PRO` for this purpose.

Subroutine `flbuf` flushes  $BUF$  in  $XINF$  to the next record  $nrec$  in  $kr02.bin$ . It also increments  $nrec$  which is kept in  $XINF$  and writes the number of bytes and the number of items in the last record written to `LREC(1,2)` and `LREC(2,2)` respectively. In this way, when `flbuf` is called for the last time to flush the list  $BUF$  (which may not be full) `LREC(:,2)` contains the length of the last record in  $kr02.bin$ , and `LREC(:,1)` contains the length of all the other records (by its initialisation in `sampini`).

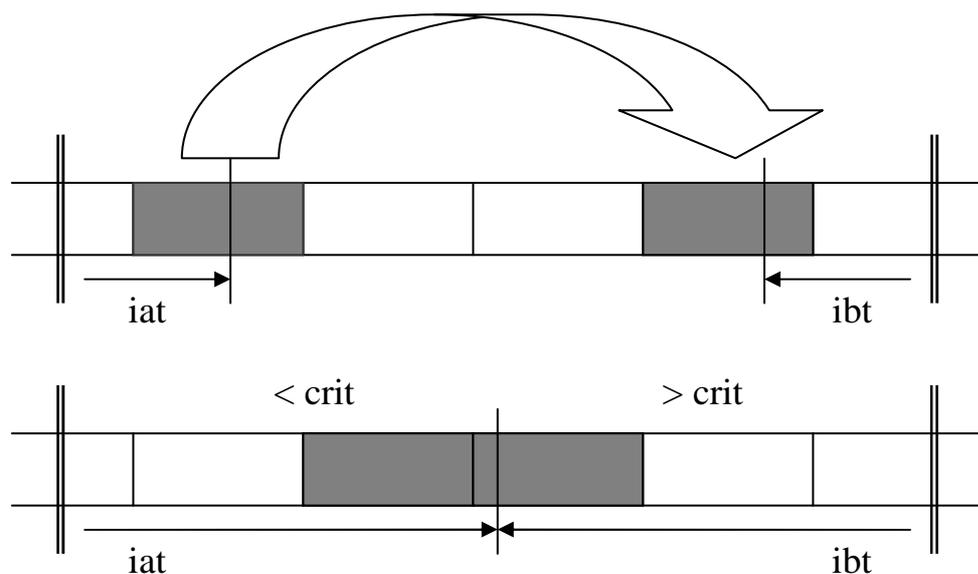
Subroutine `modtree` produces a list of at most  $nreq$  spatial statistics  $EXM(\text{ndimd}+1, *)$  where it is indicated that the data in the file  $kr02.bin$  are cross-covariant  $ix=1$ , or auto-covariant  $ix=0$ . It does this by splitting the file  $kr02.bin$  into groups as described in Section IV - 4.1. To do this efficiently, I/O to this file is managed by the routines `wirte` and `raed`, which use information about the number of groups in the splitting and the length of records in the file stored in the common

block `XINF`. It is for this purpose that the first five `integer*4` of `XINF` are set as detailed in the previous paragraph. Additionally, the number of tiles currently in the splitting is recorded by `ngp` in `XINF`. The routines `wirte` and `raed` maintain buffers of contiguous data read from, or to be written to `kr02.bin` resident in a common block `PRO`, in which is also recorded pointer and buffer information – `PRO` is initialised in a call to `bufini`.

The individual items in the list `kr02.bin` each have an address by which each block consisting of `ndimd+2 real*8`, is identified externally. Maintained in `modtree` is a list `ISP` where the elements in `ISP(:,k)` describe the addressing and properties of the  $k^{\text{th}}$  group. The list in `kr02.bin` is reordered in loop 11 so that the `ngp` tiles of lag vectors are located in contiguous blocks: the integer `ISP(1,k)` is the address in this list of the first lag in the  $k^{\text{th}}$  tile. The next integer `ISP(2,k)` describes how many such lags there are in the tile and `ISP(3,k)` is calculated to be the index of the spatial dimension with the greatest spread as measured by variance. Also calculated is a corresponding value `CRIT(k)` which is an estimate of the mode of the coordinate `ISP(3,k)` in the tile  $k$ . OK? The initial tile is actually the entire list, whose initial entry is in `ISP(:,1)`. The direction in which to split `ISP(3,1)` and the mode criterion for the split `CRIT(1)` are calculated by an initial call to subroutine `sorty`. `ISP(2,1)` is therefore the total number of lags on the list, so this is divided by `nreq`; the number of tiles required, to obtain `need`; the maximum number of lags in any one tile. Loop 13 in loop 11 finds the first group `ink` in the list `ISP` that needs splitting because it is too big, and splits it into two new groups in a call to `sortit`. These groups occupy `ISP(:,ink)` and `ISP(:,ink+1)`, so the subsequent groups are copied sequentially to locations further up the list before the call to `sortit` in loop 19. Once the file `kr02.bin` is reordered and the final list `ISP` obtained, spatial statistics are calculated for each tile in loop 17. Subroutine `svario` effects this if a variogram structure is desired and the statistics are auto-covariant, otherwise subroutine `sampnt` calculates a correlation coefficient. Finally `nreq` is set to the number of spatial statistics so generated by the `ngp` tiles.

Subroutine `sorty` writes the first group `ISP(:,1)`, passed into it as `ISP(*)` and the mode of the splitting coordinate `crit`. If `ix` is zero an auto-covariant structure requires that the first split is symmetrical so `crit=0.0`. The total number of points `ISP(2)` is first calculated from the file record information in `XINF`. Then in loop 11, each of these points is read in succession from `kr02.bin` via a call to `raed` which fetches the lag and associated values in `PT`. These points are copied to a buffer `BUFA` which is ‘emptied’ by a call to `flusta` when it reaches capacity. Subroutine `flusta` also resets `nba`, the number of lags in `BUFA`, to zero and calculates a statistical summary stored in the lists `STA` and `JSA` of `nsa` such summaries. The array `STA` contains the statistics (averages, variances and quartiles of each spatial coordinate) and the vector `JSA` stores the number of lags in each flush of `BUFA` supporting them. Upon exit of loop 11, `flusta` is called once more for any remaining items in `BUFA`. The `nsa` statistical summaries in `STA` and `JSA` are then collated by `calvar` which finds `ISP(3)`, the ordinate with the greatest spatial spread as measured by the total variance of all the summaries. This then allows a call to `calmod` to calculate `crit`, the mode of spatial ordinate `ISP(3)`.

Subroutine `sortit` splits the group `ISP(:,1)` into two groups according to `CRIT(1)` and (over-)writes them to the new groups `ISP(:,1)` and `ISP(:,2)`. Firstly, `sortit` sets `div` in `XINF` to the splitting criterion `CRIT(1)`, `max` to the maximum allowable number of lags in `BUFA` and `BUFB`, and `idi` to the splitting ordinate



**Figure E—6: Splitting a tile (between double lines) in a segment of `kr02.bin`.**

ISP(3,1). Communication of these values to `nxtswap` is via `XINF`. The pointers `iat` and `ibt` are set to point to the first lag in the tile and the last lag respectively, and the counters `npet` and `nbig` of elements that are respectively larger than the criteria, and less than the criteria, are set to zero. The number of lags in the buffers `BUFA` and `BUFB`, `nba` and `nbb`, are respectively set to zero, as are the numbers of flushes `nsa` and `nsb` made from these buffers to arrays of summary statistics in `STA` and `STB`. In loop 11 the pointer `iat` is advanced forward by `nxtswap` and `ibt` is moved backwards by `nxtswap` until a suitable pair of lags to swap is found. Lags for whom `QT(idi, :)` is smaller than `div` are moved towards the beginning of the list (smaller addresses) and lags for whom `PT(idi, :)` is larger than `div` are swapped to the end of the list (larger addresses), as shown in Figure E—6. When a suitable pair `PT` and `QT` is found, a call is made to `wirte` to swap their positions in the file, and the pointers `iat` and `ibt` are advanced. As they are advanced in `sortit` or in `nxtswap`, they copy the lags that will eventually arrive in each group to `BUFA` and `BUFB`, which are flushed to the statistics buffers `STA` and `STB` when they fill up. When the pointers `iat` and `ibt` finally cross over, loop 11 terminates and the groups are determined. Any remaining lags in `BUFA` and `BUFB` are flushed and the new pointers `ISP(1,1)` and `ISP(1,2)` are determined with the number of lags in each new tile `ISP(2,1)` and `ISP(2,2)`. The splitting directions `ISP(3,1)` and `ISP(3,2)` are then determined by calls to `calvar` which aggregate the statistical summaries in `STA` and `STB`. Similar calls to `calmod` then determine the mode of the ordinate; the new criteria `CRIT(1)` and `CRIT(2)`.

Subroutine `nxtswap` finds a lag `PT` at `ipt` to swap to the other end of the tile to be split. It first checks that the buffer `BUF` needs to be flushed to the summary statistics `AVAQ`, and then reads in `PT` from location `ipt` in `kr02.bin`. `PT(krit)` is then checked and depending on the direction in which `ipt` is advanced (incremented or decremented by `igo` which can be either +1 or -1) returns with `PT` if it is to be swapped to the other end or advances `ipt` and checks the next `PT(krit)` in loop 13. This is repeated until a lag is found for swapping – obviously as the criterion is the mode of the data, `ipt` cannot be advanced past the halfway point (roughly). However, `ipt` would advance past halfway if multiple `PT(krit)` equal to `crit` existed, except that a total of the number of lags `nat` in `nxtswap`'s end and lags `non` in the other end is also passed in and compared to stop one group getting bigger than

the other in such situations. Each new lag `PT` which is not to be swapped is added to `BUF`, which is flushed when full.

Much use is made of subroutine `flusta` to calculate intermediate statistics `AVAQ` from a buffer `BUF(ndimd+2,*)` containing `k` lags. These statistics are appended to the existing lists `AVAQ` and `NPB`. Firstly `nf` (the number of flushes) is incremented and then the support of the statistics is recorded in `NPB(nf)` and `workav` is called to calculate the statistics appended at `AVAQ(:,nf)`. Finally `k` is reset to zero.

Subroutine `workav` takes the `npt` lags in `PTS(ndimd+2,*)` and calculates seven statistics in `AVAQ(7,*)` for each of the `ndimd` spatial dimensions. For each spatial dimension in loop 11, it copies the ordinate from `PTS(ia,:)` to the array `REO`, which is reordered by calls to the `speed` routine `smallh` so that the quartiles are migrated to `REO(nq1)`, `REO(nq2)` and `REO(nq3)`. With the maximum value `pma` and minimum value `pmi` found in loop 13, these quartiles make up the last five values of each `7×real*8` block. The first two values are the mean and the unscaled variance, calculated in a call to the `speed` routine `pst2mo`.

Subroutine `calvar` is used to aggregate the `nsub` sets of `7×ndimd` statistics in `SUB(7,ndimd,*)` where each statistic `SUB(:, :, k)` has been generated from `NPT(k)` tile ordinates each. It finds the ordinate possessing the greatest spatial spread in terms of its aggregated variance and passes it out in `krit`. Loop 9 indexes the spatial ordinates and compares the aggregated variance `sig` against the biggest so far, `big`. The ordinate producing `big` is stored in `krit`. The aggregated variance is calculated by the weighted average `ave` of `SUB(1,ia,ib)` in loop 11, and the weighted sum of squares is calculated in loops 11 and then in loop 13. Finally the weighted sum of squares is factored by `1/(nptot-1)` for the final variance estimate `sig`.

Subroutine `calmod` aggregates `nst` sets of `7×ndimd` statistics in `AVAQ(7,ndimd,*)` where each statistic `AVAQ(:, :, k)` has been generated from `NPB(k)` tile ordinates each. This routine instead finds the mode `crit` of the `krit`<sup>th</sup> ordinate of the lag vectors. It constructs this from the quartiles in `AVAQ(3:7,krit,:)`, concatenating quartiles zero, one, two and three in `QUAR(:, :, 1)` and the concatenating quartiles one, two, three and four in

QUAR(:, :, 2). QUAR(:, :, 1) is then reordered with the new index order stored in IND(:) for reference. The now ordered list QUAR is then stepped through in loops 15 and 17 and tot is incremented from zero, sometimes fractionally as more quartile bands are passed. When tot meets the halfway mark at hwa, the loop terminates. The final fractional quartile is evaluated and passed out in crit.

As mentioned before, subroutine bufini initialises the space in PRO for use by the subroutines raed and wirte. The stride of the data nbit is set by nword (effectively ndimd+2), the arrays LIM and IREC are set entirely to zero and last is initialised to 2. The basic idea is to buffer binary records requested for a read or a write in one of two buffers BUF(:, 1) and BUF(:, 2) in PRO. The integer\*4 LIM(1, 1) and LIM(2, 1) describe the range of addresses available in BUF(:, 1) and LIM(1, 2) and LIM(2, 2) describe the range of addresses available in BUF(:, 2). The integer\*4 IREC(1) describes the record number of the data resident in BUF(:, 1) and IREC(2) describes the record number of the data resident in BUF(:, 2). The integer last is either 1 or 2 and is the last buffer to have been written to (or fetched).

The subroutines raed and wirte operate in a very similar manner. The only difference is that raed reads, and wirte writes. Both take an address iad and attempt to read or write to a buffer comprising the record that contains the address iad in kr02.bin. First, it is checked in loop 11 whether or not the desired record is already loaded, and if so in which buffer ibuf. If ibuf is non-zero, control skips to the end of the routine where PT is copied straight to or from BUF(iat, ibuf). If ibuf is zero then a new record must be fetched and the old one written back to kr02.bin. This is done in the following manner. If the desired address iad lies in a record somewhere inbetween the two records already fetched, itmp is less than zero and the buffer nearest to the desired address is written back out to disk and replaced by the incoming record. The nearest buffer is found by loop 13. However, if the desired address lies outside of this range, the buffer that was *not* accessed last – as not indicated by last, is flushed and replaced instead. This behaviour is designed so that when raed and wirte are used in tandem to access addresses that are working in towards each other, nearly optimal disk I/O is achieved. This is of course exactly the

situation arising in subroutine `sortit`. The addresses and record numbers in `PRO` are kept up to date by these routines, and discarded when they are no longer required.

Subroutine `centre` is used to calculate and remove a spatial shift or offset from the `nex` cross-covariant statistics passed in by the array `EXM(ndimd+1,*)`. This is done in a more or less heuristic manner at the moment, and may be improved in the future. Firstly the largest `npk` statistics in `EXM(ndimm,:)` are selected by copying them to `SOR(:)`, multiplying by `-1` and then using `smallh` to pick the smallest `npk`. Their original locations in `EXM` are stored in `KOR`. A matrix of distances between these `npk` spatial statistics is then created in loops 15 and 17. Twice the smallest distance squared between any two statistics in this matrix is stored in `rad`. In loops 19 and 21, each node is then weighted by cumulatively adding the covariance of its neighbours, *if* its neighbours lie within a radius `sqrt(rad)`. The total weight for each node is stored in `SCOR`, and a weighted average of the nodes is calculated for the centre in loop 23, and stored in `CNT`. This offset is then subtracted from the statistics in loop 13 and returned in the output `H`.

Subroutine `inlier` is designed to remove the outlying statistics heuristically from the set of spatial statistics in `EGS`. It assumes that the largest values for covariance should occur towards the origin of the covariance function. The squared distance to the origin is stored in `DIST(:)` and the corresponding covariance values `EGS(ndimd+1,:)` are stored in `VAR(:)` in loop 11, and the `IND` and `INV` index vectors for each are also initialised. `DIST` and `VAR` are then sorted in ascending order by speed routine `shells`, and the original positions are kept in `IND` and `INV`. Loops 13 and 15 then look through `INV`, and if any of the `mod` largest covariances at the tail of `INV` are found amongst the `mod` largest distances at the tail of `IND`, they are removed as outliers. The outliers are replaced by values we wish to keep in loop 17, the number of remaining statistics is returned in `neg`.

# Appendix F - Publications

Two significant papers produced during the candidature are included after this page, the second of which is at the time of writing accepted in the journal *Computers and Fluids*.