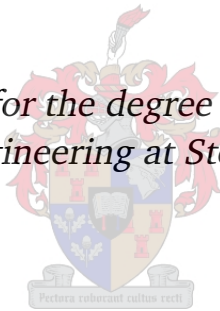


# Inverse modelling and optimisation in numerical groundwater flow models using proper orthogonal decomposition

by

John Nathaniel Wise

*Dissertation presented for the degree of Doctor of Philosophy  
in the Faculty of Engineering at Stellenbosch University*



Promoters:

Prof. Gerhard Venter  
Prof. Mireille Batton-Hubert

March 2015



UNIVERSITEIT  
STELLENBOSCH  
UNIVERSITY



NNT: 2015 EMSE 0773

## THESIS DISSERTATION

presented by

**John N. Wise**

to obtain a Doctorate Degree from the Ecole des Mines de Saint Etienne in  
collaboration with Stellenbosch University.

Speciality: Environmental Science

### Inverse modelling and optimisation in numerical groundwater flow models using proper orthogonal decomposition

Presented at Saint-Etienne, December, 2014

#### Jury

Reviewer/examiners:	Rachid ABABOU	Professor, University of Toulouse, (INPT)
	Emmanuel LERICHE	Professor, University of Lille I
Examiners:	Albert A. Groenwold	Professor, University of Stellenbosch
Thesis directors:	Mireille BATTON-HUBERT	Professor, Ecole des Mines de Saint-Etienne
	Gerhard VENTER	Professor, University of Stellenbosch
	Eric TOUBOUL	Researcher, Ecole des Mines de Saint-Etienne
Invited	Asdin Aoufi	Researcher, Ecole des Mines de Saint-Etienne

# Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: ..... 2015/01/12 .....

Copyright © 2015 Stellenbosch University  
All rights reserved.

# Abstract

## **Inverse modelling and optimisation in numerical groundwater flow models using proper orthogonal decomposition**

J. N. Wise

Dissertation: PhD (Env. Eng.)

March 2015

Numerical simulations are widely used for predicting and optimising the exploitation of aquifers. They are also used to determine certain physical parameters, for example soil conductivity, by inverse calculations, where the model parameters are changed until the model results correspond optimally to measurements taken on site. The Richards' equation describes the movement of an unsaturated fluid through porous media, and is characterised as a non-linear partial differential equation. The equation is subject to a number of parameters and is typically computationally expensive to solve. To determine the parameters in the Richards' equation, inverse modelling studies often need to be undertaken. In these studies, the parameters of a numerical model are varied until the numerical response matches a measured response. Inverse modelling studies typically require 100's of simulations, which implies that parameter optimisation in unsaturated case studies is common only in small or 1D problems in the literature.

As a solution to overcome the computational expense incurred in inverse modelling, the use of Proper Orthogonal Decomposition (POD) as a Reduced Order Modelling (ROM) method is proposed in this thesis to speed-up individual simulations. An explanation of the Finite Element Method (FEM) is given using the Galerkin method, followed by a detailed explanation of the Galerkin POD approach. In the development of the Galerkin POD approach, the method of reducing matrices and vectors is shown, and the treatment of Neumann and Dirichlet boundary values is explained.

The Galerkin POD method is applied to two case studies. The first case study is the Kogelberg site in the Table Mountain Group near Cape Town in South Africa. The response of the site is modelled at one well over the period of 2 years, and is assumed to be governed by saturated flow, making it a linear problem. The site is modelled as a 3D transient, homogeneous site, using 15 layers and  $\approx 20000$  nodes, using the FEM implemented on the open-source software FreeFem++.

The model takes the evapotranspiration of the fynbos vegetation at the site into consideration, allowing the calculation of annual recharge into the aquifer. The ROM is created from high-fidelity responses taken over time at different parameter points, and speed-up times of  $\approx 500$  are achieved, corresponding to speed-up times found in the literature for linear problems. The purpose of the saturated groundwater model is to demonstrate that a POD-based ROM can approximate the full model response over the entire parameter domain, highlighting the excellent interpolation qualities and speed-up times of the Galerkin POD approach, when applied to linear problems.

A second case study is undertaken on a synthetic unsaturated case study, using the Richards' equation to describe the water movement. The model is a 2D transient model consisting of  $\approx 5000$  nodes, and is also created using FreeFem++. The Galerkin POD method is applied to the case study in order to replicate the high-fidelity response. This did not yield in any speed-up times, since the full matrices of non-linear problems need to be recreated at each time step in the transient simulation.

Subsequently, a method is proposed in this thesis that adapts the Galerkin POD method by linearising the non-linear terms in the Richards' equation, in a method named the Linearised Galerkin POD (LGP) method. This method is applied to the same 2D synthetic problem, and results in speed-up times in the range of 10 to 100. The adaptation, notably, does not use any interpolation techniques, favouring a code intrusive, but physics-based, approach. While the use of an intrusively linearised POD approach adds to the complexity of the ROM, it avoids the problem of finding kernel parameters typically present in interpolative POD approaches.

Furthermore, the interpolation and possible extrapolation properties inherent to intrusive POD-based ROM's are explored. The good extrapolation properties, within predetermined bounds, of intrusive POD's allows for the development of an optimisation approach requiring a very small Design of Experiments (DOE) sets (e.g. with improved Latin Hypercube sampling). The optimisation method creates locally accurate models within the parameter space using Support Vector Classification (SVC). The region inside of the parameter space in which the optimiser is allowed to move is called the confidence region. This confidence region is chosen as the parameter region in which the ROM meets certain accuracy conditions. With the proposed optimisation technique, advantage is taken of the good extrapolation characteristics of the intrusive POD-based ROM's. A further advantage of this optimisation approach is that the ROM is built on a set of high-fidelity responses obtained prior to the inverse modelling study, avoiding the need for full simulations during the inverse modelling study.

In the methodologies and case studies presented in this thesis, initially infeasible inverse modelling problems are made possible by the use of the POD-based ROM's. The speed up times and extrapolation properties of POD-based ROM's are also shown to be favourable.

In this research, the use of POD as a groundwater management tool for

saturated and unsaturated sites is evident, and allows for the quick evaluation of different scenarios that would otherwise not be possible. It is proposed that a form of POD be implemented in conventional groundwater software to significantly reduce the time required for inverse modelling studies, thereby allowing for more effective groundwater management.

# Uittreksel

## **Inverse modelering en optimeering van numeriese grondwater vloeï models deur die gebruik van eie ortogonale ontbinding**

*(“Inverse modelling and optimisation in numerical groundwater flow models using proper orthogonal decomposition”)*

J. N. Wise

Proefskrif: PhD (Omgewings Ingenieurswese)

Maart 2015

Die Richards vergelyking beskryf die beweging van ’n vloeistof deur ’n onversadigde poreuse media, en word gekenmerk as ’n nie-lineêre partiële differensiaalvergelyking. Die vergelyking is onderhewig aan ’n aantal parameters en is tipies berekeningsintensief om op te los. Om die parameters in die Richards vergelyking te bepaal, moet parameter optimering studies dikwels onderneem word. In hierdie studies, word die parameters van ’n numeriese model verander totdat die numeriese resultate die gemete resultate pas. Parameter optimering studies vereis in die orde van honderde simulasies, wat beteken dat studies wat gebruik maak van die Richards vergelyking net algemeen is in 1D probleme in die literatuur.

As ’n oplossing vir die berekeningskoste wat vereis word in parameter optimering studies, is die gebruik van Eie Ortogonale Ontbinding (POD) as ’n Verminderde Orde Model (ROM) in hierdie tesis voorgestel om individuele simulasies te versnel in die optimering konteks. Die Galerkin POD benadering is aanvanklik ondersoek en toegepas op die Richards vergelyking, en daarna is die tegniek getoets op verskeie gevallestudies.

Die Galerkin POD metode word gedemonstreer op ’n hipotetiese gevallestudie waarin water beweging deur die Richards-vergelyking beskryf word. As gevolg van die nie-lineêre aard van die Richards vergelyking, het die Galerkin POD metode nie gelei tot beduidende vermindering in die berekeningskoste per simulatie nie. ’n Verdere gevallestudie word gedoen op ’n ware grootskaalse terrein in die Tafelberg Groep naby Kaapstad, Suid-Afrika, waar die grondwater beweging as versadig beskou word. Weens die lineêre aard van die vergelyking wat die beweging van versadigde water beskryf, is merkwaardige versnellings van  $> 500$  in die ROM waargeneem in hierdie gevallestudie.

Daarna was die die Galerkin POD metode aangepas deur die nie-lineêre terme in die Richards vergelyking te lineariseer. Die tegniek word die geLineariseerde Galerkin POD (LGP) tegniek genoem. Die aanpassing het goeie resultate getoon, met versnellings groter as 50 keer wanneer die ROM met die oorspronklike simulatie vergelyk word. Al maak die tegniek gebruik van verder lineariseering, is die metode nogsteeds 'n fisika-gebaseerde benadering, en maak nie gebruik van interpolasie tegnieke nie. Die gebruik van 'n fisika-gebaseerde POD benaderings dra by tot die kompleksiteit van 'n volledige numeriese model, maar die kompleksiteit is geregverdig deur die merkwaardige versnellings in parameter optimerings studies.

Verder word die interpolasie eienskappe, en moontlike ekstrapolasie eienskappe, inherent aan fisika-gebaseerde POD ROM tegnieke ondersoek in die navorsing. In die navorsing word 'n tegniek voorgestel waarin hierdie inherente eienskappe gebruik word om plaaslik akkurate modelle binne die parameter ruimte te skep. Die voorgestelde tegniek maak gebruik van ondersteunende vektor klassifikasie. Die grense van die plaaslik akkurate model word 'n vertrouens gebied genoem. Hierdie vertrouens gebied is gekies as die parameter ruimte waarin die ROM voldoen aan vooraf uitgekiesde akkuraatheidsvereistes. Die optimeeringsbenadering vermy ook die uitvoer van volledige simulaties tydens die parameter optimering, deur gebruik te maak van 'n ROM wat gebaseer is op die resultate van 'n stel volledige simulaties, voordat die parameter optimering studie gedoen word. Die volledige simulaties word tipies uitgevoer op parameter punte wat gekies word deur 'n proses wat genoem word die ontwerp van eksperimente.

Verdere hipotetiese grondwater gevallestudies is onderneem om die LGP en die plaaslik akkurate tegnieke te toets. In hierdie gevallestudies is die grondwater beweging weereens beskryf deur die Richards vergelyking. In die gevalle studie word komplekse en tyd-rowende modellerings probleme vervang deur 'n POD gebaseerde ROM, waarin individuele simulaties merkwaardig vinniger is. Die spoed en interpolasie/ekstrapolasie eienskappe blyk baie gunstig te wees.

In hierdie navorsing is die gebruik van verminderde orde modelle as 'n grondwaterbestuursinstrument duidelik getoon, waarin voorsiening geskep word vir die vinnige evaluering van verskillende modellerings situasies, wat andersins nie moontlik is nie. Daar word voorgestel dat 'n vorm van POD in konvensionele grondwater sagteware geïmplementeer word om aansienlike versnellings in parameter studies moontlik te maak, wat na meer effektiewe bestuur van grondwater sal lei.



# Acknowledgements

I would like to acknowledge the CSIR for their assistance in acquiring field data, and for Umvoto Africa for extended leave to complete the research.

# Dedications

*Dedicated to Gabrielle and Adrienne Lupion for their support and undeserved hospitality throughout my time in France, their faith in God and their lives example.*

# Contents

<b>Declaration</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Uittreksel</b>	<b>v</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>Dedications</b>	<b>viii</b>
<b>Contents</b>	<b>ix</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xvi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Goals . . . . .	2
1.2 Context and contribution of work . . . . .	3
1.3 Document layout . . . . .	4
<b>2 Finite Element discretisation of groundwater mass flow equations</b>	<b>6</b>
2.1 Developing the mass flow equation . . . . .	7
2.1.1 The mass flow equation . . . . .	7
2.1.2 Simplifications . . . . .	8
2.1.3 Mass flow in terms of pressure head . . . . .	10
2.1.4 Adaptation for saturated flow . . . . .	11
2.1.5 Adaptation to unsaturated flow . . . . .	11
2.1.6 Non-linear terms . . . . .	12
2.2 Applying the Finite Element Method . . . . .	13
2.2.1 Weak formulation . . . . .	15
2.2.2 Discretisation of unsaturated flow equation . . . . .	16
2.2.3 Matrix representation . . . . .	17
2.2.4 Treatment of the time term . . . . .	17
2.3 Applying Dirichlet boundary conditions . . . . .	18
2.3.1 Enforcing the Dirichlet boundary condition . . . . .	20

2.3.2	Alternate approach to treating the Dirichlet boundary . . .	21
2.4	Unsaturated flow example . . . . .	21
2.5	Conclusion . . . . .	23
<b>3</b>	<b>Proper Orthogonal Decomposition</b>	<b>24</b>
3.1	Background and literature review . . . . .	25
3.1.1	POD in groundwater problems . . . . .	25
3.1.2	POD in inverse modelling studies . . . . .	26
3.2	Model reduction with Proper Orthogonal Decomposition . . . . .	27
3.2.1	Choosing representative snapshots . . . . .	27
3.2.2	Galerkin POD method . . . . .	29
3.2.3	Reducing full Finite-Element matrices . . . . .	32
3.2.4	Reducing the vector terms . . . . .	33
3.2.5	Reduction of Dirichlet and gravity mean terms . . . . .	34
3.2.6	Solving the reduced model equation . . . . .	35
3.2.7	Error calculation . . . . .	35
3.3	Benchmark example of van Genuchten . . . . .	36
3.4	Conclusion . . . . .	40
<b>4</b>	<b>Applying POD to saturated and unsaturated groundwater case studies</b>	<b>41</b>
4.1	Saturated case study: Table Mountain Group Aquifer . . . . .	42
4.1.1	Site description and assumptions . . . . .	42
4.1.2	System response and evapotranspiration . . . . .	43
4.1.3	Soil parameters . . . . .	46
4.1.4	Model construction . . . . .	47
4.1.5	Numerical modelling . . . . .	47
4.1.6	Application of Galerkin POD approach to Darcy flow . . . . .	49
4.1.7	Optimisation definition . . . . .	50
4.1.8	Optimisation using a POD surrogate model . . . . .	51
4.1.9	Results of Kogelberg case study . . . . .	53
4.1.10	Conclusion . . . . .	55
4.2	Synthetic unsaturated case study . . . . .	56
4.2.1	Problem definition . . . . .	56
4.2.2	Time and accuracy results . . . . .	57
4.3	Conclusion . . . . .	59
<b>5</b>	<b>Improving CPU performance using the Hybrid and Linearised-POD methods</b>	<b>61</b>
5.1	Hybrid approach to treatment of non-linear terms . . . . .	62
5.1.1	Linearising the non-linear terms . . . . .	62
5.1.2	Creating a metamodel for the non-linear terms . . . . .	64
5.1.3	Matrix reduction with the Hybrid approach . . . . .	66
5.1.4	Reducing vector terms using the Hybrid approach . . . . .	68
5.1.5	Solving for pressure head with the Hybrid approach . . . . .	70

5.2	The Linearised POD approach . . . . .	71
5.2.1	Finding weight vectors . . . . .	72
5.2.2	Solving for pressure head with the LGP method . . . . .	72
5.3	Example evaluation of Hybrid and LGP approaches . . . . .	75
5.3.1	Weight values, eigenvalues and eigenvectors . . . . .	76
5.3.2	Model reduction using the Hybrid method . . . . .	77
5.3.3	Model reduction using the LGP method . . . . .	78
5.3.4	Accuracy and time results . . . . .	80
5.4	Discussion on the Hybrid and LGP approaches . . . . .	80
5.5	Conclusion . . . . .	82
<b>6</b>	<b>Inverse modelling and updating strategies</b>	<b>83</b>
6.1	Trust region approach . . . . .	84
6.1.1	Creating a quadratic approximation . . . . .	86
6.1.2	Updating the trust region . . . . .	86
6.1.3	Trust region POD . . . . .	88
6.2	Enrichment of POD basis set . . . . .	89
6.2.1	Enrichment from one new response . . . . .	90
6.2.2	Enrichment from a new set of responses . . . . .	90
6.3	Defining a confidence region with SVC . . . . .	91
6.3.1	Using SVC to define a confidence region . . . . .	92
6.3.2	Support Vector Classification: Linear case . . . . .	93
6.3.3	Non-linear mapping using kernels . . . . .	94
6.3.4	Dual optimisation problem . . . . .	95
6.3.5	Dealing with the discontinuous constraint boundary . . . . .	96
6.3.6	Implementation of the SVC-POD . . . . .	97
6.4	Conclusion . . . . .	99
<b>7</b>	<b>Optimisation case studies</b>	<b>100</b>
7.1	Problem description . . . . .	100
7.2	Example: Direct optimisation with SVC-POD . . . . .	101
7.2.1	Extrapolation with SVC-POD . . . . .	102
7.2.2	Optimisation with SVC-POD . . . . .	103
7.3	Example: Direct optimisation with TRPOD . . . . .	106
7.3.1	Adaptation of the TRPOD method . . . . .	106
7.3.2	TRPOD results . . . . .	107
7.4	Example: Inverse modelling with 5 design variables . . . . .	109
7.5	Comparison between TRPOD and SVC-POD . . . . .	111
7.6	Conclusion . . . . .	112
<b>8</b>	<b>Conclusion</b>	<b>114</b>
8.1	Application of POD to groundwater models . . . . .	114
8.2	Extrapolation and optimisation with POD . . . . .	114
8.3	POD in coupled surface-groundwater problems . . . . .	115
8.4	Local updating of POD models . . . . .	115

<i>CONTENTS</i>	<b>xii</b>
8.5 Future work . . . . .	116
<b>Appendices</b>	<b>117</b>
<b>A Gradient based numerical optimisation</b>	<b>118</b>
A.1 Optimisation problem . . . . .	118
A.2 Sequential Quadratic Programming . . . . .	119
A.3 Modified Method of Feasible Directions (MMFD) . . . . .	119
A.4 Design of experiments . . . . .	120
<b>B Code segments</b>	<b>122</b>
<b>C Evapotranspiration</b>	<b>124</b>
C.1 Adjusting the $ET_o$ with a single crop factor . . . . .	124
C.2 Adjusting the $ET_o$ with dual crop factors . . . . .	125
<b>List of References</b>	<b>128</b>

# List of Figures

1.1	Document layout . . . . .	5
2.1	Simplified layout of water cycle, showing the saturated and unsaturated zones. . . . .	6
2.2	Control volume showing fluid flow through a porous medium . . . . .	8
2.3	Hydraulic head in relation to pressure and elevation head (Diersch (2014))	10
2.4	Example of the change in moisture capacity $C(h)$ and conductivity $K(h)$ with a change in head pressure . . . . .	13
2.5	Example of a model domain ( $\Omega$ ) and boundary conditions . . . . .	15
2.6	Fixed point form method for iterative convergence . . . . .	18
2.7	One dimensional grid showing the shape functions and boundary values	19
2.8	Head pressure development over time after infiltration, showing a replication of the benchmark simulation done by Celia and Bouloutas (1990). The time in days is shown on the plots, showing infiltration and the corresponding conductivity and moisture content values at $t=0, 0.25, 0.5, 0.75$ and 1 day . . . . .	22
2.9	The convergence of a typical time step in the simulation using the Picard convergence scheme . . . . .	23
3.1	Snapshot set being projected onto two eigenvectors . . . . .	28
3.2	One dimensional grid with Dirichlet boundary values separated from the pressure head response . . . . .	30
3.3	The benchmark problem as presented by van Genuchten (1978). The different layers of soil are shown with labels corresponding to Table 3.1, and the soil moisture increasing with time as the wetting front descends . . . . .	37
3.4	A replication of the van Genuchten problem with the FE method, derived by implementing the method presented in this research in FreeFem++, and the subsequent ROM results using POD . . . . .	39
4.1	Perspectives of Kogelberg site . . . . .	44
4.2	Response in water table corresponding to rainfall . . . . .	45
4.3	Stress factors due to limited water availability . . . . .	45
4.4	Adaptation of crop factor . . . . .	46

4.5	Potential evapotranspiration calculated with the Penman-Monteith method . . . . .	47
4.6	Domain of numerical model showing the mesh and boundary conditions	48
4.7	Error graph showing the mean, maximum and minimum RRMS errors yielded on the validation set, when using the POD model created with the training parameter set . . . . .	52
4.8	Eigenvectors showing spatial variation of head pressure at the Kogelberg site . . . . .	52
4.9	Resulting response at well $K_2$ after inverse estimation study . . . . .	54
4.10	Evapotranspiration response for 2008 and 2009, where <b>2008*</b> indicates that the $ET_0$ for 2008 is assumed to be the same as the $ET_0$ of 2009, due to data lacking in 2008 . . . . .	55
4.11	Numerical case study mesh with $\approx 5000$ node points. The parabolic upper surface represents the soil topography . . . . .	57
4.12	Initial and final head pressure given in centimetres . . . . .	58
4.13	The first three pressure head eigenvectors derived from the snapshot set, showing the principal variations in the system occurring around the Neumann boundaries. The figures correspond to the problem in Fig. 4.12, and are plotted at an angle to view the variation. . . . .	58
4.14	Time and error vs. the number of shape function per ROM evaluation using the GPOD approach, where RRMS is the relative root mean squared error (Eq. 3.40). The original simulation requires 140 s. . . . .	59
4.15	Pie charts showing time, in seconds, required per function evaluation using 10 shape functions. The computational expense shifts from inverting the matrix to creating and reducing the matrix when creating the ROM. . . . .	60
5.1	Projecting snapshots onto a reduced plane . . . . .	63
5.2	Interpolation of the $i^{th}$ weight value between a series of training points, and the weight function approximation at a new parameter set $\mathbf{p}_{new}$ . . . . .	64
5.3	Approximating a weight function $\alpha^f$ with Radial Basis Functions . . . . .	66
5.4	The linearisation of the mass matrix, where the weight value vector $\hat{\alpha}^K$ is the unknown . . . . .	68
5.5	Fixed point form method for iterative convergence . . . . .	73
5.6	Numerical case study mesh with $\approx 5000$ node points . . . . .	75
5.7	Exponentially decreasing eigenvalues resulting from the PCA on the normalised snapshot sets . . . . .	76
5.8	First three pressure head eigenvectors . . . . .	76
5.9	First three moisture content eigenvectors . . . . .	77
5.10	First three conductivity eigenvectors . . . . .	77
5.11	Weight values over time corresponding to the eigenvectors. These values are found by projecting the true response onto the derived eigenvectors directly. . . . .	79
5.12	Comparison of time and accuracy using an increasing number of GSF's	81



6.1	Model updating using the trust region POD approach . . . . .	85
6.2	Changes in the trust radius in the parameter space from $\delta_1$ to $\delta_2$ , and changes in quadratic model initial value, moving from $\mathbf{p}_1$ to $\mathbf{p}_2$ . . . . .	87
6.3	One dimensional example of the trust region method . . . . .	88
6.4	Model updating using stored high-fidelity responses to create a locally accurate ROM. In this flow-chart, $\hat{h}$ refers to the approximated head pressure response from the ROM. . . . .	92
6.5	Creating a linear separating hyperplane between two data sets . . . . .	94
6.6	Comparison of the use of a linear and radial base kernel for the same set of data that is not linearly separable . . . . .	96
6.7	Example in which a starting point $\mathbf{p}_1$ has a SVC boundary defined as $\rho_1^{SVC}$ . Within the boundary, an optimum $\mathbf{p}_2$ is found. The ROM is then recreated at $\mathbf{p}_2$ , and a new confidence region $\rho_2^{SVC}$ is defined. . . . .	98
7.1	Mesh and location of boundary conditions of case study . . . . .	101
7.2	The two-parameter Design of Experiment points across the design space, using a latin hypercube generated in DOT Vanderplaats (2001) . . . . .	102
7.3	Classification of the regions of high and low accuracy . . . . .	104
7.4	Example showing the iterative updating of the confidence region as the optimiser moves through the design space . . . . .	105
7.5	Absolute error $ \mathbf{h}_{nts} - \hat{\mathbf{h}}_{nts} $ at the final time step . . . . .	106
7.6	Movement of the trust region through the design space, showing the movements at iteration 1, 3, 7 and 9 . . . . .	108
7.7	Water level approximations and corresponding adjustment in trust radius . . . . .	109
A.1	Latin Hypercube designs using 9 design points . . . . .	121
A.2	Optimum Latin Hypercube design generated by VisualDOC (Vanderplaats (2001)) . . . . .	121
B.1	FreeFem++ code segment solving the Richards' equation . . . . .	122
B.2	Implementation of the Galerkin-POD method in FreeFem++ . . . . .	123
C.1	Value of true evapotranspiration derived from $K_c$ and $K_s$ factors (Allen (2000)) . . . . .	126
C.2	Crop factor and transpiration stress factor . . . . .	126
C.3	Soil evaporation reduction coefficient . . . . .	127

# List of Tables

2.1	Different approximations of $s_e - h$ (from Todd and Mays (2005) and Diersch (2014)) . . . . .	14
3.1	Parameters used by van Genuchten (1978) . . . . .	38
4.1	The bounds of the design variables . . . . .	51
4.2	Results of parameter optimisation . . . . .	53
4.3	Time and error results . . . . .	53
4.4	Recharge results . . . . .	54
5.1	Example of training inputs and outputs for the Hybrid method . . . . .	64
7.1	SVC-POD results using 20 global shape functions . . . . .	106
7.2	TRPOD results using 20 global shape functions . . . . .	108
7.3	The bounds of the design variables . . . . .	110
7.4	Inverse optimisation results . . . . .	111
7.5	Discussion of differences between the TRPOD and SVC-POD methods . . . . .	112
C.1	Parameters in Penman-Monteith equation . . . . .	125

# Nomenclature

$\alpha$	Weight values for pressure head
$\alpha^C$	Weight values for $C$
$\alpha^K$	Weight values for $K$
$\delta t$	Time interval
$\delta_k$	Trust radius at iteration $k$
$\hat{h}$	Pressure head approximated by a ROM
$\hat{M}^P$	Reduced stiffness matrix
$\hat{M}^S$	Reduced mass matrix
$\Psi$	Eigenvector for pressure head in vector form
$\Psi^C$	Eigenvector for moisture storage in vector form
$\Psi^K$	Eigenvector for conductivity in vector form
$\mathbf{B}$	Hessian matrix
$\mathbf{C}$	Moisture content snapshot set
$\mathbf{G}^C$	Mean of $\mathbf{C}$
$\mathbf{G}^h$	Mean of $\mathbf{H}$
$\mathbf{G}^K$	Mean of $\mathbf{K}$
$\mathbf{H}$	Pressure head snapshot set
$\mathbf{h}$	Pressure head vector
$\mathbf{K}_s$	Conductivity vector
$\mathbf{K}$	Conductivity snapshot set
$\mathbf{M}^P$	Stiffness matrix

$\mathbf{M}^s$	Mass matrix
$\mathbf{p}$	Vector containing design parameters
$\mathbf{p}^l$	Lower limits design parameters
$\mathbf{p}^u$	Upper limits of design parameters
$\mathbf{s}$	Step vector
$\mathbf{U}^C$	Set of global shape functions from $\mathbf{C}$
$\mathbf{U}^h$	Set of global shape functions from $\mathbf{H}$
$\mathbf{U}^K$	Set of global shape functions from $\mathbf{K}$
$\mathbf{V}^n$	Neumann vector
$\mathbf{w}$	Gradient of linear hyperplane in SVC
$\mathbf{x}$	Spatial axes
$\Psi$	Eigenvector function for pressure head
$\Psi^C$	Eigenvector function for moisture content
$\Psi^K$	Eigenvector function for conductivity
$\rho_k^{SVC}$	SVC confidence region
$\sigma$	Parameter in radial basis function
$\tau$	Time step index
$\varrho$	SVC Lagrange multipliers
$\xi$	Measure of misclassification error in SVC
$b^h$	Piecewise continuous Dirichlet function
$b_{svc}$	Bias in SVC derivation
$C$	Moisture content
$C_{svc}$	Penalty value for outliers in SVC
$ET_c$	Actual evapotranspiration
$ET_o$	Potential evapotranspiration
$ET_{total}$	Total yearly evapotranspiration
$F(\mathbf{p})$	True function response at $\mathbf{p}$

$F^M(\mathbf{p})$	Reduced order response $\mathbf{p}$
$f_n$	Neumann function
$h$	Pressure head
$h_{wt}^M$	ROM water table height
$h_b$	Bubbling capillary pressure
$h_e$	Elevation head
$h_{init}$	Initial value of pressure head
$h_{wt}$	Full model water table height
$K$	Conductivity
$k(\mathbf{p}, \mathbf{p}_i)$	Kernel mapping function in SVC
$K_e$	Evaporation scaling value
$K_r$	Evaporation limiting function
$K_s$	Transpiration limiting function
$K_{c,var}$	Transpiration scaling function
$K_c$	Transpiration scaling value
$m^C$	Number of GSF's to approximate $C$
$m^h$	Number of pressure head GSF's
$m^K$	Number of GSF's to approximate $K$
$n_{calls}$	Number of function calls
$n_{close}$	Number of closest points to the initial parameter value
$n_{it}$	Number of iterations
$n_{par}$	Number of parameter values in $\mathbf{p}$
$n_{SV}$	Number of support vectors
$n_{TP}$	Number of training points
$nts$	Number of time steps
$Q_k(\mathbf{p})$	Quadratic approximation at $\mathbf{p}$
$RAW$	Readily available water

<i>REW</i>	Readily evaporable water
$S_s$	Specific storage
$t_{HF}$	Time per high fidelity simulation
$t_{iter}$	Time per iteration
$t_{opt}$	Time for optimisation
$t_{ROM}$	Time per ROM simulation
<i>TAW</i>	Total available water
<i>TEW</i>	Total evaporable water
$v_i$	Piecewise continuous basis function
$y_i$	Training point response $i$ in SVC
FEM	Finite Element Method
GIS	Geographic Information System
GPOD	Galerkin POD
GSF	Global Shape Function
I	Precipitation
LGP	Linear Galerkin POD
LH	Latin Hypercube
m.a.s.l.	Meters above sea level
OLH	Optimum Latin Hypercube
PCA	Principal Component Analysis
PDE	Partial differential equation
POD	Proper Orthogonal Decomposition
R	Runoff
RBF	Radial Basis Function
RBS	Reduced Basis Set
ROM	Reduced Order Model
RRMS	Relative root mean squared

SQP	Sequential Quadratic Programming
SVC	Support Vector Classification
SVD	Singular Value Decomposition
TMG	Table Mountain Group
TRPOD	Trust Region POD

# Chapter 1

## Introduction

**T**HE use of numerical models for water resource management is becoming increasingly important as already limited water resources are placed under increasing pressure. Numerical models are typically used as a management tool of groundwater systems, where advanced analyses, such as inverse modelling or parameter optimisation, can be done. However, the numerical models are often computationally infeasible for these advanced analyses, due to long run times.

A significant advantage of using accurate numerical models of groundwater resources, is that invasive and expensive measuring and tests on the aquifer can be minimised. Using numerical models, unknown soil and infiltration parameters can be approximated by means of inverse modelling, or parameter calibration. The numerical groundwater models are dependent on a number of parameters, such as conductivity, which are not always measurable due to the cost of sampling and other constraints. In order to estimate the unknown parameters, a response that depends on the unknown parameters are measured. In the numerical model, these unknown parameters can be varied according to an optimisation algorithm, until the numerical response adequately approximates the measured response.

Furthermore, well placement and extraction rates at wells can be optimised when using a numerical model. Inverse modelling studies typically require in the order of 100's of function evaluations to find an optimum in the parameter space. The time and effort required to conduct inverse modelling studies with computationally expensive models can lead to these studies being ignored, or superficially executed

In this thesis, we propose the use of Reduced Order Model (ROM) methodologies to accelerate inverse modelling studies, by accelerating individual simulations. The ROM methodologies are developed and applied to the equations describing groundwater flow, particularly the Richards' equation. The Richards' equation is a non-linear Partial Differential Equation (PDE), describing water movement in unsaturated porous media.

The reduced order models used and developed in this thesis are based on Proper Orthogonal Decomposition (POD). This method typically captures prevalent behaviour of discretised systems in a spatial sense, and recreates a model



based on the governing PDE. In this way the physics and dynamics of the entire system can be preserved, which is typically not the case in other metamodelling techniques such as kriging. In the POD approaches presented in this research, the use of interpolative techniques such as kriging are avoided in order to avoid the complexity in calibrating that is required when using interpolative techniques. Each POD-based ROM can serve as a metamodel, and by substituting the full model with the ROM, significant speed up times in inverse modelling and optimisation studies can be achieved.

In this thesis, two areas of focus exist. The focus is initially on the limitation, in terms of speed-up times, of the classical application of POD method when applied to the Richards' equation. This limitation is addressed by developing an alternative implementation of the POD based model. The second focus area is on the use of POD-based ROM's as a surrogate model for optimisation, and the inherent interpolation and extrapolation qualities in ROM's are exploited with locally accurate models. These focus areas are explained in the following sections.

## 1.1 Goals

The principal goal of this thesis is to develop a POD-based ROM for transient, non-linear groundwater models in order to significantly reduce the simulation time per function call. Each POD-based surrogate model is based on the dominant responses of the full model, where each model response is a function of the initial conditions, the model parameters and boundary conditions. The POD models used and developed in this research are predominantly intrusive, meaning that there are different degrees of interaction with the full simulation programming code. The intrusive property in POD models is often seen as a disadvantage because the full system matrices typically have to be set up or approximated at each new set of parameters or each new time step. However, the advantages of intrusive modelling are that the model is entirely physics based, and the parameter optimisation required to create a metamodel is avoided.

A second goal is to develop an optimisation algorithm that makes use of the interpolation qualities inherent to the physics based ROM, by using the developed ROM as a surrogate model. Furthermore, in the literature that exists on the Reduced Order Modelling with POD, allusion is often made to strong interpolation and extrapolation qualities that exist in these models (see De Vuyst (2009), Burkardt *et al.* (2006) and Audouze *et al.* (2009)). This interpolative quality is investigated, initially using the Trust Region POD (TRPOD) method developed by Fahl (2000). The TRPOD defines a region of confidence in the parameter space which acts as a constraint in the optimisation problem. To update the POD model using the TRPOD approach, high-fidelity simulations have to be executed. In order to eliminate the execution of these high-fidelity model executions during inverse modelling studies, a technique is proposed in this research that makes use Support Vector Classification (see Gunn (1998)) to

define a confidence region. The method is named SVC-POD, and does not require full FE simulations during optimisation studies. The two methods are compared on a numerical case study.

## 1.2 Context and contribution of work

A number of journal papers have been published on the subject of using reduced modelling in the field of groundwater. Some relevant papers include those by Siade *et al.* (2010), Vermeulen *et al.* (2004), McPhee and Yeh (2008) and Winton *et al.* (2011), where the application of POD to saturated flow problems is treated, and shown to yield speed-up times of  $\approx 1000$  with very low ROM errors. The research in this thesis builds on the research done to date by applying the ROM methodology to unsaturated flow problems.

The POD-based ROM's developed in this thesis are based on the Finite Element method, but they can easily be adapted to other discretisation approaches. A number of different commercial software exist for creating high-fidelity groundwater models, such as ModFlow (Arlen W. Harbaugh (2005)), Hydrus 2D/3D (Šimůnek *et al.* (2007)) and FEFLOW (Diersch (2014)). Modflow uses a Finite Difference approach, while Hydrus and FEFLOW use a Finite Element approach. The standard Finite Element software packages are not suitable for this research, due to the difficulties of accessing and modifying and editing the source code, and extracting the integral matrices. As an alternative, a software package called FreeFem++ by Hecht (2012) is used. The software uses the Finite Element method, and requires the governing equations in their weak formulation as input. FreeFem++ is chosen because it is open-source and because it allows easy manipulation and extraction of integral matrices and vectors.

Currently, the commercial groundwater modelling software packages Hydrus, Modflow and FEFLOW use gradient based approaches for inverse modelling and optimisation, such as the software Parameter ESTimation (PEST) in Modflow and FEFLOW (Doherty (2010)). Furthermore, in the technical manual for Hydrus by Jirka Šimůnek and Miroslav Šejna (2011), it is stated that the Marquardt-Levenberg approach for inverse modelling has become a standard in parameter calibration for soil scientists and hydrologists. In the inverse modelling studies in this thesis, gradient based techniques are also used to search for optima in the parameter space, using the commercial software DOT by Vanderplaats (2001). While a significant problem in groundwater inverse modelling problems is the existence of local optima (see Vrugt *et al.* (2008)), this problem is not addressed in depth in this thesis. However, the use of ROM's, and the associated reduction in computational complexity, lends itself to use in evolutionary and genetic optimisation algorithms, such as particle swarm optimisation, to address the problem of local optima.

As shown in previous research, such as Astrid (2004), this research shows that the application of Galerkin POD to non-linear problems does not result in feasible speed-up times. This is demonstrated on a case study, and two alternative

approaches are subsequently developed that yield attractive speed-up times. The developed approach is named the Linearised Galerkin POD (LGP) approach. Following the development of the LGP approach, the interpolative properties of the non-linear POD models is highlighted using the TRPOD and the SVC-POD approaches.

### 1.3 Document layout

The layout of the document is shown in Fig. 1.1, where the focal areas of each chapter are highlighted. In chapter 2 the groundwater flow equations are developed and discretised with the finite element method. Subsequently, the application of the Galerkin POD method is applied to the Richards' equation in chapter 3. In chapter 4 the interpolation/extrapolation qualities of POD models is explored on a site in the Table Mountain Group (TMG) in South Africa (see Jovanovic and Bagan (2011)), using a saturated flow model. An invented case study is used to show that the GPOD approach is not feasible for unsaturated flow problems.

A method, named the Linearised Galerkin POD (LGP) approach, is then developed and applied to the transient, non-linear Richards' equation in chapter 5. The method is applied to the same unsaturated problem analysed in chapter 4, showing the significant speed-up achieved with the LPG method. Finally, the TRPOD and SVC-POD methods are developed in chapter 6 and demonstrated in case studies in chapter 7, in order to highlight and exploit the strong interpolation and localised extrapolation properties inherent to the POD-based ROM's.

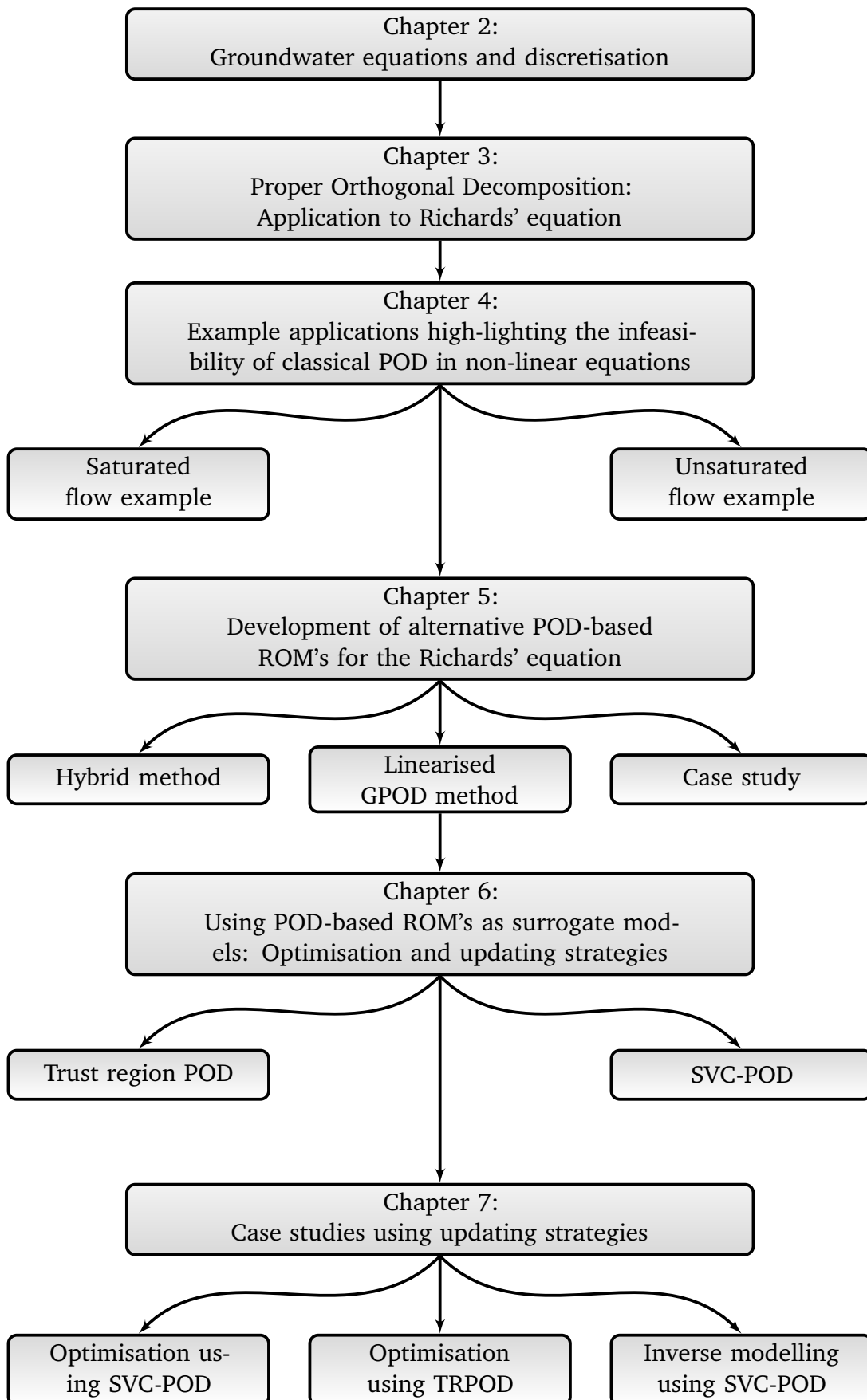


Figure 1.1: Document layout

## Chapter 2

# Finite Element discretisation of groundwater mass flow equations

*I*N this chapter the groundwater equations describing water movement through saturated and unsaturated media are derived and discretised using the Finite Element method. The application of Dirichlet boundary conditions, also known as first type or fixed boundary conditions, and Neumann boundary conditions, or second type boundary conditions, are developed in detail for system response.

As an introduction to the groundwater equations developed in this chapter, an overview of the interaction between groundwater and surface water is shown in Fig. 2.1, where a simplified drawing of the water cycle is shown. Water is seen to enter the system by precipitation from rainfall and irrigation and subsurface flow. Water then leaves the system by subsurface flow, extraction, evapotranspiration (*ET*) and run-off. The water that infiltrates the ground is called infiltration.

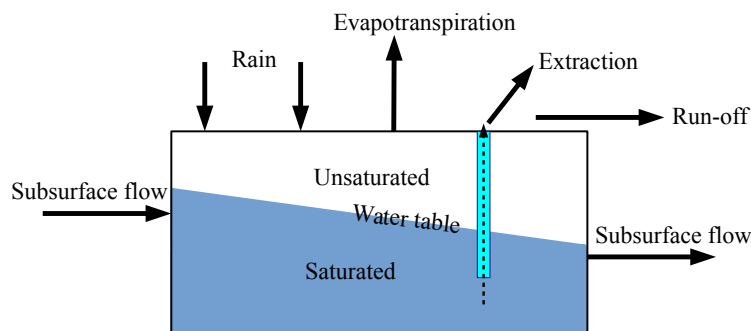


Figure 2.1: Simplified layout of water cycle, showing the saturated and unsaturated zones.

The movement of water in the saturated and unsaturated zones are subject to a number of parameters, such as conductivity, that need to be correctly identified in order to accurately approximate the reality. In the following sections, the mass

flow equations are derived and the parameters are identified and explained. These parameters become the subject of inverse modelling studies in later chapters.

## 2.1 Developing the mass flow equation

In this section the mass flow equation is derived for flow through a porous medium, and subsequently applied to saturated and unsaturated conditions. The derivations in this section are a summary of the flow equations derived in Diersch (2014) for variably saturated, time varying flow problems. A typical control volume of a porous medium with liquid movement through the boundaries is shown in Fig. 2.2. The porosity  $\varepsilon$  can be defined as the ratio of the interconnected volume of void space in a representative elementary volume (REV) to the total volume of the REV as follows:

$$\varepsilon = \frac{\text{Volume of interconnected void space}}{\text{Bulk volume}} \quad (2.1)$$

The porous medium is assigned a volume fraction  $\varepsilon_f$  for a fluid phase filling the void space  $\varepsilon$  as follows:

$$\varepsilon_f = \frac{\text{Volume of fluid}}{\text{Bulk volume}} \quad (2.2)$$

so that the fraction of fluid volume to total volume can be written as  $\varepsilon_f = s_f \varepsilon$ , where  $s_f$  is the percentage fluid saturation in the void space. The fluid phases present can be liquid or gas, and the volume fraction of the liquid, gas and solid phases are represented denoted respectively as:

$$\begin{aligned} \varepsilon_l &= \varepsilon s_l \\ \varepsilon_g &= \varepsilon s_g \\ \varepsilon_s &= 1 - \varepsilon \end{aligned} \quad (2.3)$$

where  $s_l$  is the percentage liquid saturation, and  $s_g$  is the percentage gas saturation.

### 2.1.1 The mass flow equation

In this research, only the conservation of mass of liquid in a given domain is of interest. The conservation of mass over the control volume  $\Omega$  of a liquid, given a boundary  $\partial\Omega$ , is calculated as follows:

$$\underbrace{\frac{\partial}{\partial t} \int_{\Omega} \varepsilon_l \rho_l d\Omega}_{\text{Change in mass}} + \underbrace{\int_{\partial\Omega} \varepsilon_l \rho_l (\mathbf{v}_l \cdot \mathbf{n}) \cdot d\mathbf{A}}_{\text{Boundary flow}} = 0 \quad (2.4)$$

where  $\rho_l$  is the liquid density,  $s_l$  is the saturation percentage of the liquid of the available void space  $\varepsilon$ ,  $\mathbf{v}_l$  is the velocity vector of the liquid and  $\mathbf{n}$  is normal to the

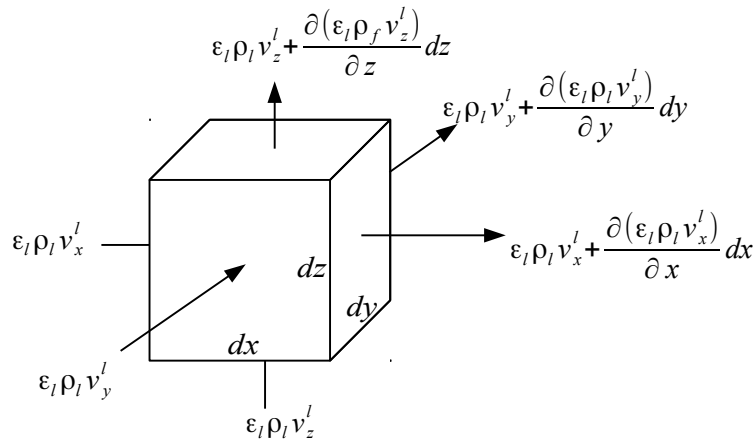


Figure 2.2: Control volume showing fluid flow through a porous medium

boundary directed away from the control volume. This equation is expanded by summing the terms entering and leaving the control volume (see Hiscock (2005)) as follows:

$$\frac{\partial \epsilon_l \rho_l}{\partial t} + \frac{\partial(\epsilon_l \rho_l v_x)}{\partial x} + \frac{\partial(\epsilon_l \rho_l v_y)}{\partial y} + \frac{\partial(\epsilon_l \rho_l v_z)}{\partial z} = 0 \quad (2.5)$$

A compact form of the equation describing a liquid conservation is written as:

$$\frac{\partial \epsilon_l \rho_l}{\partial t} + \nabla \cdot (\rho_l \epsilon_l \mathbf{v}_l) = 0 \quad (2.6)$$

The liquid flux vector is defined as the difference in solid and liquid velocity vectors as follows:

$$\mathbf{q}_l = \epsilon_l (\mathbf{v}_l - \mathbf{v}_s) \quad (2.7)$$

where  $\mathbf{v}_s$  is the solid phase velocity vector. The solid phase velocity is typically much smaller than the fluid velocity, but it is relevant when compressibility is considered. Substituting Eq. 2.7 in Eq. 2.6, and expanding the time derivative term yields:

$$\epsilon_{s_l} \frac{\partial \rho_l}{\partial t} + \epsilon_l \rho_l \frac{\partial s_l}{\partial t} + s_l \rho_l \frac{\partial \epsilon}{\partial t} + \nabla \cdot (\rho_l \mathbf{q}_l) + \nabla \cdot (\epsilon_l \rho_l \mathbf{v}_s) = 0 \quad (2.8)$$

### 2.1.2 Simplifications

Several further adaptations (from Diersch (2014)) are made to further simplify Eq. 2.8. Firstly, we assume that the fluid is slightly compressible, where compressibility refers to a relative volume change, and that the medium is slowly deformable, where deformation refers to a relative change in position of points or particles in a body. If we assume that  $\epsilon_l \rho_l \gg \nabla \cdot \mathbf{v}_s$ , we can express the last term in Eq. 2.8 as:

$$\nabla \cdot (\epsilon_l \rho_l \mathbf{v}_s) \approx \epsilon_l \rho_l (\nabla \cdot \mathbf{v}_s) \quad (2.9)$$

Secondly, a mass control for the solid phase is derived in the same way as Eq. 2.6 to obtain an expression for the  $\mathbf{v}_s$ :

$$\frac{\partial \varepsilon_s \rho_s}{\partial t} + \nabla \cdot (\rho_s \varepsilon_s \mathbf{v}_s) = 0 \quad (2.10)$$

Since  $\varepsilon_s = 1 - \varepsilon$ , the equation for solid mass conservation can be rewritten as:

$$\frac{\partial (1 - \varepsilon) \rho_s}{\partial t} + \nabla \cdot (\rho_s \mathbf{v}_s [1 - \varepsilon]) = 0 \quad (2.11)$$

where  $\rho_s$  is assumed constant (i.e. incompressible solid grains) so the term  $\nabla \cdot \mathbf{v}_s$  is found as by manipulating Eq. 2.11 as follows:

$$\nabla \cdot \mathbf{v}_s \approx \left( \frac{1}{1 - \varepsilon} \right) \frac{\partial \varepsilon}{\partial t} \quad (2.12)$$

which decouples the solid and fluid phases from each other. Thirdly, an expression is derived to describe  $\frac{\partial \varepsilon}{\partial t}$  as follows:

$$d\varepsilon = \frac{\partial \varepsilon}{\partial p_l} dp_l = \left( \frac{1}{1 - \varepsilon} \frac{\partial \varepsilon}{\partial p_l} \right) (1 - \varepsilon) dp_l = v(1 - \varepsilon) dp_l \quad (2.13)$$

where the skeleton compressibility is represented by  $v$ . This approximation disregards the effects from mass fraction and temperature. As a fourth simplification, the Oberbeck-Boussinesq (OB) approximation (Oberbeck (1879)) is applied to modify the mass conservation term for a liquid as follows:

$$\nabla \cdot (\rho_l \mathbf{q}_l) = \rho_l \nabla \cdot \mathbf{q}_l \quad (2.14)$$

These simplifications are now used to further adapt Eq. 2.8. The first simplification in Eq. 2.12 is substituted into Eq. 2.9:

$$\nabla \cdot (\varepsilon s_l \rho_l \mathbf{v}_s) \approx \varepsilon s_l \rho_l \left( \frac{1}{1 - \varepsilon} \frac{\partial \varepsilon}{\partial t} \right) \quad (2.15)$$

Substituting Eq. 2.15 into Eq. 2.8, a new expression for Eq. 2.8 is obtained:

$$\varepsilon s_l \frac{\partial \rho_l}{\partial t} + \varepsilon \rho_l \frac{\partial s_l}{\partial t} + s_l \rho_l \frac{\partial \varepsilon}{\partial t} + \nabla \cdot (\rho_l \mathbf{q}_l) + \varepsilon s_l \rho_l \left( \frac{1}{1 - \varepsilon} \frac{\partial \varepsilon}{\partial t} \right) = 0 \quad (2.16)$$

Adding the like terms yields:

$$\varepsilon s_l \frac{\partial \rho_l}{\partial t} + \varepsilon \rho_l \frac{\partial s_l}{\partial t} + \frac{s_l \rho_l}{1 - \varepsilon} \frac{\partial \varepsilon}{\partial t} + \nabla \cdot (\rho_l \mathbf{q}_l) = 0 \quad (2.17)$$

Now the approximation in Eq. 2.13 is applied to yield:

$$\varepsilon s_l \frac{\partial \rho_l}{\partial t} + \varepsilon \rho_l \frac{\partial s_l}{\partial t} + s_l \rho_l v \frac{\partial p_l}{\partial t} + \nabla \cdot (\rho_l \mathbf{q}_l) = 0 \quad (2.18)$$

The OB approximation (Eq. 2.14) is applied to Eq. 2.18, and we divide throughout by  $\rho_l$  to obtain:

$$\frac{\varepsilon s_l}{\rho_l} \frac{\partial \rho_l}{\partial t} + \varepsilon \frac{\partial s_l}{\partial t} + s_l v \frac{\partial p_l}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.19)$$



### 2.1.3 Mass flow in terms of pressure head

The mass flow equation can also be expressed in terms of the pressure head  $h$  with unit [L], instead of pressure. The pressure change with depth of liquid is defined as  $dp_l = \rho_l g dz$ , where  $z$  is the vertical axis, positive in an upward direction. In order to calculate pressure head, we substitute  $dz = dh = \frac{dp_l}{\rho_l g}$ . Pressure head is typically calculated at a point in the system as  $h = \frac{p_l}{\rho_l g}$ . Alternatively, the head in a system can be referenced from a global datum as follows:

$$h_e = \frac{p_l}{\rho_l g} + z = h + z \quad (2.20)$$

where  $h_e$  is the hydraulic head,  $g = \|\mathbf{g}\|$  is the gravitational acceleration, and  $\mathbf{g}$  is positive in the negative  $z$ -axis. The relationship between pressure head and hydraulic head are shown in Fig. 2.3. The mass conservation equation can be

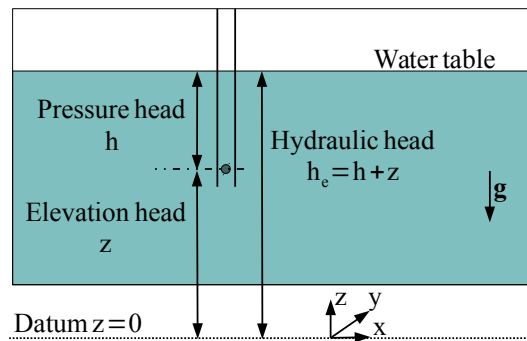


Figure 2.3: Hydraulic head in relation to pressure and elevation head (Diersch (2014))

expanded in terms of  $h$  as follows:

$$\frac{\varepsilon s_l}{\rho_l} \frac{\partial \rho_l}{\partial h_e} \frac{\partial h_e}{\partial t} + \varepsilon \frac{\partial s_l}{\partial t} + s_l v \frac{\partial p_l}{\partial h_e} \frac{\partial h_e}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.21)$$

The derivative  $\frac{\partial \rho_l}{\partial h_e} = \gamma_l \rho_o g$  is used from Diersch (2014), and the derivative  $\frac{\partial p_l}{\partial h_e} = \rho_o g$  is taken Eq. 2.20 and substituted into the mass flow equation as follows:

$$s_l \underbrace{\rho_o g (\varepsilon \gamma_l + v)}_{s_0} \frac{\partial h_e}{\partial t} + \varepsilon \frac{\partial s_l}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.22)$$

where  $S_0$  represents the specific storage coefficient. Finally, the general mass flow equation can be represented as:

$$s_l S_0 \frac{\partial h_e}{\partial t} + \varepsilon \frac{\partial s_l}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.23)$$

### 2.1.4 Adaptation for saturated flow

In a saturated porous medium, the term  $s_l = 1$ , and the mass flow equation is simplified to :

$$S_0 \frac{\partial h_e}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.24)$$

The mass flux vector describing fluid flow through a saturated zone is based on Darcy's law, published in 1856 by Darcy (1856). In his paper, Darcy showed that the hydraulic conductivity is dependent on the soil properties under consideration, so the Darcy velocity (see Kresic (2007)) can be calculated as follows:

$$\mathbf{q}_l = -\frac{\mathbf{k}}{\mu} \nabla(p + \rho g z) = -\frac{\mathbf{k} \rho g}{\mu} \nabla\left(\frac{p}{\rho g} + z\right) = -\frac{\mathbf{k} \rho g}{\mu} \nabla(h_e) \quad (2.25)$$

where  $\mathbf{k}$  is the hydraulic conductivity,  $\mu$  is the viscosity,  $p$  is the pressure,  $\rho$  is the density,  $g$  is gravitational acceleration and  $z$  is the vertical height. The hydraulic conductivity factor is combined into one factor as follows:

$$\mathbf{K}_s = \frac{\mathbf{k} \rho g}{\mu} \quad (2.26)$$

allowing us to rewrite the mass flow equation as:

$$S_0 \frac{\partial h_e}{\partial t} - \nabla \cdot (\mathbf{K}_s \nabla h_e) = 0 \quad (2.27)$$

Often, it is relevant to calculate the pressure head directly. Substituting Eq. 2.20 into Eq. 2.27, the pressure head can be calculated as:

$$S_0 \frac{\partial h}{\partial t} - \nabla \cdot (\mathbf{K}_s \nabla (h + z)) = 0 \quad (2.28)$$

The equation for saturated flow in expanded form becomes:

$$S_0 \frac{\partial h}{\partial t} = \frac{\partial}{\partial x} \left( K_x \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left( K_y \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left( K_z \frac{\partial (h + z)}{\partial z} \right) \quad (2.29)$$

### 2.1.5 Adaptation to unsaturated flow

In the unsaturated zone, the saturation  $s_l$  varies because there is both air and water in the intra-porous spaces. The first term in Eq. 2.23 is expanded as follows:

$$\frac{\partial s_l}{\partial t} = \frac{\partial s_l}{\partial h} \frac{\partial h}{\partial t} = C^* \frac{\partial h}{\partial t} \quad (2.30)$$

where  $C^* = C^*(h)$  is the moisture capacity described by an empirical relationship. The general mass equation in Eq. 2.23 is now changed to:

$$s_l S_0 \frac{\partial h}{\partial t} + \varepsilon C^* \frac{\partial h}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.31)$$

Adding the like terms reduces the equation to:

$$(s_l S_0 + \varepsilon C^*) \frac{\partial h}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \quad (2.32)$$

so that the pressure head can be calculated from one partial differential equation. The conductivity in the unsaturated also changes with a change in  $h$ , and so the flux vector in a variably saturated zone is described as:

$$\mathbf{q}_l = -\mathbf{K}(h) \nabla(h + z) \quad (2.33)$$

where  $\mathbf{K}(h)$  is the head dependent conductivity described by the term  $\mathbf{K}(h) = \mathbf{K}_s k_r(h)$ , and  $k_r(h)$  is a reduction term, reducing the conductivity with a decrease in head, according to an empirical relationship (see Table 2.1). The terms preceding the time derivative term are grouped as:

$$C = s_l S_0 + \varepsilon C^* \quad (2.34)$$

The complete equation describing mass conservation in an unsaturated medium, also known as the Richards' equation, can be obtained by substituting Eq. 2.33 into Eq. 2.32:

$$C(h) \frac{\partial h}{\partial t} = \nabla \cdot [\mathbf{K}(h) \nabla(h + z)] \quad (2.35)$$

This formulation is almost identical to the formulation for saturated flow equation in Eq. 2.28, except that the conductivity and storativity terms become dependent on the pressure head.

### 2.1.6 Non-linear terms

The conductivity and moisture capacity in the unsaturated zone change non-linearly with a change in pressure head, corresponding to the saturation in the medium. The percentage of saturation of a medium, or effective saturation  $s_e$ , can be defined as Todd and Mays (2005):

$$s_e = \frac{s_l - s_r}{s_s - s_r} = \frac{\varepsilon_l - \varepsilon_{l,r}}{\varepsilon_{l,s} - \varepsilon_{l,r}} \quad (2.36)$$

where  $s_s$  is the maximum saturation of liquid (usually =1),  $s_r$  is the residual saturation,  $\varepsilon_{l,s}$  is the saturated moisture content and  $\varepsilon_{l,r}$  is the residual moisture content. A number of analytic functions have been proposed to approximate  $s_e$ .

One of the functions that is widely used was proposed by van Genuchten (1978) as follows:

$$s_e = \begin{cases} \frac{1}{(1 - |h/h_b|^n)^m} & h < 0 \\ 1 & h \geq 0 \end{cases} \quad (2.37)$$

Making  $s_l$  the subject of the formula:

$$s_l = s_r + (s_s - s_r)(1 - |h/h_b|^n)^m \quad (2.38)$$

With this formulation, we can now find an expression for the moisture capacity:

$$C^* = \frac{\partial s_l}{\partial h} = \frac{mn|h/h_b|^{n-1}}{h_b(1 - |h/h_b|^n)^{m+1}}(s_s - s_r) \quad (2.39)$$

A saturation of  $s_l < 1$  indicates that there is both air and water in the intra-porous spaces. The presence of air means that the conductivity decreases, and is described as a fraction of the saturated conductivity,  $K_s$ . Three formulations for the relationship of water content to conductivity are also given in 2.1. In the table,  $n$ ,  $m$ ,  $A$  and  $B$  refer to empirical constants. The formulation given by van Genuchten is the most accurate, but also the most computationally expensive. An example (using parameters from the example in section 2.4) using the van Genuchten relations is shown in Fig. 2.4, where the change of moisture capacity and conductivity are plotted against a decrease in pressure head. This example highlights the non-linearity in the functions, and becomes an important consideration in later chapters.

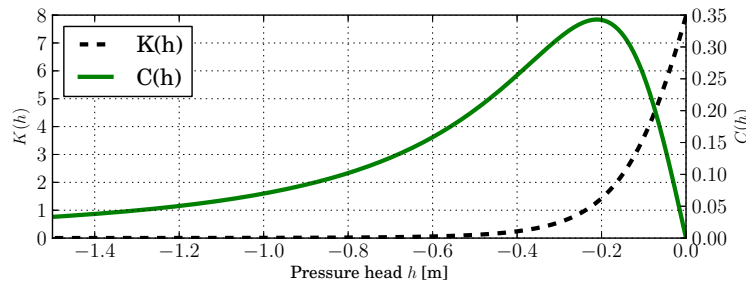


Figure 2.4: Example of the change in moisture capacity  $C(h)$  and conductivity  $K(h)$  with a change in head pressure

## 2.2 Applying the Finite Element Method

In this section the application of the Finite Element (FE) method is demonstrated on the Richards' equation, as it is derived in Eq. 2.35. The FE method is presented as a functional application, and does not enter into theoretical considerations.

Table 2.1: Different approximations of  $s_e - h$  (from Todd and Mays (2005) and Diersch (2014))

<b>van Genuchten relationship</b>	
<i>Effective saturation</i>	
$s_e = \begin{cases} \frac{1}{(1 -  h/h_b ^n)^m} & h < 0 \\ 1 & h \geq 0 \end{cases}$	$h_b =$ bubbling capillary pressure $m =$ curve fitting parameter $n = \frac{1}{1 - m}$ (pore size distribution index)
<i>Hydraulic conductivity reduction term</i>	
$k_r(h) = s_e^{1/2} (1 - [1 - s_e^{1/m}]^m)^2$	
<b>Brooks and Correy</b>	
<i>Effective saturation</i>	
$s_e = \begin{cases} \left( \frac{1}{ h/h_b ^n} \right) & h < 0 \\ 1 & h \geq 0 \end{cases}$	$h_b =$ bubbling capillary pressure $s_r =$ residual water content $s_s =$ saturation water content $n = \frac{1}{1 - m}$ (pore size distribution index)
<i>Hydraulic conductivity reduction term</i>	
$k_r(h) = \left( \frac{s - s_r}{s_s - s_r} \right)^n = s_e^n$	
<b>Haverkamp relationship</b>	
<i>Effective saturation</i>	
$s_e = \begin{cases} \frac{1}{1 + ( h/h_b Z ^m)} & h < 0 \\ 1 & h \geq 0 \end{cases}$	$h_b =$ bubbling capillary pressure $A, B =$ positive curve fitting indices $Z = 1m^{-1}$ (unit cancelling coefficient)
<i>Hydraulic conductivity reduction term</i>	
$k_r(h) = \frac{A}{(A -  zh ^B)}$	

The Galerkin approach to FE discretisation presented by Kwon and Bang (1996) is followed, and the treatment of Neumann and Dirichlet boundary conditions is discussed in detail. The discussion in this section serves as an introduction to the reduced order modelling methods used in the following sections, in which the Galerkin method is also used.

The Richards' equation is a non-linear equation, indicating that the parameters are dependent on the system response, or objective function. With the pressure head  $h$  in  $[L]$  as the objective function, the Richards' equation is dependent on two non-linear functions, namely the  $C(h)$  in  $[L^{-1}]$  and  $K(h)$  in  $[L/T]$ , introduced in section 2.1.6 as the moisture content and conductivity respectively. Assuming the medium under consideration is homogeneous, the Richards' equation from Eq. 2.35 is written:

$$C(h) \frac{\partial h}{\partial t} = \nabla \cdot [K(h) \nabla (h + z)] \quad (2.40)$$

which is subject to implicit initial and boundary conditions:

$$\begin{aligned}
 h(x, y, z, 0) &= f_0(x, y, z) && \text{(Initial conditions)} \\
 h(x, y, z, t) &= f_d(x, y, z, t), \quad (x, y, z, t) \in \Gamma_d && \text{(Dirichlet boundary)} \\
 q_n(x, y, z, t) &= f_n(x, y, z, t), \quad (x, y, z, t) \in \Gamma_n && \text{(Neumann boundary)}
 \end{aligned} \tag{2.41}$$

where  $h$  varies over  $x$ ,  $y$ ,  $z$  and  $t$ . The Dirichlet and Neumann boundaries are represented on and by functions  $f_d$  [L] and  $f_n$  [L/T] respectively. The recharge/discharge  $q_n$  is in [L/T]. A simple depiction of boundary conditions is given in Fig. 2.5, where  $\Omega$  represents the domain, and  $\Gamma$  represents the domain boundary. In the following sections the discretisation of the domain is discussed.

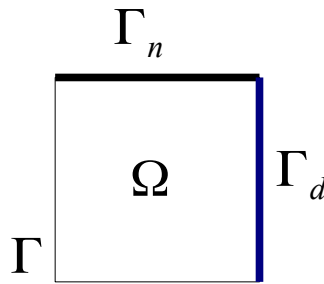


Figure 2.5: Example of a model domain ( $\Omega$ ) and boundary conditions

### 2.2.1 Weak formulation

Let us define the domain under consideration as  $\Omega$ , which is defined in  $\mathbb{R}^D$ , where  $D \in \{1, 2, 3\}$ , and is typically bounded by a Lipschitz boundary consisting of a combination of Dirichlet and Neumann boundary conditions, as defined in Eq. 2.41. In order to obtain a Finite Element solution, we first derive the weak formulation. The PDE is first multiplied by a test function  $w$  that satisfies the null condition on the Dirichlet boundary  $\Gamma_d$ . The PDE is subsequently integrated over  $\Omega$  as follows:

$$\int_{\Omega} wC(h) \frac{\partial h}{\partial t} = \int_{\Omega} w \nabla \cdot [K(h) \nabla (h + z)] \tag{2.42}$$

Applying integration by parts to Eq. 2.42 yields the Neumann boundary term as follows:

$$\int_{\Omega} wC(h) \frac{\partial h}{\partial t} = \int_{\Gamma_n} wK(h) \nabla (h + z) \cdot \mathbf{n} - \int_{\Omega} K(h) \nabla w \cdot \nabla (h + z) \tag{2.43}$$

where  $\int_{\Gamma_n} wK(h)\nabla(h+z) \cdot \mathbf{n}$  is the Neumann boundary term and  $\mathbf{n}$  is the vector normal to the surface. The final term in Eq. 2.43 can be developed as:

$$\int_{\Omega} K(h)\nabla w \cdot \nabla(h+z) = \int_{\Omega} K(h)\nabla w \cdot \nabla h + \int_{\Omega} \frac{\partial w}{\partial z} K(h) \quad (2.44)$$

Substituting Eq. 2.44 back into Eq. 2.43, we get the final weak formulation of the Richards' equation:

$$\int_{\Omega} C(h)w \frac{\partial h}{\partial t} + \int_{\Omega} K(h)\nabla w \cdot \nabla h = \int_{\Gamma_n} wK(h)\nabla(h+z) \cdot \mathbf{n} - \int_{\Omega} K(h) \frac{\partial w}{\partial z} \quad (2.45)$$

With this weak formulation we can apply the FE method to discretise the terms into piecewise components.

### 2.2.2 Discretisation of unsaturated flow equation

The spatial domain is discretised into triangles on which piecewise linear functions are considered, so that the response is defined as the sum of these functions as follows:

$$h = \sum_{j=1}^n h_j(t)v_j(x, y, z) \quad (2.46)$$

where  $n$  is the number of triangle vertices. The response  $h$  is now described as the sum of scalar values  $h_i$  multiplied by the piecewise continuous functions  $v_i$ . The piecewise functions will be referred to as ordinary basis functions. Furthermore,  $x$ ,  $y$  and  $z$  represent the different axis of the domain. Following the Galerkin approach, each test function is defined as:

$$w = v_i(x, y, z), \quad i = 1, 2, \dots, n \quad (2.47)$$

For brevity, we reduce  $v_i(x, y, z)$ ,  $h_i(t)$ ,  $C(h)$  and  $K(h)$  to  $v_i$ ,  $h_i$ ,  $C$  and  $K$  respectively. Substituting Eq. 2.46 and 2.47 into the weak formulation in Eq. 2.43, we calculate the response at each node:

$$\forall i, \sum_{j=1}^n \left( \int_{\Omega} C v_i \frac{\partial (v_j h_j)}{\partial t} + \int_{\Omega} K \nabla v_i \cdot \nabla (h_j v_j) \right) = - \int_{\Omega} K \frac{\partial v_i}{\partial z} + \int_{\Gamma_n} v_i f_n(t) \cdot \mathbf{n} \quad (2.48)$$

Now, since the base functions are not a function of time, the time-derivative changes as follows:

$$\forall i, \sum_{j=1}^n \left( \int_{\Omega} C v_i v_j \frac{\partial h_j}{\partial t} + \int_{\Omega} K \nabla v_i \cdot \nabla (h_j v_j) \right) = - \int_{\Omega} K \frac{\partial v_i}{\partial z} + \int_{\Gamma_n} v_i f_n(t) \cdot \mathbf{n} \quad (2.49)$$

This formulation is presented more compactly in the following section.

### 2.2.3 Matrix representation

The terms in Eq. 2.49 can be represented as a system of Ordinary Differential Equations (ODE's) in matrix form as follows:

$$\mathbf{M}^s(h) \frac{d\mathbf{h}}{dt} + \mathbf{M}^p(h)\mathbf{h} = -\mathbf{V}^p(h) + \mathbf{V}^n \quad (2.50)$$

where  $\mathbf{M}^s$  represents the mass matrix as a function of  $C(h)$  and  $\mathbf{M}^p$  is the stiffness matrices as a function of  $K(h)$ . The subject of the formula is the vector  $\mathbf{h}$ . The expanded form of the matrices is given as:

$$\mathbf{M}^s(h) = \begin{bmatrix} \int_{\Omega} C v_1 v_1 & \cdots & \int_{\Omega} C v_1 v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} C v_n v_1 & \cdots & \int_{\Omega} C v_n v_n \end{bmatrix} \quad (2.51)$$

$$\mathbf{M}^p(h) = \begin{bmatrix} \int_{\Omega} K \nabla v_1 \cdot \nabla v_1 & \cdots & \int_{\Omega} K \nabla v_1 \cdot \nabla v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} K \nabla v_n \cdot \nabla v_1 & \cdots & \int_{\Omega} K \nabla v_n \cdot \nabla v_n \end{bmatrix} \quad (2.52)$$

and the right hand side vectors in Eq. 2.50 of size  $[n \times 1]$  are expanded as follows:

$$\mathbf{V}^p(h) = \begin{bmatrix} \int_{\Omega} K \frac{\partial v_1}{\partial z} \\ \vdots \\ \int_{\Omega} K \frac{\partial v_n}{\partial z} \end{bmatrix}, \quad \mathbf{V}^n = \begin{bmatrix} \int_{\Gamma_n} f_n v_1 \cdot \mathbf{n} \\ \vdots \\ \int_{\Gamma_n} f_n v_n \cdot \mathbf{n} \end{bmatrix} \quad (2.53)$$

Combining the vector terms:

$$\mathbf{V}(h) = -\mathbf{V}^p(h) + \mathbf{V}^n \quad (2.54)$$

we rewrite Eq. 2.50 as:

$$\mathbf{M}^s(h) \frac{d\mathbf{h}}{dt} + \mathbf{M}^p(h)\mathbf{h} = \mathbf{V}(h) \quad (2.55)$$

Using the matrix format, different time discretisation techniques are considered in the following section.

### 2.2.4 Treatment of the time term

A number of different approaches can be used for the time derivative term. Here the  $\theta$ -method (see Segal (2012)) is applied to Eq. 2.55 as follows:

$$\mathbf{M}^s(h^{\tau+1}) \frac{\mathbf{h}^{\tau+1} - \mathbf{h}^{\tau}}{\delta t} + \theta \mathbf{M}^p(h^{\tau+1})\mathbf{h}^{\tau+1} + (1-\theta)\mathbf{M}^p(h^{\tau})\mathbf{h}^{\tau} = \theta \mathbf{V}(h^{\tau+1}) + (1-\theta)\mathbf{V}(h^{\tau}) \quad (2.56)$$



where  $\tau$  is the current time step, and  $\tau + 1$  is the next time step. Values used for  $\theta$  are typically either 0 (Explicit Euler), 1 (Implicit Euler) or 1/2 (Crank Nicolson). In this thesis only the Implicit Euler technique ( $\theta = 1$ ) is used due to its favourable convergence characteristics, resulting in the final formulation of the Richards' equation as:

$$[\mathbf{M}^s(h^{\tau+1}) - \delta t \mathbf{M}^p(h^{\tau+1})] \mathbf{h}^{\tau+1} = \mathbf{M}^s(h^{\tau+1}) \mathbf{h}^{\tau} + \delta t \mathbf{V}(h^{\tau+1}) \quad (2.57)$$

The Implicit Euler belongs to the class of ultra-stable methods, and the error is of  $O(\delta t)$ .

The response ( $h^{\tau+1}$ ) of the Richards' equation at time step  $\tau + 1$  is dependent on the non-linear terms  $C(h^{\tau+1})$  and  $K(h^{\tau+1})$ . This means that Eq. 2.57 needs to be calculated iteratively until  $h^{\tau+1}$  converges to an acceptable tolerance. One approach to achieve convergence is the Fixed Point Form (FPF) or Picard iteration method (see Diersch (2014)). This method is explained in Fig. 2.6. In the figure, a temporary function  $h^{temp}$  is created, and <sup>1</sup> the pressure head is updated until  $\mathbf{h}^{\tau+1}$  and  $\mathbf{h}^{temp}$  converge to some defined tolerance. Alternatively, the Newton iteration method (see Diersch (2014)) can be applied. For convenience, only the Picard iteration scheme is used in this research.

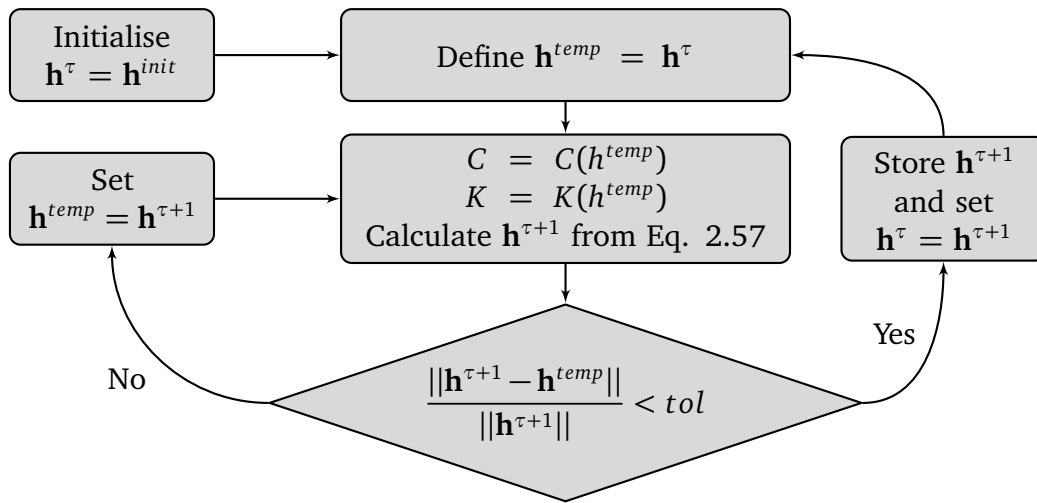


Figure 2.6: Fixed point form method for iterative convergence

## 2.3 Applying Dirichlet boundary conditions

Two different approaches of applying a Dirichlet boundary condition are explained in this section. The methods are demonstrated on a simple 1D implementation of the Richards' equation. Let us consider the 1D implementation of the Richards equation on an equi-spaced 1D domain, as shown in Fig. 2.7. This domain there

<sup>1</sup>The vector  $\mathbf{h}^{temp}$  is related to the function  $h^{temp}$  as defined in Eq. 2.46.

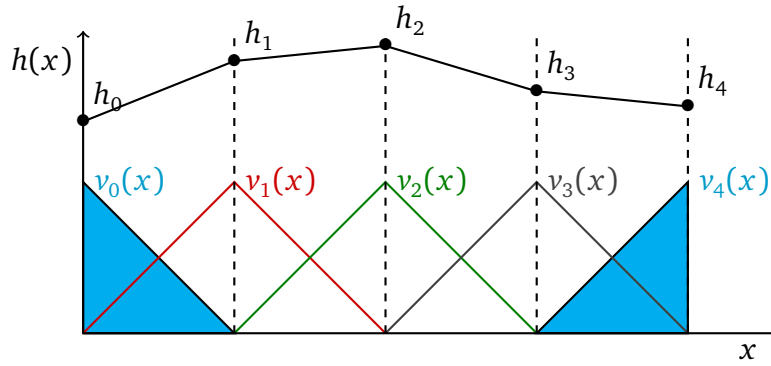


Figure 2.7: One dimensional grid showing the shape functions and boundary values

is discretised with 3 internal nodes and 2 boundary nodes. The Richards' equation is of interest normally in the vertical direction, but for demonstration purposes the equation is developed in the horizontal or  $x$ -axis. The shape functions used are piecewise linear functions, and are called P1 functions. Throughout this thesis P1 element are used, but the methods developed can equally be applied to P2 or other elements.

The solution to the 1D Richards' problem in Eq. 2.57 without Dirichlet boundary values is written in expanded form as follows:

$$\begin{bmatrix} \alpha_{00} & \beta_{01} & & & 0 \\ \beta_{10} & \alpha_{11} & \beta_{12} & & \\ & \beta_{21} & \alpha_{22} & \beta_{23} & \\ & & \beta_{32} & \alpha_{33} & \beta_{34} \\ 0 & & & \beta_{43} & \alpha_{44} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} \quad (2.58)$$

Here,  $\alpha$  and  $\beta$  are calculated as follows:

$$\alpha_{ii} = \int_{\Omega} C(h^{\tau+1})v_i v_i + \delta t \int_{\Omega} K(h^{\tau+1}) \frac{\partial^2 v_i}{\partial x^2}$$

$$\beta_{ij} = \int_{\Omega} C(h^{\tau+1})v_i v_j + \delta t \int_{\Omega} K(h^{\tau+1}) \frac{\partial v_i}{\partial x} \frac{\partial v_j}{\partial x}$$

where  $v_i$  and  $v_j$  are shape functions represented in Fig. 2.7, and  $i$  and  $j$  represent the row and column index respectively. The vector terms can be written in expanded form, considering the time discretisation as:

$$\phi_i = \int_{\Omega} C(h^{\tau+1})v_i h^{\tau} - \int_{\Omega} K(h^{\tau+1}) \frac{\partial v_i}{\partial z} + \int_{\Gamma_n} f_n v_i \cdot \mathbf{n} \quad (2.59)$$

When Dirichlet boundary conditions are present, the system of ODE's typically needs to be adapted further. In the following section the enforcement of a Dirichlet boundary condition is explained.

### 2.3.1 Enforcing the Dirichlet boundary condition

One way to enforce a value at the Dirichlet boundary is to restructure the matrix so that boundary nodes do not have an influence on nodes with Dirichlet boundary conditions. This is done by setting each row that has been assigned a Dirichlet boundary value to 0, and setting the diagonal value to one:

$$\underbrace{\begin{bmatrix} 1 & 0 & & & 0 \\ \beta_{10} & \alpha_{11} & \beta_{12} & & \\ & \beta_{21} & \alpha_{22} & \beta_{23} & \\ & & \beta_{32} & \alpha_{33} & \beta_{34} \\ 0 & & & 0 & 1 \end{bmatrix}}_{\mathbf{M}^{mod}} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} b_0^h \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ b_4^h \end{bmatrix} \quad (2.60)$$

The matrix is denoted as  $\mathbf{M}^{mod}$ , indicating that it has been modified to take the Dirichlet conditions into account. This method can be further adapted to explicitly separate the Dirichlet boundary nodes by formulating the problem slightly differently. The response can be represented explicitly as the sum of interior and boundary node values:

$$h(x, t) = \sum_{i=0}^n [h_i(t) + b_i^h(t)] v_i(x) \quad (2.61)$$

where  $n$  is the number of nodes in the design space and each  $b_i^h$  refers to the Dirichlet boundary value at node  $i$ . In this formulation, if a boundary value is imposed at a node  $b_i^h$ , the corresponding  $h_i$  is set to 0. The formulation in Eq. 2.57 is reduced to:

$$\underbrace{\begin{bmatrix} \beta_{10} & \alpha_{11} & \beta_{12} & & & \\ & \beta_{21} & \alpha_{22} & \beta_{23} & & \\ & & \beta_{32} & \alpha_{33} & \beta_{34} & \\ & & & & & \end{bmatrix}}_{\mathbf{M}^*} \begin{bmatrix} \begin{bmatrix} 0 \\ h_1 \\ h_2 \\ h_3 \\ 0 \end{bmatrix} \\ + \\ \begin{bmatrix} b_0^h \\ 0 \\ 0 \\ 0 \\ b_4^h \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} \quad (2.62)$$

where  $\mathbf{h}^*$  is the pressure head response with 0 on all the Dirichlet boundary nodes. This formulation can be reformulated to include the Dirichlet information on the RHS. With the matrix denoted as  $\mathbf{M}^*$ , the Dirichlet information can then be found as follows:

$$\mathbf{M}^* \mathbf{b}^h = \begin{bmatrix} \beta_{10} b_0^h \\ 0 \\ \beta_{34} b_4^h \end{bmatrix} \quad (2.63)$$

Applying this term to Eq. 2.62 allows us to reformulate the problem as follows:

$$\begin{bmatrix} \alpha_{11} & \beta_{12} & 0 \\ \beta_{21} & \alpha_{22} & \beta_{23} \\ 0 & \beta_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} + \begin{bmatrix} \beta_{10} b_0^h \\ 0 \\ \beta_{34} b_4^h \end{bmatrix} \quad (2.64)$$

The matrix is now a square matrix of dimension  $[n - n_{bn}]$ , where  $n_{bn}$  is the number of boundary nodes. If we want to keep the matrix in a dimension  $[n \times n]$ , where  $n$  is the total number of nodes, we can rewrite this system as:

$$\begin{bmatrix} 1 & 0 & & & 0 \\ 0 & \alpha_{11} & \beta_{12} & & \\ & \beta_{21} & \alpha_{22} & \beta_{23} & \\ & & \beta_{32} & \alpha_{33} & 0 \\ 0 & & & 0 & 1 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_5 \end{bmatrix} = \begin{bmatrix} b_0^h \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ b_4^h \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \beta_{10} b_0^h \\ 0 \\ \beta_{34} b_4^h \\ 0 \end{bmatrix}}_{\mathbf{v}^d} \quad (2.65)$$

The Dirichlet vector  $\mathbf{V}^d \in \mathbb{R}^n$  can be found from the matrix  $\mathbf{M}^{mod}$  in Eq. 2.60 as follows:

$$\mathbf{V}^d = \mathbf{M}^{mod} \mathbf{b}^h - \mathbf{b}^h \quad (2.66)$$

The formulation used in Eq. 2.65 is of particular importance in this research, and is used in the creation of the Reduced Order Models in the following chapters to treat the Dirichlet boundary conditions.

### 2.3.2 Alternate approach to treating the Dirichlet boundary

An alternative approach to impose Dirichlet boundary conditions (also the approach used by the software FreeFEM++ by Hecht (2012)) is to impose a penalty function (Barrett and Elliott (1986)). With a penalty function, we force the boundary values to assume the desired values. Assuming we have  $b_0^h$  and  $b_4^h$  as boundary nodes at  $h_0$  and  $h_4$  respectively. In the matrix described in Eq. 2.58, the diagonals corresponding to Dirichlet boundaries are set to a very high values ( $tg\nu = 10^{30}$  in FreeFem++). Correspondingly, we replace  $\phi_0$  and  $\phi_4$  with the desired boundary values multiplied by the value  $tg\nu$ . This forces the values of  $h_0$  and  $h_4$  to the desired boundary values of  $b_0^h$  and  $b_4^h$  respectively:

$$\begin{bmatrix} 10^{30} & \beta_{01} & & & 0 \\ \beta_{10} & \alpha_{11} & \beta_{12} & & \\ & \beta_{21} & \alpha_{22} & \beta_{23} & \\ & & \beta_{32} & \alpha_{33} & \beta_{34} \\ 0 & & & \beta_{43} & 10^{30} \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} 10^{30} b_0^h \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ 10^{30} b_4^h \end{bmatrix} \quad (2.67)$$

This approach forces the boundary values to the required boundary value, since the diagonal value now makes the other terms insignificant.

## 2.4 Unsaturated flow example

In order to demonstrate the vertical infiltration of water into an unsaturated medium, a parameterised 1D example is presented here. This example replicates

the benchmark example created by Celia and Bouloutas (1990), where the results are obtained by implementing the Finite Element approach presented in this chapter in the software FreeFem++. The example considers infiltration into a 1 m deep porous medium, with an initial pressure head of  $h_0 = -10$  m and a boundary head of  $h = -0.75$  m on the surface and  $h = -10$  m at the lower boundary. The model makes use of the van Genuchten approximation (Table 2.1) to estimate the moisture content and conductivity with change in pressure head. The soil parameters are defined as  $n = 2$ ,  $h_b = 1/3.35$  m,  $\varepsilon = 0.368$ ,  $s_r = 0.277$ ,  $s_s = 1$ ,  $S_s = 0$  and a saturated conductivity value of  $K_s = 0.922 \cdot 10^{-4}$  m/s.

The pressure head response is plotted in Fig. 2.8, and can be observed to penetrate the medium over the period of 1 day. The initial condition and subsequent infiltration is plotted with the corresponding changes in conductivity  $K(h)$  and moisture capacity  $C(h)$ . The non-linearity in these two functions is evident, and becomes the focus of later chapters. The development of the response is also shown at times 0.25 d, 0.5 d, 0.75 d and 1 d respectively.

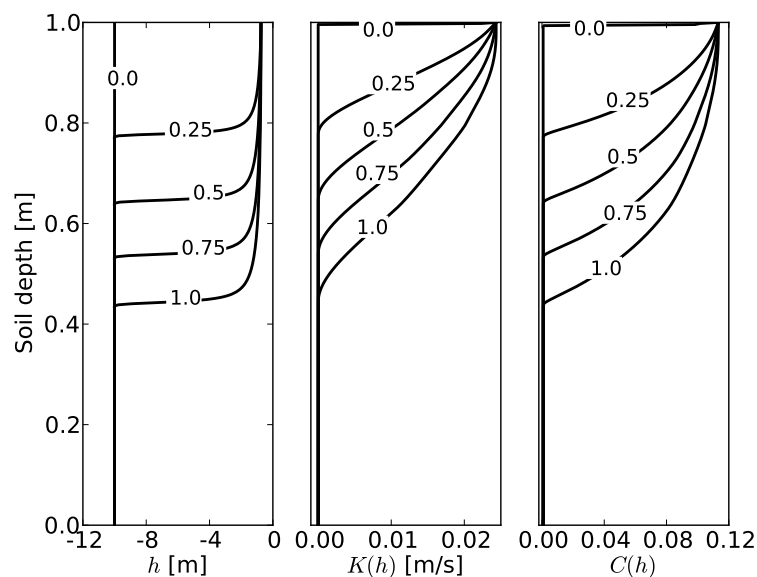


Figure 2.8: Head pressure development over time after infiltration, showing a replication of the benchmark simulation done by Celia and Bouloutas (1990). The time in days is shown on the plots, showing infiltration and the corresponding conductivity and moisture content values at  $t=0, 0.25, 0.5, 0.75$  and 1 day

The water is seen to infiltrate downward, moved by both pressure and gravity, while the Dirichlet boundary conditions keep the upper and lower boundaries constant. The conductivity and moisture content functions respond according to the van Genuchten relations from Table 2.1. The  $C(h)$  response has a sharper front at the infiltration front, while the conductivity gradient is more gentle. These large changes in the non-linear functions can be attributed to the fact that

they only have significant changes in value in the range of  $h = 0$  m to  $h = -1$  m, as shown earlier in Fig 2.4.

A segment of the code written for this example is shown in Appendix B.1, where the movement of water in 2D in response to boundary conditions is calculated using FreeFem++. The time to convergence is shown in Fig. 2.9, where the error defined in the Picard iteration scheme is shown to decrease logarithmically with an increase in iterations. In this thesis, an accuracy of  $10^{-3}$  is considered acceptable. This accuracy is achieved after 6 iterations in this example.

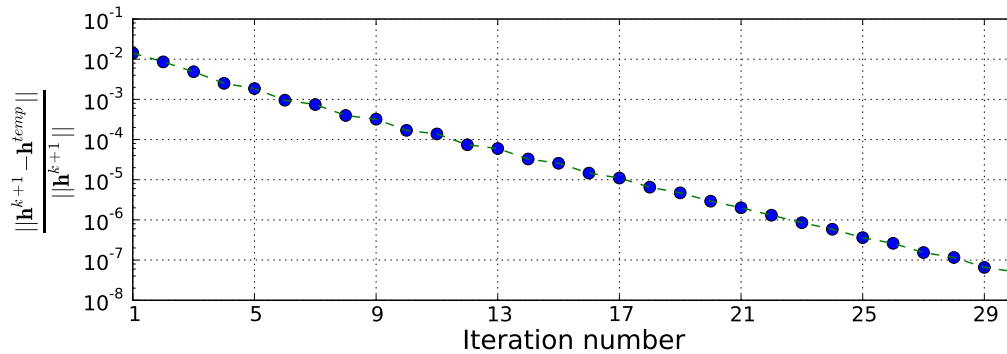


Figure 2.9: The convergence of a typical time step in the simulation using the Picard convergence scheme

## 2.5 Conclusion

In this chapter the application of the Finite Element method is demonstrated on the non-linear Richards' equation. Different ways of implementing a Dirichlet boundary condition are shown, which are a necessary introduction to the application of reduced modelling in the following chapter. The treatment of time discretisation was briefly discussed, and a benchmark example of infiltration into unsaturated soil is given, where the corresponding response of the non-linear functions can be analysed. The discretisation and boundary value approaches presented in this chapter serves as an important basis for the implementation of the reduced order models presented in the following chapter that are based on the Galerkin method.

## Chapter 3

# Proper Orthogonal Decomposition

**T**HIS chapter focuses on the application of the Proper Orthogonal Decomposition (POD) as a reduced modelling technique on the Richards' equation, a non-linear time-varying partial differential equation. The use of POD can be seen as a form of specialised form of surrogate modelling which allows us to accurately approximate the pressure or velocity responses over the model domain. In the POD based surrogate models, POD modes (alternatively called eigenvectors or shape functions) are used as global shape functions that replace local shape functions, in order to derive the surrogate model.

Throughout the thesis, a reference to a surrogate model indicates a mathematical model that substitutes a full finite element model. The word 'metamodel' is used to describe a surrogate model that uses an explicit interpolation technique, such as kriging, to evaluate a function response away from the original training points. The second surrogate model that is discussed is a Reduced Order Model (ROM) where no explicit interpolation is used, but the ROM can typically replicate the system response away from the original training points. The word 'reduction' refers to the reduction in shape functions used to describe the system response, and a POD-based ROM indicates that the POD modes are the shape functions used to describe the system response. Furthermore, instead of local values assigned to each node, weight values are assigned to each global shape functions. These weight values typically become the unknown of the problem.

The application of POD is seen in a wide variety of research fields, principally in fluid flow problems based on the Navier-Stokes equations such as the papers by Sirovich (1987), Couplet (2003), Astrid (2004), Rowley *et al.* (2004), Burkardt *et al.* (2006), Liberge and Hamdouni (2010) and Xiao *et al.* (2013), as well in many other applications such as heat transfer by Pinnau (2008) and Ostrowski *et al.* (2005), mechanical structures by Liang *et al.* (2002) and bio-medical simulations by Boulakia *et al.* (2011). In this research, the focus of using a ROM is principally to speed up time-consuming numerical groundwater simulations, especially in an inverse-modelling context.

Two general approaches are used when creating ROM's using POD, namely non-intrusive and intrusive. The former approach uses some interpolative tech-

nique (like kriging) to approximate weight functions, and the latter reconstructs the numerical model based on the original PDE without any interpolative techniques. The intrusive approaches typically requires manipulation of the code that is used when executing high-fidelity simulations, while the non-intrusive methods avoid modifying the source code.

Typically, POD-based ROM's have inherent interpolative qualities, indicating that the ROM can approximate results at parameter points which are not used to create the ROM. The development of non-intrusive approaches are largely neglected in this research, in order to avoid the complexity of calibrating the interpolative models such as kriging. These interpolative approaches are also called data-driven models, since a response is found by interpolating between a number of training data points. However, the focus in this research is on reconstructing the original PDE with the reduced basis set. One motivation for this is that extensive work has already been done on non-intrusive models. Another motivation is that the interpolation qualities inherent to purely intrusive ROM's can be explored. These interpolative qualities are explored and exploited in later chapters.

In this chapter, the Galerkin POD approach is applied to the Richards' equation, showing how the Galerkin method is adapted to create a ROM of the original PDE using global shape functions. A treatment of the Neumann and Dirichlet boundaries is included. This chapter subsequently serves as a tutorial showing how to create a ROM from the weak formulation.

## 3.1 Background and literature review

A short review on some relevant applications of POD is given in this section, followed by a comprehensive development of POD when applied to the Richards' equation, and finally a benchmark example to test and verify the approach. The context and contribution made by this thesis is outlined in section 1.2.

### 3.1.1 POD in groundwater problems

This research on groundwater modelling builds on the work already done by Siade *et al.* (2010), Vermeulen *et al.* (2004), McPhee and Yeh (2008) and Winton *et al.* (2011). In these works, reduced models using POD are applied to saturated groundwater models. Excellent results were obtained for this linear problem, with speed-up times up to 1000 reported by Siade *et al.* (2010). An example application of POD on a saturated flow problem is given in section 4.1, where similar speed-up times are obtained.

Following on the application of POD in saturated flow problems, classified as linear problems, this research focuses on the application of POD to the non-linear Richards' equation, where non-linearity means that the parameters are dependent on the objective function. A significant limitation in non-linear PDE's, as reported by Astrid (2004), is that the direct application of the Galerkin POD method does



not result in significant speed-up times. This is due to the necessity of creating full matrices at each time step, and subsequently reducing them, resulting in no speed-up in the final ROM execution times. A 1D example given in section 3.3, and a 2D example in section 4.2 both illustrate that the standard application of the Galerkin POD does not yield noteworthy speed-up times when executing the ROM's. In section 5.2, an alternative method to achieve feasible speed-up times, while using POD as a surrogate model, is developed to overcome the limitations in standard POD and interpolative ROM's.

### 3.1.2 POD in inverse modelling studies

Inverse modelling using POD-based models have been applied in a number of fields. Some of these approaches can be applied directly to inverse modelling in groundwater problems. A heat transfer problem is solved in the paper by Ostrowski *et al.* (2005), where a parameterised steady state model is approximated using POD. In the study, kriging was used to interpolate the weight values, but exact results were not obtained. Another study using kriging to interpolate the weight values is done by Gunes and Cadirci (2009), where a steady state Navier Stokes problem is solved using different Reynolds numbers. The use of kriging indicates that the ROM is non-intrusive. Again using a combination of kriging and POD, the shape optimisation of an intake port is done by Xiao *et al.* (2009), with a speed-up time of 26 achieved in this paper, using parallel processors. A speed-up of 26 can be considered as an acceptable speed-up ratio when considering the added complexity of creating a ROM for optimisation purposes.

Research into the use of an intrusive POD approach to reduce the simulation time of myocardial infarctions was done by Boulakia *et al.* (2011). The goal was to replicate the restitution curve using POD, which is done with limited success, due to the sharp non-linearity in the infarcted zones.

A POD based model was used to approximate the variation in the hydraulic conductivity in a saturated groundwater problem by Winton *et al.* (2011). It is shown that using a POD model, the function calls required to reach an optimum during an inverse modelling study are significantly fewer than using a full model due to the ROM having lower degrees of freedom.

Of particular interest in this thesis is the creation of ROM's using POD in space-time parameterised problems. Time can be treated as a parameter, and snapshots chosen according to some selection scheme, where each snapshot is a response of the system to a given set of parameters. One snapshot selection scheme is proposed by Siade *et al.* (2010), where the variation of the response is observed to decrease with time, for example a recovery curve after drawdown. Accordingly, a greater number of snapshots are chosen where the response has a steeper gradient. Similarly, an efficient Design Of Experiments (DOE) for ROM's is proposed in a parameterised space by Haasdonk and Ohlberger (2006), where low diffusivity values are selected with a higher frequency.

Two important approaches to dealing with a parameterised space is the

method of In-Situ Adaptive Tabulation, initially proposed by Pope (1997) (see also Haasdonk (2011) and De Vuyst (2009)), and the Greedy algorithm by Audouze *et al.* (2009). The In-Situ method effectively approximates a response based on a number of stored matrices. The greedy algorithm upgrades the ROM across the parameter space by iteratively evaluating the ROM error across the parameter space, and updating the ROM from high-fidelity simulations at the points with high errors, until some accuracy tolerance is reached.

## 3.2 Model reduction with Proper Orthogonal Decomposition

In this section the creation of a Reduced Order Model, based on the Richards' equation is described, using Proper Orthogonal Decomposition (alternatively called Principal Component Analysis (PCA) or the Karhunen Loève Decomposition). In this application, the ROM is created using a reduced basis set, which is obtained by performing a PCA. The goal of a PCA is to obtain an optimal reduced basis set based on some maximum energy approach, where energy refers to the contribution of each basis function. The reduced basis set is interchangeably called global shape functions, modes, eigenvectors or maximum energy modes.

### 3.2.1 Choosing representative snapshots

In order to create a suitable reduced basis set (RBS), a set of snapshots is collected, typically taken across the parameter space. As stated earlier, each snapshot is a response of the system to a certain set of parameters. Each snapshot must be representative of the response of the system. Since the Richards' equation typically describes a transient response, time is considered as an extra parameter in the parameter space from which to extract snapshots. In this research, snapshots are typically a set of responses taken from a full finite element simulation, but can also be derived from other discretisation schemes such as Finite Difference or Finite Volume methods.

Various techniques for optimally selecting optimal snapshots have been proposed by Siade *et al.* (2010). From the modified snapshot set  $\bar{\mathbf{Y}} \in \mathbb{R}^{n \times d}$  we derive an optimal set of global shape functions  $\Psi_i$  for  $i = 1, 2, \dots, e$  with  $e$  is the number of global shape functions with non-zero eigenvalues, so that  $e \leq d$ . The functions are called principal modes, and are found by performing a PCA (see De Vuyst (2009)).

To find the reduced basis set, a data set of  $d$  'snapshots' across time and space (see Sirovich (1987)) is assembled as  $\mathbf{H} = [\mathbf{h}_1^*, \mathbf{h}_2^*, \dots, \mathbf{h}_d^*]$  where each  $\mathbf{h}_i^* \in \mathbb{R}^n$ ,  $\mathbf{H} \in \mathbb{R}^{n \times d}$ . These snapshots are 0 on the exterior, or Dirichlet boundary nodes, as introduced in Eq. 2.62. Each snapshot is in dimension  $\mathbb{R}^n$  when projected onto the local basis set  $v_i$ ,  $i = 1, \dots, n$ . In the case of a finite difference or finite volume simulation, each  $v_i$  will be represented by the discretised volumes.

Each snapshot can therefore be represented as a point in an  $n$ -dimensional space. The cloud of snapshots, or snapshot set, is shown in Fig. 3.1. In order to find the principal components of the snapshot set, the snapshot set is first centred about the mean of the snapshot set  $\mathbf{G}^h$  as follows:

$$\mathbf{Y}_i = \mathbf{H}_i - \mathbf{G}^h, \quad i = 1, \dots, d \quad (3.1)$$

Instead of choosing  $\mathbf{G}^h$  as the mean, it can be chosen as either the initial condition

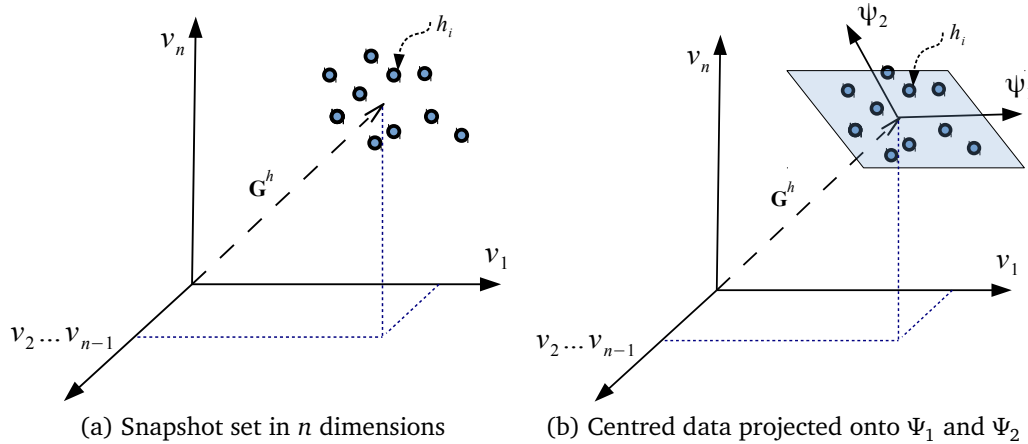


Figure 3.1: Snapshot set being projected onto two eigenvectors

(see De Vuyst (2009)) of the simulation, the steady state condition (see McPhee and Yeh (2008)) or simply  $\mathbf{0}$  (see Siade *et al.* (2010)). The modified snapshot set  $\mathbf{Y}$  is then normalised and denoted as  $\bar{\mathbf{Y}}$ . Now we find a reduced basis set by iteratively finding orthogonal directions  $\Psi_i$  for  $i = 1, \dots, e$  onto which the projection of  $\bar{\mathbf{Y}}$  has a maximum variance, where each successive  $\Psi_i$  is a solution to the optimisation problem:

$$\min_{\Psi_i} \|\bar{\mathbf{Y}} - \text{proj}_{\Psi} \bar{\mathbf{Y}}\|^2 \quad \text{with } \langle \Psi_i, \Psi_j \rangle = 0, \quad j = 1, \dots, i-1 \quad (3.2)$$

Finally,  $\{\Psi_1, \dots, \Psi_e\}$  is the subspace of dimension  $e$  that best approximates the data given in the snapshot set. It has been shown De Vuyst (2009) that the optimal reduced basis set for the optimisation problem in Eq. 3.2 are the eigenvectors of the correlation matrix  $\mathbf{M}_{corr}$ :

$$\mathbf{M}_{corr} = \bar{\mathbf{Y}}\bar{\mathbf{Y}}' \quad (3.3)$$

where  $\mathbf{M}_{corr}$  is symmetric and positive with rank  $r \leq d$ , and typically  $r \ll n$ . The eigenvectors are derived indirectly from a method known as singular value decomposition (see De Vuyst (2009) and Pinnau (2008)), since  $\mathbf{M}_{corr}$  is often too large to compute directly. The singular value decomposition of the  $n \times d$  matrix  $\bar{\mathbf{Y}}$  is given by:

$$\bar{\mathbf{Y}} = \mathbf{U}\mathbf{h}'\Sigma\mathbf{V} \quad (3.4)$$

where  $\mathbf{V}$  is a  $d \times d$  matrix, where the columns are orthogonal, and the matrix contains the eigenvectors of the matrix  $\bar{\mathbf{Y}}'\bar{\mathbf{Y}} \in \mathbb{R}^{d \times d}$ . Similarly,  $\mathbf{U}^h$  is a  $n \times n$  matrix that has orthogonal columns, and the columns represent the eigenvectors of  $\bar{\mathbf{Y}}\bar{\mathbf{Y}}' \in \mathbb{R}^{n \times n}$ . Lastly,  $\mathbf{\Sigma}$  is a real diagonal matrix containing the singular, or eigenvalues, of the matrix decomposition:

$$\mathbf{\Sigma} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & & 0 \\ 0 & 0 & & \lambda_d \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (3.5)$$

The set of eigenvectors representing the reduced basis set of a snapshot set is represented henceforth as follows:

$$\mathbf{U}^h = \{\Psi_1, \dots, \Psi_e\} \quad (3.6)$$

Each eigenvector  $\Psi_i$  has a corresponding eigenvalue  $\lambda_i$ , where  $\lambda_i > 0$ . The eigenvectors are sorted in decreasing order of ‘energy’, where the level of energy is determined by the magnitude of the corresponding eigenvalues. The ‘energy’ contained by each eigenvector is calculated as follows:

$$E_i = \frac{\lambda_i}{\sum_{j=1}^e \lambda_j} \quad (3.7)$$

To represent the original data, a smaller number ( $m^h \leq e$ ) of eigenvectors can be chosen. The value of  $m^h$  is selected by  $\sum_{i=1}^{m^h} E_i > \Theta_{min}$ , where  $\Theta_{min}$  is selected to be at least 99% (see Vermeulen *et al.* (2004) and Astrid *et al.* (2008)). In being able to choose the number of eigenvectors  $m^h$ , it is possible to place a limit on the error. An increase in the number of eigenvectors typically leads to a corresponding increase in time and corresponding decrease in the error, as demonstrated in an example in section 4.2.

### 3.2.2 Galerkin POD method

Once we have the RBS, or global shape functions, it is possible to create a Reduced Order Model (ROM) of the full FE formulation of the Richards’ equation. In this section we show the derivation of the Galerkin technique with POD applied to the Richards’ equation. The ROM development follows the Galerkin method used in FE discretisation, with the exception that it uses the first  $m^h$  global shape functions  $\Psi_i$ ,  $i = 1, 2, \dots, m^h$ , in the place of ordinary basis functions  $v_i$ ,  $i = 1, \dots, n$ .

With the set of global shape functions, the response is now approximated by a projection onto the shape functions, with the Dirichlet boundaries values explicitly represented, as:

$$\hat{h}(\mathbf{x}, t) = \sum_{j=1}^{m^h} \alpha_j(t) \Psi_j(\mathbf{x}) + G^h(\mathbf{x}) + b^h(t) \quad (3.8)$$

where  $\hat{h}$  is the head pressure approximated by the global shape functions,  $\alpha_j$  is the time-varying weight value assigned to each global shape function  $\Psi_j$ ,  $G^h$  is the mean of the snapshot set  $\mathbf{H}$ , and  $b^h$  is a function containing the Dirichlet boundary values. Each global shape function will be 0 at the Dirichlet nodes, since the snapshot set  $\bar{\mathbf{Y}}$  is zero at the Dirichlet boundary nodes, shown in Fig. 3.2. For brevity, we do not rigorously present time or the spatial axes ( $t$  and  $\mathbf{x}$ ) in the following equations.

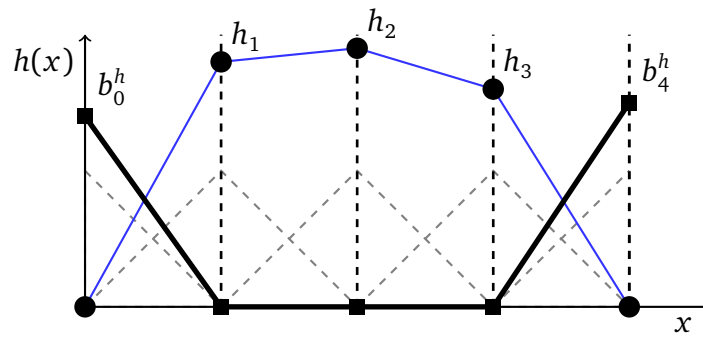


Figure 3.2: One dimensional grid with Dirichlet boundary values separated from the pressure head response

With the pressure head response defined in terms of the Reduced Basis Set, the Richards' equation is now be reformulated using this reduced set. As a reminder, the weak formulation used in Eq. 2.45 is:

$$\underbrace{\int_{\Omega} C(h)w \frac{\partial h}{\partial t} + \int_{\Omega} K(h) \nabla w \cdot \nabla h}_{LHS} = \int_{\Gamma_n} w K(h) \nabla(h+z) \cdot \mathbf{n} - \int_{\Omega} K(h) \frac{\partial w}{\partial z} \quad (3.9)$$

The first term in Eq. 3.9 can be discretised according to the Implicit Euler approach:

$$\int_{\Omega} Cw \frac{\partial h}{\partial t} \rightarrow \int_{\Omega} C(\hat{h}^{\tau+1})w \frac{(\hat{h}^{\tau+1} - \hat{h}^{\tau})}{\delta t} \quad (3.10)$$

In the Implicit Euler scheme, the non-linear functions are always evaluated at the next time step as  $K(\hat{h}^{\tau+1})$  as  $C(\hat{h}^{\tau+1})$ . For brevity these arguments are dropped, and the non-linear functions are represented as  $K$  and  $C$  in the following

derivations. The weak formulation in Eq. 3.9 is reformulated by substituting the time discretisation in Eq. 3.10 into it, and multiplying by  $\delta t$  as follows :

$$\underbrace{\int_{\Omega} C w \hat{h}^{\tau+1} + \delta t \int_{\Omega} K \nabla w \cdot \nabla \hat{h}}_{LHS} = \int_{\Omega} C w \hat{h}^{\tau} + \delta t \int_{\Gamma_n} w f^n \cdot \mathbf{n} - \delta t \int_{\Omega} K \frac{\partial w}{\partial z} \quad (3.11)$$

Now if we substitute the approximate pressure head response in Eq. 3.8 into the weak formulation, the left hand side terms *LHS* in Eq. 3.11 are expanded as follows:

$$LHS = \int_{\Omega} C w \left[ \sum_{j=1}^{m^h} (\alpha_j^{\tau+1} \Psi_j) + b^{h,\tau+1} + G^h \right] + \delta t \int_{\Omega} K \nabla w \cdot \nabla \left[ \sum_{j=1}^{m^h} (\alpha_j^{\tau+1} \Psi_j) + b^{h,\tau+1} + G^h \right] \quad (3.12)$$

where  $b^{h,\tau+1}$  is evaluated at time step  $\tau + 1$ . In this formulation,  $\alpha^{\tau+1}$  is the unknown, and so the terms containing  $\alpha$  are grouped:

$$\begin{aligned} LHS = & \sum_{j=1}^{m^h} \left( \int_{\Omega} C w (\alpha_j^{\tau+1} \Psi_j) + \delta t \int_{\Omega} K \nabla w \cdot \nabla (\alpha_j^{\tau+1} \Psi_j) \right) \\ & + \int_{\Omega} C w (b^{h,\tau+1} + G^h) + \delta t \int_{\Omega} K \nabla w \cdot \nabla (b^{h,\tau+1} + G^h) \end{aligned} \quad (3.13)$$

Following the Galerkin approach introduced in section 2.2.2, the test function  $w$  is chosen as follows:

$$w = \Psi_i(\mathbf{x}), \quad i = 1, 2, \dots, m^h \quad (3.14)$$

The new test function  $w$  and the LHS terms in Eq. 3.13 are now substituted into the original variational formulation in Eq. 3.11 to yield:

$$\begin{aligned} \forall i, \quad \sum_{j=1}^{m^h} \left( \int_{\Omega} C \Psi_i (\alpha_j^{\tau+1} \Psi_j) + \delta t \int_{\Omega} K \nabla \Psi_i \cdot \nabla (\alpha_j^{\tau+1} \Psi_j) \right) = \\ \int_{\Omega} C \Psi_i \hat{h}^{\tau} + \delta t \int_{\Gamma_n} \Psi_i f_n \cdot \mathbf{n} - \delta t \int_{\Omega} K \frac{\partial \Psi_i}{\partial z} \\ - \underbrace{\int_{\Omega} C \Psi_i G^h - \delta t \int_{\Omega} K \nabla \Psi_i \cdot \nabla G^h}_{\hat{\mathbf{V}}_i^g} \\ - \underbrace{\int_{\Omega} C \Psi_i b^{h,\tau+1} - \delta t \int_{\Omega} \nabla K \Psi_i \cdot \nabla b^{h,\tau+1}}_{\hat{\mathbf{V}}_i^d} \end{aligned} \quad (3.15)$$

The above equation can be compactly represented as a system of Ordinary Differential Equations (ODE's) as follows:

$$\hat{\mathbf{M}}^s(\hat{h}) \alpha^{\tau+1} + \delta t \hat{\mathbf{M}}^p(\hat{h}) \alpha^{\tau+1} = \hat{\mathbf{V}}^s(\hat{h}) + \delta t [\hat{\mathbf{V}}^n(f_n) - \hat{\mathbf{V}}^p(\hat{h})] - \hat{\mathbf{V}}^g(\hat{h}) - \hat{\mathbf{V}}^d(\hat{h}) \quad (3.16)$$

where each matrix term  $\hat{\mathbf{M}}^{\mathbf{s},\mathbf{p}}$  represents the reduced form of the full matrix  $\mathbf{M}^{\mathbf{s},\mathbf{p}}$ , and similarly the hat on each vector represents the reduced form of the vector. These reduced matrices and vectors are similar to the terms in the full FE derivation (see section 2.2.4), with the addition of the gravity mean term ( $\hat{\mathbf{V}}^{\mathbf{s}}$ ) and the explicit Dirichlet vector ( $\hat{\mathbf{V}}^{\mathbf{d}}$ ). The response at the current time step is indexed with  $\tau$ , and the next time step is indexed with  $\tau + 1$ , and the Euler time discretisation coupled with the Picard iteration method is applied to Eq. 3.16 so that:

$$[\hat{\mathbf{M}}^{\mathbf{s}}(\hat{h}^{\tau+1}) + \delta t \hat{\mathbf{M}}^{\mathbf{p}}(\hat{h}^{\tau+1})] \boldsymbol{\alpha}^{\tau+1} = \hat{\mathbf{V}}^{\mathbf{s}}(\hat{h}^{\tau+1}) + \delta t [\hat{\mathbf{V}}^{\mathbf{n}}(f_n^{\tau+1}) - \hat{\mathbf{V}}^{\mathbf{p}}(\hat{h}^{\tau+1})] - \hat{\mathbf{V}}^{\mathbf{g}}(\hat{h}^{\tau+1}) - \hat{\mathbf{V}}^{\mathbf{d}}(\hat{h}^{\tau+1}) \quad (3.17)$$

With the reduced formulation available in matrix form, it is now necessary to find a way to create and solve the integral matrices and vectors given in Eq. 3.15. The relationship between the full and reduced forms is explained in the following section, where the full matrices have to be computed at each time step before they are reduced.

### 3.2.3 Reducing full Finite-Element matrices

It is possible to derive the integral matrices and vectors defined in the proper function space in terms of the integral matrices already defined, by using the ordinary basis as follows:

$$\Psi_k(\mathbf{x}) = \sum_{i=1}^n \Psi_{ik} v_i(\mathbf{x}) \quad (3.18)$$

where  $\Psi_{ik}$  is a scalar value at the  $i^{\text{th}}$  node of the  $k^{\text{th}}$  global shape function, and  $v_i$  is the local basis function at the  $i^{\text{th}}$  node. Let us consider the terms in matrix  $\hat{\mathbf{M}}^{\mathbf{s}}(\hat{h})$  as an example:

$$\hat{\mathbf{M}}_{kl}^{\mathbf{s}}(\hat{h}) = \int_{\Omega} C \Psi_k \Psi_l \quad (3.19)$$

Substituting Eq. 3.18 into Eq. 3.19, we obtain:

$$\hat{\mathbf{M}}_{kl}^{\mathbf{s}}(\hat{h}) = \int_{\Omega} C \sum_{i=1}^n (\Psi_{ik} v_i) \sum_{j=1}^n (\Psi_{jl} v_j) \quad (3.20)$$

The eigenvectors are separated from the integral matrix as follows:

$$\hat{\mathbf{M}}_{kl}^{\mathbf{s}}(\hat{h}) = \Psi'_k \begin{bmatrix} \int_{\Omega} C v_1 v_1 & \dots & \int_{\Omega} C v_1 v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} C v_n v_1 & \dots & \int_{\Omega} C v_n v_n \end{bmatrix} \Psi_l \quad (3.21)$$

effectively replicating the integral matrix  $\mathbf{M}^{\mathbf{s}}(h)$  derived in Eq. 2.51. The reduced matrix is now represented as:

$$\hat{\mathbf{M}}^{\mathbf{s}}(\hat{h}) = \mathbf{U}^{h'} \begin{bmatrix} \int_{\Omega} C v_1 v_1 & \dots & \int_{\Omega} C v_1 v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} C v_n v_1 & \dots & \int_{\Omega} C v_n v_n \end{bmatrix} \mathbf{U}^h = \mathbf{U}^{h'} \mathbf{M}^{\mathbf{s}}(\hat{h}) \mathbf{U}^h \quad (3.22)$$

which effectively reduces the size of the original matrix from  $[n \times n]$  to  $[m^h \times m^h]$ , hence the name *Reduced Order Model*. Similarly, the stiffness matrix in Eq. 2.52 is reduced as follows:

$$\hat{\mathbf{M}}^p(\hat{h}) = \mathbf{U}^{h'} \begin{bmatrix} \int_{\Omega} K \nabla v_1 \cdot \nabla v_1 & \dots & \int_{\Omega} K \nabla v_1 \cdot \nabla v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} K \nabla v_n \cdot \nabla v_1 & \dots & \int_{\Omega} K \nabla v_n \cdot \nabla v_n \end{bmatrix} \mathbf{U}^h = \mathbf{U}^{h'} \mathbf{M}^p(\hat{h}) \mathbf{U}^h \quad (3.23)$$

Now the reduced terms in the left hand side of Eq. 3.16 are grouped as:

$$\hat{\mathbf{M}}(\hat{h}^{\tau+1}) = \mathbf{U}^{h'} [\mathbf{M}^s(\hat{h}^{\tau+1}) + \delta t \mathbf{M}^p(\hat{h}^{\tau+1})] \mathbf{U}^h \quad [m^h \times m^h] \quad (3.24)$$

In order to create the reduced matrices and vectors, the full mass and stiffness matrices need to be calculated several times during every time step, which makes this approach computationally expensive to evaluate, as reported also by Astrid *et al.* (2008). A method to reduce this computational cost is developed later in this thesis in chapter 5.2. Using the same approach, the reduction of the vector terms to vectors of dimension  $[m^h \times 1]$  is addressed in the following sections.

### 3.2.4 Reducing the vector terms

In this section, the remainder of the vector terms on the right hand side of Eq. 3.17 are treated. The non-linear functions  $K(\hat{h}^{\tau+1})$  as  $C(\hat{h}^{\tau+1})$  are again represented as  $K$  and  $C$ . The vector terms are recalled:

$$\hat{\mathbf{V}}(\hat{h}^{\tau+1}) = \hat{\mathbf{V}}^s(\hat{h}^{\tau+1}) + \delta t [\hat{\mathbf{V}}^n(f_n^{\tau+1}) - \hat{\mathbf{V}}^p(\hat{h}^{\tau+1})] - \hat{\mathbf{V}}^g(\hat{h}^{\tau+1}) - \hat{\mathbf{V}}^d(\hat{h}^{\tau+1}) \quad (3.25)$$

The first vector term is recalled, considering the  $i^{\text{th}}$  entry in the vector as follows:

$$\hat{\mathbf{V}}_i^s(\hat{h}^{\tau+1}) = \int_{\Omega} C \Psi_i \hat{h}^{\tau} \quad \text{for } i = 1, \dots, m^h \quad (3.26)$$

The pressure head  $\hat{h}^{\tau}$  are now written in terms of the local basis functions:

$$\hat{h}^{\tau}(\mathbf{x}) = \sum_{i=1}^n \hat{h}_i^{\tau} v_i(\mathbf{x}) \quad (3.27)$$

Substituting the head pressure and eigenvector as a function of local shape functions (Eq. 3.27 and Eq. 3.18) into Eq. 3.26 yields:

$$\hat{\mathbf{V}}_i^s(\hat{h}^{\tau+1}) = \mathbf{U}_i^{h'} \begin{bmatrix} \int_{\Omega} C v_1 v_1 & \dots & \int_{\Omega} C v_1 v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} C v_n v_1 & \dots & \int_{\Omega} C v_n v_n \end{bmatrix} \hat{\mathbf{h}}^{\tau} = \mathbf{U}_i^{h'} \mathbf{M}^s \hat{\mathbf{h}}^{\tau} \quad (3.28)$$

where  $\mathbf{U}_i^h$  is the vector corresponding to the  $i^{\text{th}}$  global shape function  $\Psi_i$ . The first vector term is seen to be a function of the mass matrix  $\mathbf{M}^s$ . The first vector term can now be written in its entirety as:

$$\hat{\mathbf{V}}^s(\hat{h}^{\tau}) = \mathbf{U}^{h'} \mathbf{M}^s \hat{\mathbf{h}}^{\tau} \quad (3.29)$$



The second term is the Neumann vector term ( $\hat{\mathbf{V}}^n$ ), which can also be approximated by projecting the full Neumann vector term onto the reduced basis set  $\mathbf{U}^h$ . If we again consider the  $i^{\text{th}}$  global shape function, the  $i^{\text{th}}$  term in the reduced vector can be found as:

$$\hat{\mathbf{V}}_i^n(f_n^{\tau+1}) = \int_{\Gamma_n} f_n^{\tau+1} \Psi_i \cdot \mathbf{n} = \mathbf{U}_i^{h'} \begin{bmatrix} \int_{\Gamma_n} f_n^{\tau+1} v_1 \cdot \mathbf{n} \\ \vdots \\ \int_{\Gamma_n} f_n^{\tau+1} v_n \cdot \mathbf{n} \end{bmatrix} = \mathbf{U}_i^{h'} \mathbf{V}^n(f_n^{\tau+1}) \quad (3.30)$$

Using the entire reduced basis set, the Neumann vector is represented in reduced form as:

$$\hat{\mathbf{V}}^n(f_n^{\tau+1}) = \mathbf{U}^{h'} \mathbf{V}^n(f_n^{\tau+1}) \quad [m \times 1] \quad (3.31)$$

The conductivity vector is similarly reduced by projecting the full matrix onto the RBS:

$$\hat{\mathbf{V}}_i^p(\hat{h}^{\tau+1}) = \int_{\Omega} K(\hat{h}^{\tau+1}) \frac{\partial \psi_i}{\partial z} = \mathbf{U}_i^{h'} \begin{bmatrix} \int_{\Omega} K \frac{\partial v_1}{\partial z} \\ \vdots \\ \int_{\Omega} K \frac{\partial v_n}{\partial z} \end{bmatrix} = \mathbf{U}_i^{h'} \mathbf{V}^p(\hat{h}^{\tau+1}) \quad (3.32)$$

so that the complete reduced vector is written as:

$$\hat{\mathbf{V}}^p(\hat{h}^{\tau+1}) = \mathbf{U}^{h'} \mathbf{V}^p(\hat{h}^{\tau+1}) \quad (3.33)$$

A trend can be seen where the original vectors from the full Finite Element discretisation are simply multiplied by the reduced basis set, reducing the vectors from length  $n$  to length  $m^k$ . In the following section, the two remaining terms containing the  $G^h$  and  $b^{h,\tau+1}$  terms, which do not exist in the original full discretisation, are treated.

### 3.2.5 Reduction of Dirichlet and gravity mean terms

The Finite Element Method requires the basis functions to be zero on the edge of the design space. This edge is typically represented by the Dirichlet boundaries. In the formulation of Eq. 3.8, the Dirichlet boundary values are explicitly separated from the interior nodes to meet this condition. The Dirichlet term from Eq. 3.15 is recalled:

$$\hat{\mathbf{V}}_i^d(\hat{h}^{\tau+1}) = \int_{\Omega} C \Psi_i b^{h,\tau+1} - \delta t \int_{\Omega} K \nabla \Psi_i \cdot \nabla b^{h,\tau+1} \quad (3.34)$$

If the eigenvector  $\Psi_i$  and the Dirichlet function  $b^{h,\tau+1}$  are written as a function of the original basis set, each  $i^{th}$  term in the equation can be written as:

$$\hat{\mathbf{V}}_i^d(\hat{h}^{\tau+1}) = \mathbf{U}_i^{h'} \begin{bmatrix} \int_{\Omega} C v_1 v_1 & \dots & \int_{\Omega} C v_1 v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} C v_n v_1 & \dots & \int_{\Omega} C v_n v_n \end{bmatrix} \mathbf{b}^{t,\tau+1} + \delta t \mathbf{U}_i^{h'} \begin{bmatrix} \int_{\Omega} K \nabla v_1 \cdot \nabla v_1 & \dots & \int_{\Omega} K \nabla v_1 \cdot \nabla v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} K \nabla v_n \cdot \nabla v_1 & \dots & \int_{\Omega} K \nabla v_n \cdot \nabla v_n \end{bmatrix} \mathbf{b}^{t,\tau+1} \quad (3.35)$$

The mass and stiffness matrices are again produced, so that the Dirichlet term can be written compactly as:

$$\hat{\mathbf{V}}^d(\hat{h}^{\tau+1}) = \mathbf{U}^{h'} [\mathbf{M}^s(\hat{h}^{\tau+1}) + \delta t \mathbf{M}^p(\hat{h}^{\tau+1})] \mathbf{b}^{t,\tau+1} \quad (3.36)$$

Following the exact same derivation, the gravity mean term can be written as:

$$\hat{\mathbf{V}}^g(\hat{h}^{\tau+1}) = \mathbf{U}^{h'} [\mathbf{M}^s(\hat{h}^{\tau+1}) + \delta t \mathbf{M}^p(\hat{h}^{\tau+1})] \mathbf{G}^h \quad (3.37)$$

It is interesting to note that the mass and stiffness matrices are reused a number of times throughout the development of the reduced order model. With the reduced matrices and vectors now developed, the pressure head at the next time step is solved in the following section.

### 3.2.6 Solving the reduced model equation

In order to find the initial weight values  $\alpha^0$ , the initial conditions  $\mathbf{h}^{init}$  are projected onto the reduced basis set  $\mathbf{U}^h$ . Now for each following time step, Eq. 3.16 is solved by using the matrix expression developed in Eq. 3.24 and the vector expression in Eq. 3.25 to find the weight vectors at the next time step:

$$\alpha^{\tau+1} = \hat{\mathbf{M}}^{-1}(\hat{h}^{\tau+1}) \hat{\mathbf{V}}(\hat{h}^{\tau+1}) \quad (3.38)$$

The pressure head response at the next time step can now be calculated:

$$\hat{\mathbf{h}}^{\tau+1} = \mathbf{U}^h \alpha^{\tau+1} + \mathbf{G}^h \quad (3.39)$$

The system to be solved is now of the magnitude  $[m^h \times m^h]$ , where  $m^h$  is the number of shape functions chosen. An error measure to evaluate the developed ROM accuracy is now shown.

### 3.2.7 Error calculation

To calculate the error between full and Reduced Order Models, we use an adaptation of the Relative Root Mean Squared (RRMS) error used by Vermeulen *et al.*

(2004) as follows:

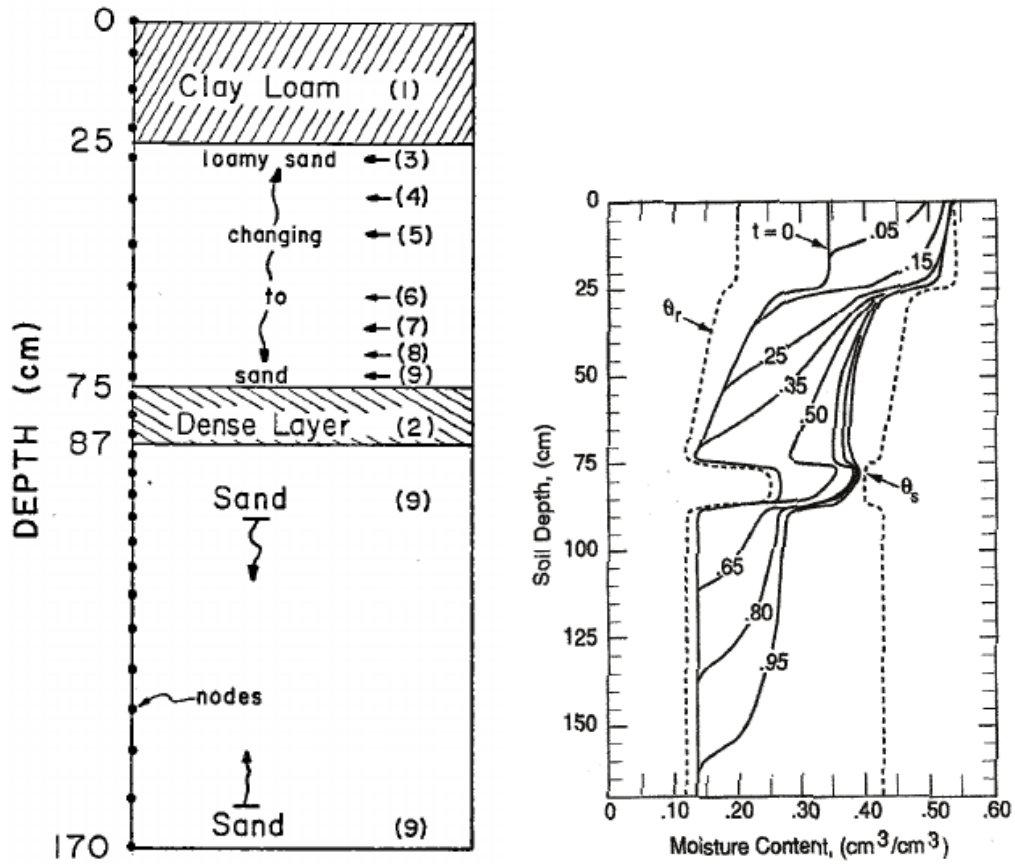
$$\text{RRMS}_{m^h}(\mathbf{H}(\mathbf{p}), \hat{\mathbf{H}}(\mathbf{p})) = 100 \cdot \frac{1}{nts} \sum_{i=1}^{nts} \sqrt{\frac{\|\mathbf{h}_i(\mathbf{p}) - \hat{\mathbf{h}}_i(\mathbf{p})\|^2}{\|\mathbf{h}_i(\mathbf{p}) - \mathbf{G}_h\|^2}} \quad (3.40)$$

where  $\mathbf{H}(\mathbf{p}) = \{\mathbf{h}_1(\mathbf{p}), \dots, \mathbf{h}_{nts}(\mathbf{p})\}$  is the full model set of responses over time,  $nts$  is the number of time steps and  $\mathbf{p}$  is the parameter set. The set of responses approximated by the ROM is given as  $\hat{\mathbf{H}}(\mathbf{p}) = \{\hat{\mathbf{h}}_1(\mathbf{p}), \dots, \hat{\mathbf{h}}_{nts}(\mathbf{p})\}$ , and the subscript  $m^h$  refers to the error of a ROM built using a total of  $m^h$  global shape functions. This error measure is used throughout this research to determine the accuracy of the ROM models. The method developed is now tested on an example problem.

### 3.3 Benchmark example of van Genuchten

In an article by van Genuchten (1978), he develops an example infiltration problem with water infiltrating into a multi-layer soil sample. It has subsequently been used as a benchmark of various numerical approaches (see Diersch (2014)). The same example is replicated here with the FE method, as well as the ROM approach explained in this chapter in order to test whether the lower order solution using POD can replicate the problem. In the application of POD, it is found that, despite the complexity and non-linearity in the parameters and response, the benchmark can be replicated with reasonable accuracy.

The physical layout of the test, as well as the response over a period of one day is shown in Fig. 3.3, where the wetting front is shown to progress from  $t = 0$  d to  $t = 0.95$  d on Fig. 3.3b. The parameters used in the example are shown in Table 3.1, and a constant flux of  $q_{in} = 0.95$  m/d is supplied at the upper surface. An initial condition of  $h = 3.5$  cm is given. At the lower extent of the model, a flux of  $q_{out} = -4$  m/d is imposed. At the surface there is a 25 cm thick layer of clay loam, with a layer below it that varies from loamy sand to sand. Under this layer is a layer of low conductivity, and finally the base of the problem consists entirely of sand. Although the original problem consists of a recharge and a discharge phase, we will only consider the recharge phase.



(a) Original physical layout of soil

(b) The results obtained by van Genuchten (1978) for moisture content in the soil, where  $\theta_s$  and  $\theta_r$  represent  $\epsilon_{l,s}$  and  $\epsilon_{l,r}$  respectively

Figure 3.3: The benchmark problem as presented by van Genuchten (1978). The different layers of soil are shown with labels corresponding to Table 3.1, and the soil moisture increasing with time as the wetting front descends

The FE response is used to generate the snapshots from which the ROM is created. In this example, a total of 35 global shape functions are chosen to replicate the response from 300 snapshots. The snapshots are taken at intervals of 288s. As prescribed in Vermeulen *et al.* (2004), 99% of the total weight of the eigenvectors, resulting in a choice of 35 global shape functions to accurately capture the original response. However, the ROM requires more time to execute than the full model (also reported by Astrid *et al.* (2008)). This is due to the GPOD approach requiring the calculation of the full matrices, as well as the creation of the ROM. However, it can be seen that even a response as complex and non-linear as the one presented by van Genuchten (1978) can be replicated using the GPOD method.

As is typically shown in other POD demonstration examples, we show a subset of the global shape functions for the objective function (in this case the pressure head) in Fig. 3.4a. Due to the discontinuities in the parameters, there are discontinuities in the global shape functions. In this example,  $G_h$  is taken to be zero for simplicity. The first 5 weight values  $\alpha$  are shown to vary over time in Fig. 3.4b. As can be seen they follow a continuous trajectory. The FE and ROM response,  $h$  and  $\hat{h}$  respectively, are shown in Fig. 3.4c. The ROM response  $\hat{h}$  is constructed entirely of the reduced basis set, and the approximation in Fig. 3.4c is seen to fluctuate significantly around the wetting front, arising from the discontinuities in the original response. The error with 35 eigenfunctions is  $RRMS_{35} = 2.2\%$ , which can be improved with a snapshot selection technique. In the following section a study is done on the accuracy when the number of eigenvectors are increased. From this example it can be assumed that the GPOD method is suitable for application on the non-linear Richards' equation, and is further explored in the following chapters. A code snippet of the ROM implementation is given in Appendix B in Fig. B.2.

Table 3.1: Parameters used by van Genuchten (1978)

Soil layer	Type	$\varepsilon_{l,r}$	$\varepsilon_{l,s}$	$1/h_b$ [cm <sup>-1</sup> ]	$n$	$K_s$ [cm/d]	$S_s$ [cm <sup>-1</sup> ]
1	Clay loam	0.2	0.54	0.008	1.8	25	4.00E-007
2	Dense layer	0.25	0.4	0.009	3	10	5.00E-008
3	Loamy sand	0.17	0.47	0.01	2	75	1.00E-007
4		0.1611	0.4611	0.01306	2.178	132.8	1.00E-007
5		0.15	0.45	0.0108	2.4	205	1.00E-007
6		0.14	0.44	0.0112	2.6	270	1.00E-007
7		0.1311	0.4311	0.01156	2.778	327.8	1.00E-007
8		0.1244	0.4244	0.01182	2.911	371.1	1.00E-007
9	Sand	0.12	0.42	0.012	3	400	1.00E-007

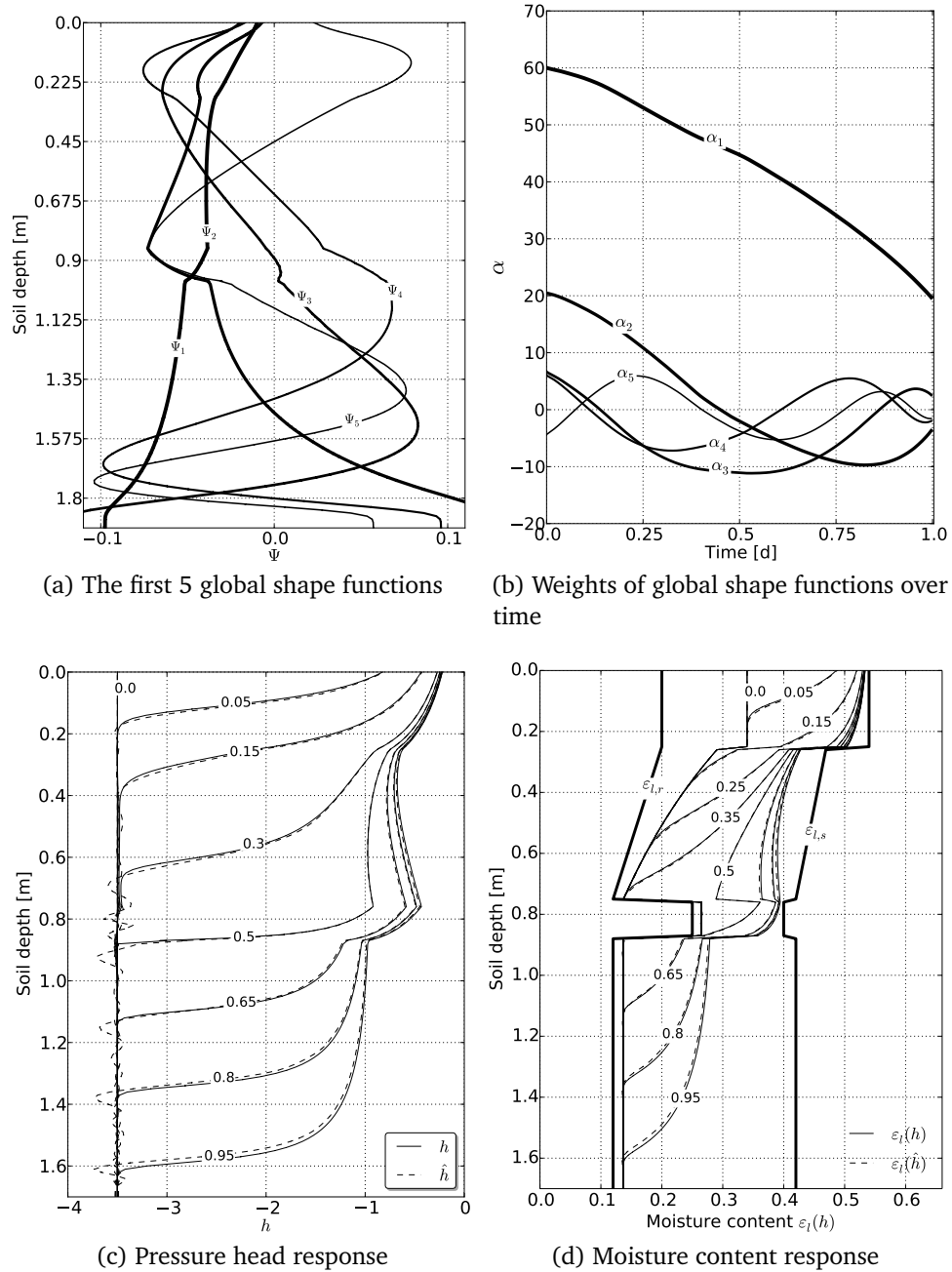


Figure 3.4: A replication of the van Genuchten problem with the FE method, derived by implementing the method presented in this research in FreeFem++, and the subsequent ROM results using POD

## 3.4 Conclusion

In this chapter, the GPOD approach is applied to the Richards' equation, explaining the reduction of all terms in the ROM creation. The method is seen to replicate a non-linear benchmark problem with reasonable accuracy, but without suitable speed-up characteristics. In order to better understand the time and accuracy yielded by the GPOD method, two problems are analysed in the following chapter, one a saturated problem, and one an unsaturated problem. This is followed by the development of a method that yields feasible speed-up times.

## Chapter 4

# Applying POD to saturated and unsaturated groundwater case studies

**I**N this chapter the use of the Galerkin Proper Orthogonal Decomposition (GPOD) method is applied to two example case studies. The first is a saturated flow problem applied to a groundwater site in the Kogelberg area in South Africa. In this problem, the recharge is an unknown boundary value, and the model parameters need to be calibrated to allow us to determine the amount of water that infiltrates as a result of the difference in precipitation and evapotranspiration. A numerical model is created of the Kogelberg site, and subsequently calibrated to the response at one well. A reduced order model of the full model is created using the GPOD method, which is shown to capture the system response at the well with reasonable accuracy. The system is approximated as a confined aquifer, and errors arise due to the model being unable to capture slower infiltration in the unsaturated zone. However, due to the size of the spatial domain and time limitations on the research, the model is not developed as an unsaturated problem, testing only the interpolation qualities of the GPOD approach on a saturated model.

The second example is a synthetic unsaturated problem that was created as a case study on which to demonstrate the GPOD method in a non-linear environment, and to obtain an idea of the run-time and accuracy of the method. In both examples, the speed-up times attained in the application of POD on saturated and unsaturated flow are measured. Furthermore, the interpolation properties of the GPOD method is tested across the parameter space.

In both examples, there are constant Dirichlet boundary conditions, and time-varying Neumann boundary conditions. In the saturated example, the application of POD yields very significant speed-up times, as seen in the literature (section 3.1.1), as well as excellent interpolative qualities. In contrast, the GPOD approach applied to the unsaturated example is shown to be infeasible in terms of time gains. In the following chapter, the GPOD method is adapted to create a POD-



based Reduced Order Model (ROM) that has significant speed-up times when compared to the full model.

## 4.1 Saturated case study: Table Mountain Group Aquifer

In this section we consider a site in the Western Cape that has been monitored and studied extensively by Colvin *et al.* (2009a), Colvin *et al.* (2009b), Barrow (2010), Jovanovic and Bugan (2010a), Jovanovic and Bugan (2010b), Jovanovic and Bugan (2011), Wu (2005) and Xu *et al.* (2009), amongst others. The site is situated in the Oudebosch catchment in the Western Cape, South Africa, and is pristine in the sense that the groundwater levels are not affected by pumping or construction activities. The aquifer group, classified as the Table Mountain Group (TMG), is under consideration as a source of water for the greater Cape Town area, and as such the monitoring activities in the cited articles form part of the investigation into the feasibility of water extraction from the TMG. A significant percentage of water is removed from the site by evapotranspiration (appendix C), which is in turn sensitive to groundwater availability. To better understand the water behaviour, a detailed model linking the groundwater to evapotranspiration is developed in this section.

The approach used in this case study allows the evaluation of quantitative recharge values, with which to make decisions on groundwater extraction. This study serves to evaluate the application of a POD model to saturated flow in terms of speed-up times and interpolation/extrapolation qualities.

### 4.1.1 Site description and assumptions

A conceptual, parameterised 3D numerical model is created that links the groundwater recharge, the evapotranspiration, and the groundwater flow. The model is assumed to be governed by saturated flow, and simulates surface and groundwater interaction in the catchment site for the years 2008 and 2009. The unknown parameters are found by calibrating the measured response to the calculated numerical response.

The site vegetation and geology is shown in Fig. 4.1a, where a large number of monitoring wells, coloured in red and orange, are seen across the model domain. These monitoring wells are used in various other related studies, in an attempt to analyse different systems responses such as water movement, evapotranspiration and solute concentration in the soil.

The well locations, the model domain and the stream are shown from an elevated position in Fig. 4.1b, with well positions from the study by Colvin *et al.* (2009a), and are shown using Google Earth. The perimeter of the site is  $\approx 10$  km long, with an area of  $\approx 6$  km<sup>2</sup>. The altitude rises to  $\approx 500$  m in the model. The

section was chosen to contain as much of the watershed as possible, following the domain selection by Jovanovic and Bugan (2010b).

The site is subject to significant fracturing in the rock, and a dual porosity porous media is assumed. Furthermore, the model assumes the following:

1. The site is governed by saturated flow.
2. The model has only a stream as a Dirichlet boundary value and daily rainfall and evapotranspiration as a Neumann boundary condition on the upper surface.
3. The values of water availability at the single monitoring well are used to determine evapotranspiration across the spatial domain.
4. The flow is anisotropic, with constant conductivity values along the directional axes as  $K_x$ ,  $K_y$  and  $K_z$ .
5. Potential evapotranspiration calculated for 2009 is assumed to be the same for 2008, since reliable climatic data for 2008 is not available.

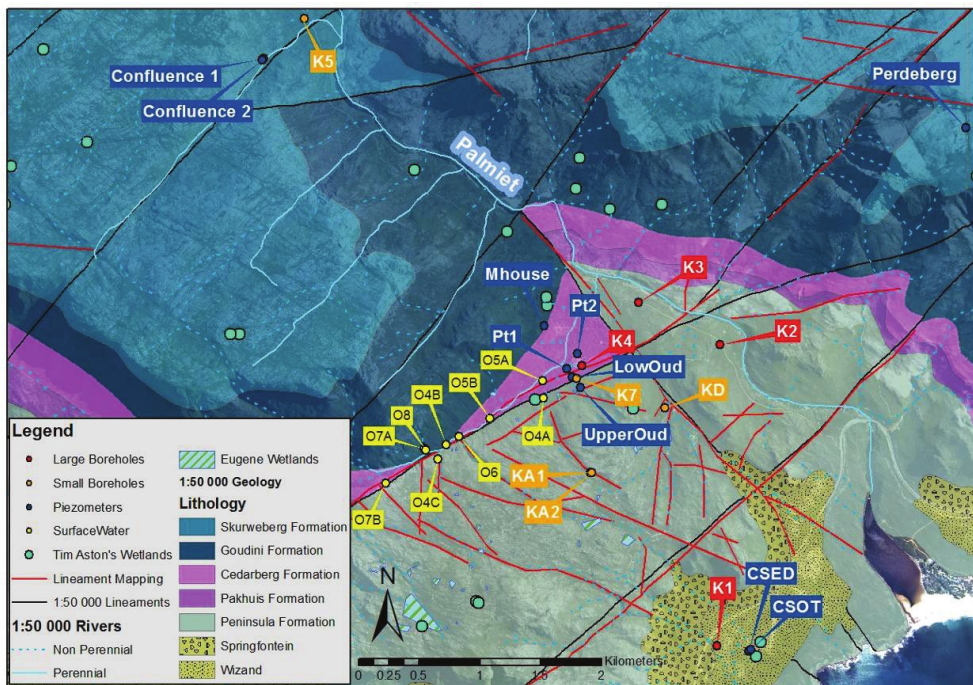
The results obtained in this case study are of limited value, since the calibration of one measuring well does not guarantee that the water movement is adequately represented across the spatial domain. To improve the spatial accuracy, calibration has to be done on multiple wells at the same time across the model domain, which has been delegated to a future research project.

#### 4.1.2 System response and evapotranspiration

The response at well  $K_2$  from January 2008 to end December 2009 is shown in Fig. 4.2a in response to rainfall over the same period, shown in Fig. 4.2b. In the well response, there is a significant fluctuation in the water table from June to December each year in response to the rainfall events, while from January to June there is very little fluctuation in the water level. This measured response is used to calibrate the numerical response by varying parameters in the porous media and evapotranspiration values.

The principal vegetation at the site is fynbos vegetation. In the report by Colvin *et al.* (2009a), it was found that “all the fynbos species tested...were vulnerable to reduced water availability”. Furthermore, it is stated that “deep rooted Proteas...are known to have root systems up to 10 m deep”. This information was used to specify the parameters in the soil and vegetation stress factors,  $K_r$  and  $K_s$  respectively (derived in appendix C.2).

For the soil evaporation stress factor  $K_r$ , two parameters need to be defined. The readily evaporable water (REW) and total evaporable water (TEW) are chosen as  $-10\text{m}$  and  $-2\text{m}$  respectively. Secondly, the transpiration stress factor  $K_s$  was chosen to change with the amount of surface water. The amount of Readily

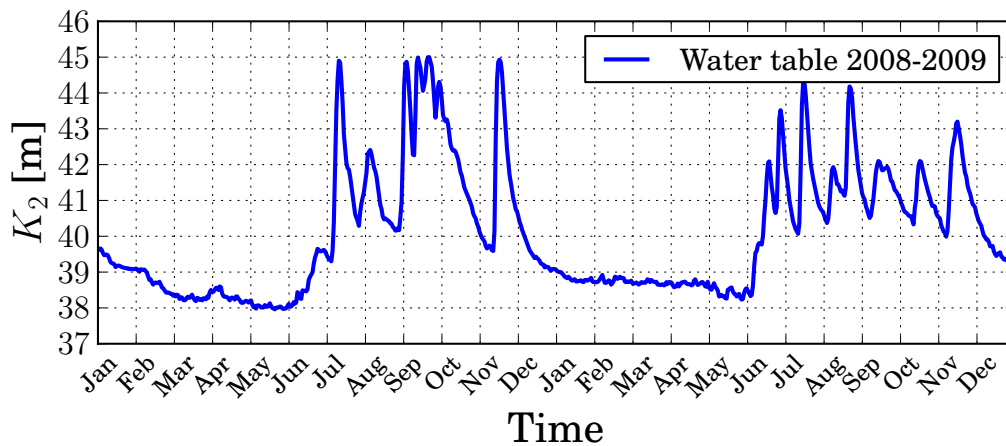


(a) Monitoring sites with outcropping geological formations (Colvin *et al.* (2009b))

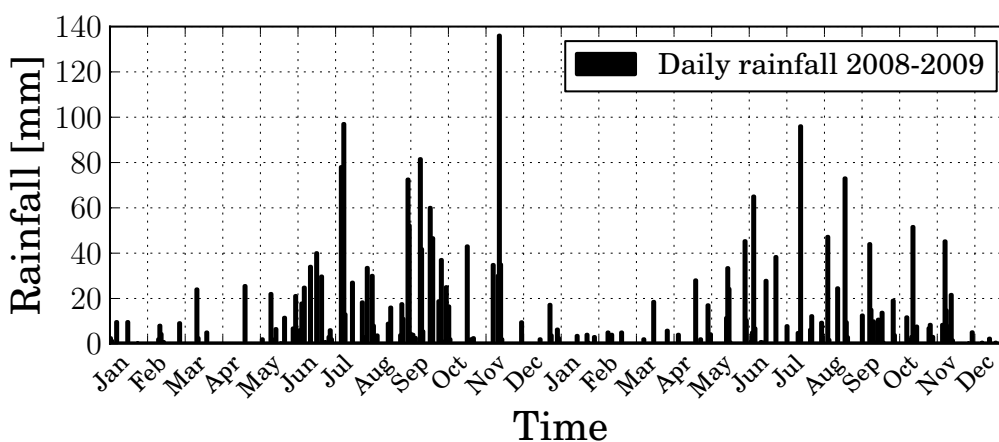


(b) Location of well markers, streams and model limits of the Kogelberg site

Figure 4.1: Perspectives of Kogelberg site

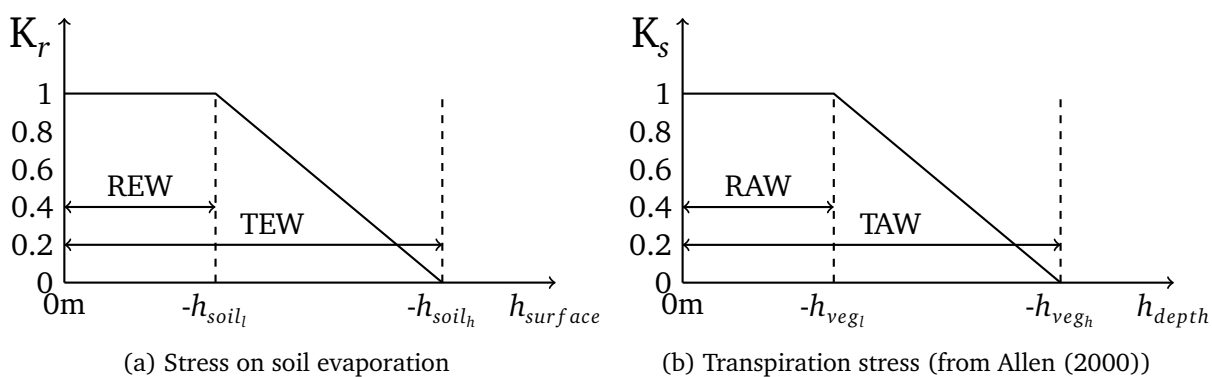


(a) Pressure head at well  $K_2$  in m.a.s.l.



(b) Rainfall in Kogelberg region for 2008 and 2009

Figure 4.2: Response in water table corresponding to rainfall



(a) Stress on soil evaporation

(b) Transpiration stress (from Allen (2000))

Figure 4.3: Stress factors due to limited water availability

Available Water (RAW) was taken at 1 m below the ground surface. The Wilting Point (WP or TAW) is chosen as 7 m below ground level as shown in Fig. 4.3b.

A crop factor is introduced to approximate the increase in transpiration in the rain season (appendix C.1). The adapted crop factor is shown in Fig. 4.4, and is given a relative increase in transpiration in the rain season, since the predominant vegetation, Fynbos, has a growth period corresponding to the rainy season over the winter months. Implementing the crop factors allows us to calculate the true evaporation as:

$$ET_c = (K_s K_{c,var} + K_r K_e) ET_0 \quad (4.1)$$

where  $K_s$  and  $K_r$  are the vegetation and soil stress factors, and  $K_{c,var}$  and  $K_e$  are the constants describing the transpiration and evaporation factors respectively.

In order to evaluate the potential evapotranspiration,  $ET_0$ , with the Penman-Monteith method (section C), a significant amount of climate data is needed. This data includes precipitation, temperature, wind speed, humidity, and the actual and saturation vapour pressures. The data sets logged and presented in the technical report by Jovanovic and Bugan (2010b) are used to calculate  $ET_0$ , shown in Fig. 4.5. The data follows a near co-sinusoidal trend due to the large influence of radiation. It is interesting to note that the  $ET_0$  is lowest in June, corresponding to the start of the rainfall season.

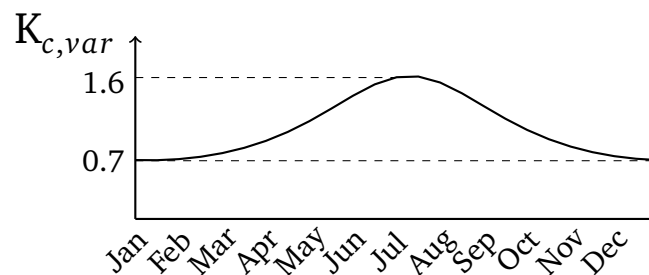


Figure 4.4: Adaptation of crop factor

### 4.1.3 Soil parameters

In the report by Xu *et al.* (2009), values for storativity for an unconfined aquifer in the Peninsula region are given as  $S_s = 5 \times 10^{-3}$  and  $S_s = 10^{-4}$ . Conductivity was found to decrease with depth in the TMG, with conductivity values at the surface in the order of  $10^{-1}$  m/d decreasing with depth to  $10^{-4}$  m/d in Xu *et al.* (2009). In the same report, proposed storativity values are in the range  $[10^{-4} - 1.2 \times 10^{-2}]$ <sup>1</sup>, and the fynbos crop factor is shown to vary between 0.2 and 0.6. These values are used as guidelines to define limits on the soil parameters in the inverse modelling study.

<sup>1</sup>These storativity values are derived from pumping tests

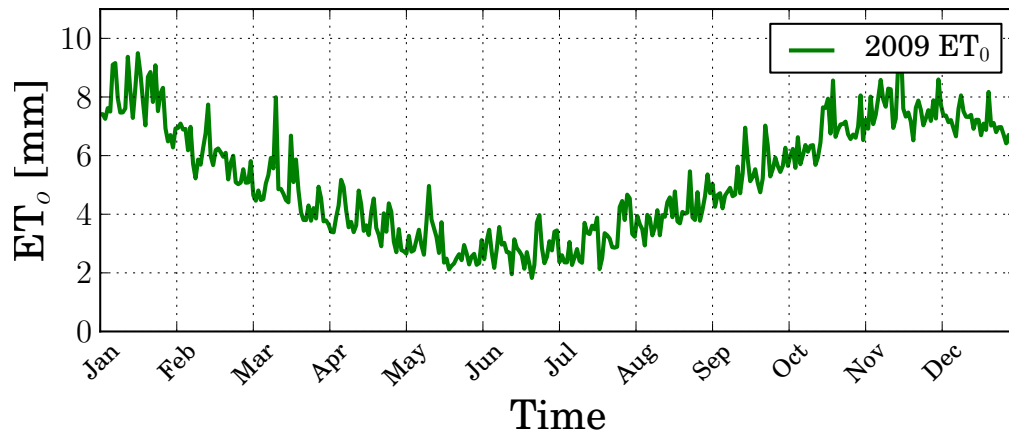


Figure 4.5: Potential evapotranspiration calculated with the Penman-Monteith method

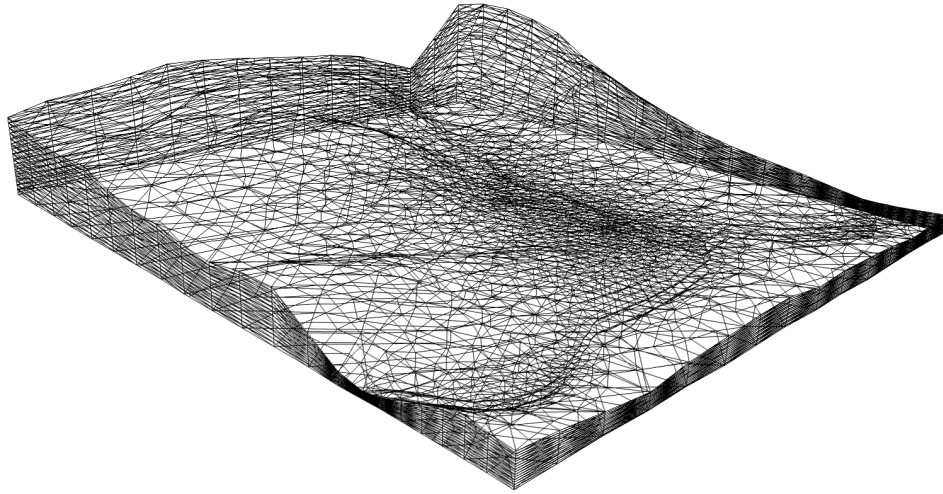
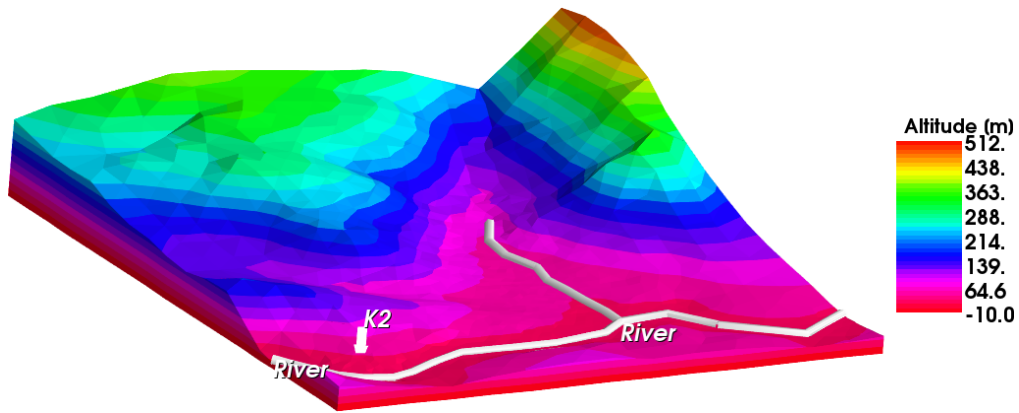
#### 4.1.4 Model construction

A Digital Elevation Model (DEM) with 20m resolution was obtained from the Geology Department at Stellenbosch University. A 2D mesh was created of the region shown in Fig. 4.1. The elevation at each point on the 2D mesh was then evaluated using the software Quantum GIS (QGIS Development Team, 2009) to determine the elevation from the DEM raster. A mesh of the 3D site could then be created using the software FreeFem++. The mesh is shown in Fig. 4.6a, with elevation and the river locations shown in Fig. 4.6b. The river is used as a Dirichlet boundary value at the surface, with a pressure head value of 0 m.

A total of 15 horizontal layers, with a total of  $\approx 20000$  nodes are used to create the phreatic model. The high number of layers allows us to approximate the delayed response of infiltration reaching the water table. Typically, only one layer is used (Arlen W. Harbaugh (2005)), but since the water table at this site drops to significant depths, a delay in the water table response to rainfall needs to be captured. This delay will be better approximated with the Richards equation, and is being considered for further research on the site.

#### 4.1.5 Numerical modelling

This section presents the equation for saturated flow, and the subsequent creation of a Reduced Order Model (ROM) following the Galerkin POD method. In this example, the Finite Element Method (FEM) is used to run high-fidelity simulations. The implementation of the equations and assumptions presented in this section were done using FreeFem++ (Hecht (2012)), an open-source Finite Element software. The site is approximated as a saturated zone, and so the equation for saturated groundwater flow based on Darcy's law (section 2.1.4) is used as


 (a) Numerical model mesh, containing  $\approx 20000$  vertices


(b) Altitude and boundary conditions

Figure 4.6: Domain of numerical model showing the mesh and boundary conditions

follows:

$$S_s \frac{\partial h}{\partial t} = \frac{\partial}{\partial x} K_x \frac{\partial h}{\partial x} + \frac{\partial}{\partial y} K_y \frac{\partial h}{\partial y} + \frac{\partial}{\partial z} K_z \frac{\partial (h+z)}{\partial z} \quad (4.2)$$

$$h(\mathbf{x}, 0) = h_{init}$$

$$h(\mathbf{x}, t) = f_d(t) \quad \text{on } \Gamma_d \times [t_0, t_{final}] \quad (4.3)$$

$$q_n(\mathbf{x}, t) = I(t) - ET_c(t), \quad \text{on } \Gamma_n \times [t_0, t_{final}]$$

The initial condition  $h_{init}$  is found by solving the model at steady state with the average yearly infiltration, and consequently augmenting the steady state

response to match the initial height at well  $K_2$ . Infiltration, denoted as  $I$ , is simply calculated as:

$$I(t) = \begin{cases} P(t) & \text{if } h < 0 \\ 0 & \text{if } h \geq 0 \end{cases} \quad (4.4)$$

where  $P$  is the precipitation shown in Fig. 4.2a, and the upper surface of the entire boundary is considered as a Neumann boundary, effectively adding water to the system by precipitation, and removing water by evapotranspiration. The river is used as a Dirichlet boundary with a pressure head of 0 m.

The application of the Galerkin method (presented in section 2.2.1) to linearise Eq. 4.2 yields a system of ordinary differential equations as follows:

$$\mathbf{M}^s \frac{d\mathbf{h}}{dt} + \mathbf{M}^p \mathbf{h} = \mathbf{V} \quad (4.5)$$

where  $\mathbf{M}^s$  is the  $n \times n$  stiffness matrix dependent on the storativity,  $\mathbf{M}^p$  is the  $n \times n$  mass matrix dependent on the conductivity vector  $\mathbf{K} = \{K_x \mathbf{i}, K_y \mathbf{j}, K_z \mathbf{k}\}$  and  $\mathbf{V}$  is the vector derived from the discretisation and contains the boundary information. Assuming constant conductivity and specific storage, the terms in the matrices Eq. 4.5 are calculated as follows:

$$\mathbf{M}^s_{ij} = S_s \int_{\Omega} v_i v_j \quad (4.6)$$

$$\mathbf{M}^p_{ij} = K_x \int_{\Omega} \frac{\partial v_i}{\partial x} \frac{\partial v_j}{\partial x} + K_y \int_{\Omega} \frac{\partial v_i}{\partial y} \frac{\partial v_j}{\partial y} + K_z \int_{\Omega} \frac{\partial v_i}{\partial z} \frac{\partial v_j}{\partial z} \quad (4.7)$$

and the vector terms, including Neumann boundary conditions, are calculated similarly to Eq. 2.53:

$$\mathbf{V}_i = \int_{\Omega} K_z \frac{\partial v_i}{\partial z} + \int_{\Gamma_n} q_n(\mathbf{x}, t) v_i \quad (4.8)$$

The subsequent application of Dirichlet boundary conditions is explained in section 4.5. This formulation yields 6 unknown parameters to be approximated in the inverse modelling study, namely  $K_x$ ,  $K_y$ ,  $K_z$  and  $S_s$  and the 2 evapotranspiration parameters. These parameters become the subject of the inverse modelling study. To reduce the time per simulation, a ROM is created to accelerate the inverse modelling study. The ROM development is briefly described in the following section.

#### 4.1.6 Application of Galerkin POD approach to Darcy flow

Following the ROM approach presented for the Richards equation (section 3.2.2), we derive the Galerkin POD model for a saturated flow problem, defined as a linear problem, by projecting the response onto an optimal set of global shape functions  $\mathbf{U}^h = \{\Psi_1, \dots, \Psi_{m^h}\} \in \mathbb{R}^{n \times m}$ . The response is represented by:

$$\mathbf{h} = \bar{\mathbf{h}} + \mathbf{U}^h \boldsymbol{\alpha} \quad (4.9)$$



where  $\bar{\mathbf{h}}$  is the mean of the snapshot set, and  $\alpha$  is the vector of weight values (section 3.2). The test function is chosen as  $w = \Psi_i$  for  $i = 1, \dots, m^h$ , where  $m^h$  is the number of global shape functions chosen. Substituting Eq. 4.9 into Eq. 4.5 we obtain:

$$\mathbf{U}' \left( \mathbf{M}^s \frac{d(\mathbf{U}^h \alpha)}{dt} + \mathbf{M}^p(\mathbf{U}^h \alpha) \right) = \mathbf{U}^{h'}(\mathbf{V} - \mathbf{M}^p \bar{\mathbf{h}}) \quad (4.10)$$

Finally, we represent the reduced matrices and vectors with a hat as follows:

$$\widehat{\mathbf{M}}^s \frac{d\alpha}{dt} + \widehat{\mathbf{M}}^p \alpha = \widehat{\mathbf{V}} \quad (4.11)$$

For the time discretisation, the implicit Euler technique is used (section 2.2.4). Given that the parameters are constant, matrices do not need to be re-evaluated at each time step. This applies to full and reduced matrices. As a result, the majority of the computational expense for a full model are the matrix inversion operations. By using the ROM of size  $m^h \times m^h$  in Eq. 4.11, the cost of inverting the matrices becomes negligible, leading to the possibility of speed-up times of  $\approx 1000$  reported by Siade *et al.* (2010).

### 4.1.7 Optimisation definition

In this section, an inverse modelling study is undertaken to calibrate the parameters in the numerical model so that the measured response at well  $K_2$  approximates the response in the numerical model at  $K_2$ . The optimal parameters are considered to be the ones that minimise the error between the two models. The objective function is defined as the integral of the difference between the measured and calculated response as follows:

$$F_{err}(\mathbf{p}) = \left[ \int_0^{t_{final}} \frac{(K_2 - \hat{K}_2(\mathbf{p}))^2}{t_{final}} dt \right]^{0.5} \quad (4.12)$$

where  $t_{final}$  is the total time period over which the simulation is run,  $K_2$  represents the measured well response over a year and  $\hat{K}_2(\mathbf{p})$  represents the parameterised ROM response over the same period at the  $K_2$  well location. Finally, the optimisation problem to calibrate the numerical response to the measured response is defined as:

$\begin{aligned} \text{Min}_{\mathbf{p}} : & \quad F_{err}(\mathbf{p}) \\ \text{Subject to :} & \quad \mathbf{p}^l \leq \mathbf{p} \leq \mathbf{p}^u \end{aligned}$	(4.13)
---	--------

where  $\mathbf{p}^u$  and  $\mathbf{p}^l$  are the upper and lower parameter boundaries of the optimisation problem, as shown in Table 4.1. The evapotranspiration values are defined between 0 and 2 to allow a wide range of movement in the parameter space. These evapotranspiration limits are based on those given for different vegetation types in the FAO (Allen (2000)), while the soil parameter limits are based on those found in the literature (section 4.1.3).

Table 4.1: The bounds of the design variables

	$K_x$ [m/d]	$K_y$ [m/d]	$K_z$ [m/d]	$S_s$	$K_c$	$K_e$
$\mathbf{p}^l$	0.01	0.01	0.01	0.0001	0	0
$\mathbf{p}^u$	0.1	0.1	0.1	0.01	2	2

#### 4.1.8 Optimisation using a POD surrogate model

In this example the space filling Optimum Latin Hypercube (OLE) Design of Experiments (DOE) approach is used (appendix A.4) to generate training and verification data sets. These two parameter sets are generated, using the software VisualDoc (Vanderplaats (2001)), over the parameter space (defined in Table 4.1) using the OLE. A total of 16 validation points and 5 training points are generated across the design space. The motivation for such a low number of training points is to determine whether the system response captured in the subsequent snapshot collection would be able to represent the system response over the entire parameter domain. A higher number of validation points are used to determine whether the subsequent ROM can be considered accurate over the entire domain, effectively evaluating the interpolation/extrapolation qualities inherent to the ROM. A high-fidelity simulation is run at each design point, and the resulting head pressure at each time step is stored. If a snapshot is taken at each day, and each simulation yields a response over 365 days, a total of 5 training points generates  $5 \times 365 = 1825$  potential snapshots.

In creating the global shape functions, no special selection criteria was used, and all 1825 snapshots were used. Subsequently, the 16 validation points were calculated using the ROM. The RRMS error between the full and ROM response is calculated at all 16 validation point responses, and the resulting maximum, mean and minimum RRMS values are shown in Fig. 4.7. It is clear that the 5 training points create a model that is accurate over the entire design space, with reasonable accuracy at all validation points. Furthermore, it is found that the time per ROM simulations is  $\approx 0.095s$ , and shows insignificant changes in time with an increase in the number of global shape functions. The full simulations require  $\approx 56s$ , and so the ROM runtime represents a speed-up of  $>500$ , with excellent interpolation characteristics. Since the ROM is accurate beyond the range of its initial training points, the ROM can be said to have good extrapolation qualities, at least within the bounds of the parameter space where the ROM accuracy is verified.

The first four eigenvectors used in the ROM are shown in Fig 4.8. It is interesting to note that there is variation in the horizontal and vertical direction. The horizontal direction variation can be partially attributed to the Neumann boundary being higher in areas of higher water availability, such as the valleys. The vertical variation is captured because of the multiple layers in the mesh, and allows an approximation of the time delay of water reaching the water table that would be seen with the Richards equation.

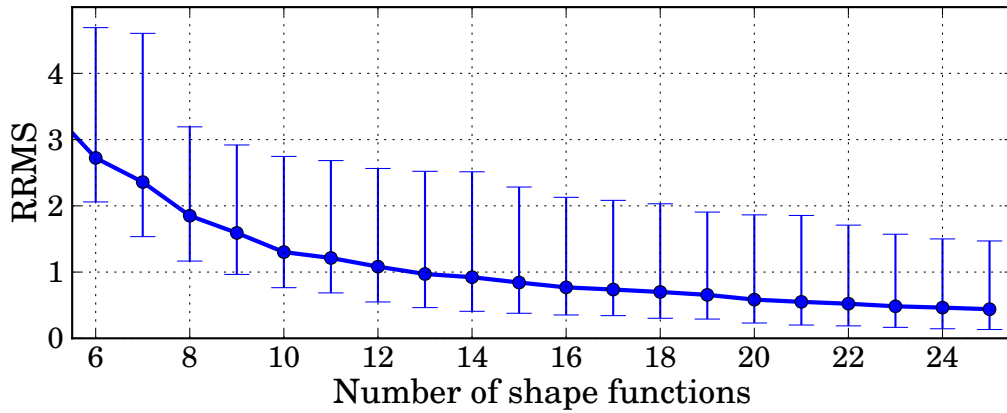


Figure 4.7: Error graph showing the mean, maximum and minimum RRMS errors yielded on the validation set, when using the POD model created with the training parameter set

A high number of local minima were found to be present in the response over the parameter space. To overcome this problem, the optimiser was started at each validation point and the subsequent optimum was stored. The optimal parameter set for 2008 is chosen from the optimum with the lowest RRMS, and the optimal parameter set  $\mathbf{p}_{2008}^*$  is used as the starting point to find the optimum parameter set  $\mathbf{p}_{2009}^*$  for 2009.

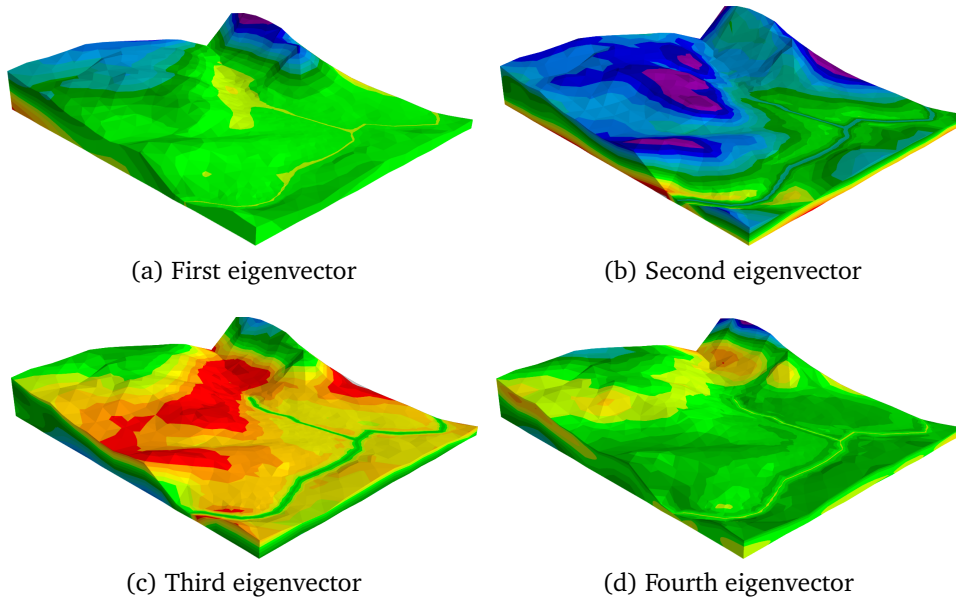


Figure 4.8: Eigenvectors showing spatial variation of head pressure at the Kogelberg site

### 4.1.9 Results of Kogelberg case study

The optimised parameters are presented in Table 4.2. The values lie within the parameter limits, indicating that the parameter space did not restrict the results. The resulting crop factors are found to be similar to those presented by Xu *et al.* (2009). These parameter values are therefore considered to be acceptable.

Table 4.2: Results of parameter optimisation

	$K_x$ [m/d]	$K_y$ [m/d]	$K_z$ [m/d]	$S_s$	$K_c$	$K_e$
$\mathbf{p}_{2008}^*$	0.027	0.0173	0.018	0.0009	0.59	1.44
$\mathbf{p}_{2009}^*$	0.027	0.0179	0.0194	0.0008	0.35	1.41

The time per high-fidelity simulation over a year period using 1 day time steps is  $\approx 36$  s. The time required per ROM simulation ( $t_{ROM}$ ) is presented in Table 4.3. The time per ROM simulation is given here with the overhead costs (i.e. loading all matrices and climate data), and the total time per optimisation is given as  $t_{opt}$ . The ROM uses a total of  $m^h = 40$  eigenvectors. Using the SQP optimiser in DOT (Vanderplaats (2001)), a total of 66 function calls are required to find the chosen optimum. The low number of function evaluations is due to the optimiser terminating in one of the local optima near the starting point.

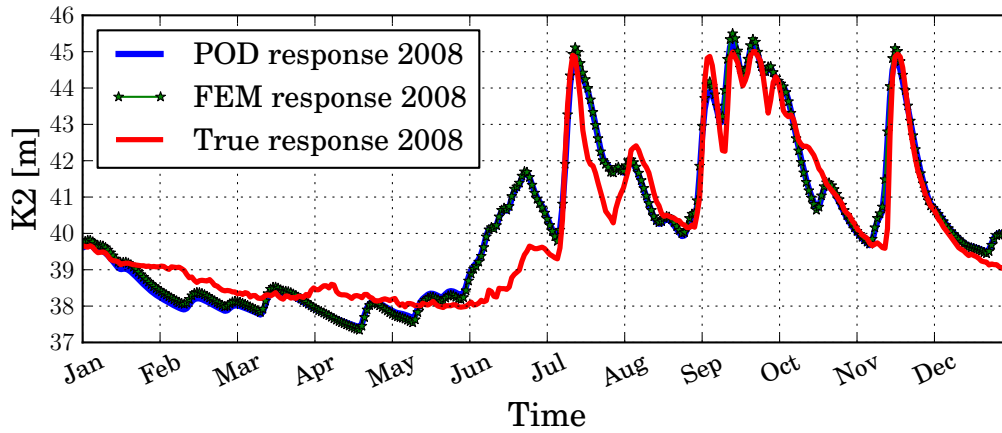
Table 4.3: Time and error results

	$t_{HF}$ [s]	$t_{ROM}$ [s]	$n_{fc}$	$t_{opt}$	$F_{err}(\mathbf{p})$
Optimisation of $\mathbf{p}_{2008}^*$ :	56	0.095	66	9.8	0.83
Optimisation of $\mathbf{p}_{2009}^*$ :	56	0.095	57	8.9	0.85

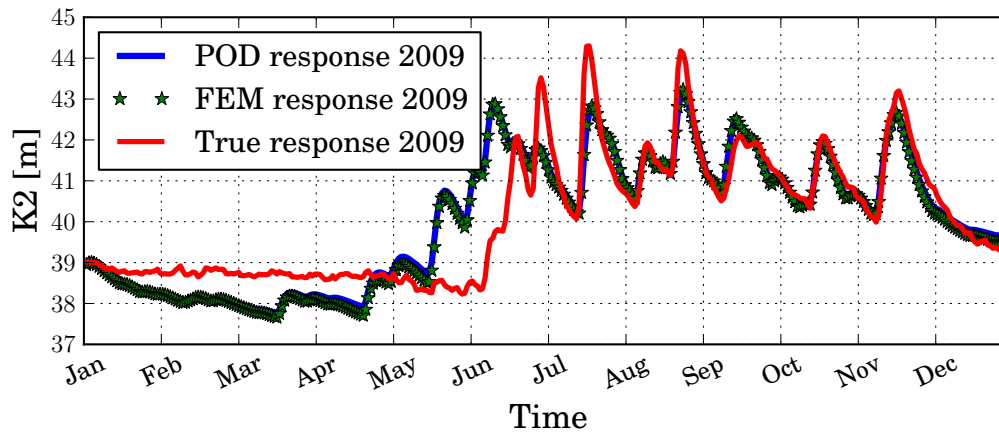
After starting the optimisation at each of the 16 validation points, the resulting optimum with the smallest error is chosen. In Fig.'s 4.9a and 4.9b the numerical responses obtained from the ROM for 2008 and 2009 at well  $K_2$  are presented. Once satisfactory parameters were found using the ROM, a full simulation is done at the same optimal parameter sets. The high-fidelity and ROM models are in excellent agreement, while the error between the ROM and the measured response is considered acceptable.

During the first half of each year the measured response shows little or no response to precipitation, while a response to rainfall events is evident in the numerical responses. In both years, the error at the beginning of the rainy season is significant due to the numerical model not being able to capture the saturating of the unsaturated zone. However, the ROM is highly accurate in the months August to December, during which time the soil is effectively saturated.

With the optimised results, the recharge can be calculated. In Table 4.4 the total rainfall and total evapotranspiration is presented. In the optimisation study,



(a) Optimised response for 2008, with the response in m.a.s.l.



(b) Optimised response for 2009, with the response in m.a.s.l.

Figure 4.9: Resulting response at well  $K_2$  after inverse estimation study

recharge estimates varied between 0 – 5%. In the response chosen, the recharge is 4.5% and 3.9% for the two years considered. These estimates correspond to the low end of estimates for recharge in the Peninsula layer in the TMG group in Jia (2007)<sup>2</sup>.

Table 4.4: Recharge results

	Rain [mm]	$ET_{total}$ [mm]	Infiltration [mm]	Recharge %
2008	1525	1458	66.2	4.5
2009	1032	994	38.9	3.9

The daily true evapotranspiration is plotted against the potential evapotranspiration over the two years in Fig. 4.10. The two responses do not seem to be

<sup>2</sup>Estimates are given in Table 1.1. No estimates are given for the Oudebosch site

related, due partly to the abundance of water when  $ET_0$  is low.

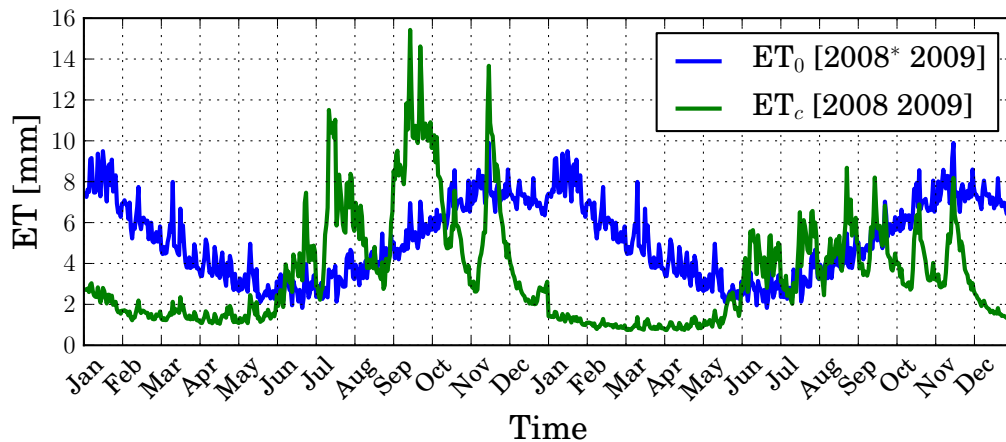


Figure 4.10: Evapotranspiration response for 2008 and 2009, where **2008\*** indicates that the  $ET_0$  for 2008 is assumed to be the same as the  $ET_0$  of 2009, due to data lacking in 2008

The accuracy in the numeric model response is limited by approximating the unsaturated zone as a phreatic saturated zone. The use of the Richards equation could improve the model accuracy. The model accuracy is also dependent on a number of parameters that were not optimised. These factors include the values defining the stress factors RAW, TAW, REW and TEW. Furthermore, the crop factor can be set to vary monthly to imitate growing crops.

Lastly, since the ROM is valid over the entire design space, alternative optimisation techniques could be investigated such as particle swarm. This could help to overcome the problem of multiple local minima.

#### 4.1.10 Conclusion

In this section a 3D numerical model of the Oudebosch site was created, and an inverse modelling study was undertaken, using a ROM created with the GPOD method. The optimisation methodology was explained and some optimal responses are shown.

The evaporation response was found to be highly dependent on the availability of water. The annual recharge is the difference between annual rainfall and annual evapotranspiration, yielding recharge estimates between 3 and 5% of the annual rainfall.

The ROM model was found to be  $> 500$  times faster than the original model, which is in the same order of magnitude as speed-up times reported in the literature. With the overhead costs of the simulation included, optimisation of the system requires  $\approx 10$  s.

This study serves as a step towards the creation of a model that is calibrated at all measuring wells. Such a model could consider different soil and rock layers and different vegetation cover at the different wells, and so yield recharge estimates that are valid over the entire Oudebosch region.

## 4.2 Synthetic unsaturated case study

A hypothetical numerical example in 2D is considered to demonstrate the application of the GPOD approach on a synthetic unsaturated case study. The problem considers combined saturated-unsaturated flow using the Richards equation in Eq. 2.28, and the van Genuchten relations described in Table 2.1. The model is a vertical section through an unconfined aquifer, and has Neumann and Dirichlet boundary types applied at the boundaries shown in Fig. 4.11. This example has a transient response, subject to initial and boundary conditions specified in Eq. 4.14. The example is simulated using a high-fidelity (or full FE) simulation, after which the solution is ‘approximated’ using the GPOD approach developed in this chapter. It is shown that the GPOD approach does not yield feasible speed-up times when applied to the non-linear Richards equation, and an adaptation to the GPOD method is proposed in chapter 5 to improve speed-up times.

### 4.2.1 Problem definition

The problem, with the non-linear function’s parameters, is presented mathematically as:

$$\begin{aligned}
 C(h) \frac{\partial h}{\partial t} &= \nabla \cdot [K(h) \nabla (h + z)] \\
 h(x, z, 0) &= h_{init} \\
 f_n(x, z, t) &= -1 \text{ cm/hr on } \Gamma_{w1} \\
 f_n(x, z, t) &= -1 \text{ cm/hr on } \Gamma_{w2} \\
 f_n(x, z, t) &= q_{rain}(t) \cdot \mathbf{n} \text{ cm/hr on } \Gamma_r \\
 h(x, z, t) &= (180 - z) \text{ cm on } \Gamma_{d1}, \Gamma_{d2} \\
 t &\in [0, 500 \text{ hr}]
 \end{aligned} \tag{4.14}$$

where the vertical direction is denoted by  $z$ . The Neumann boundary conditions are labelled  $f_n(x, z, t) = -K(h) \nabla (h + z) \cdot \mathbf{n}$ . Two of the three Neumann boundaries exist as round sinks on boundaries  $\Gamma_{w1}$  and  $\Gamma_{w2}$ . These sink boundaries are circular with a radius of 10 cm. The third Neumann boundary occurs on boundary  $\Gamma_r$ , simulating recharge and evapotranspiration. The soil parameters are given as:

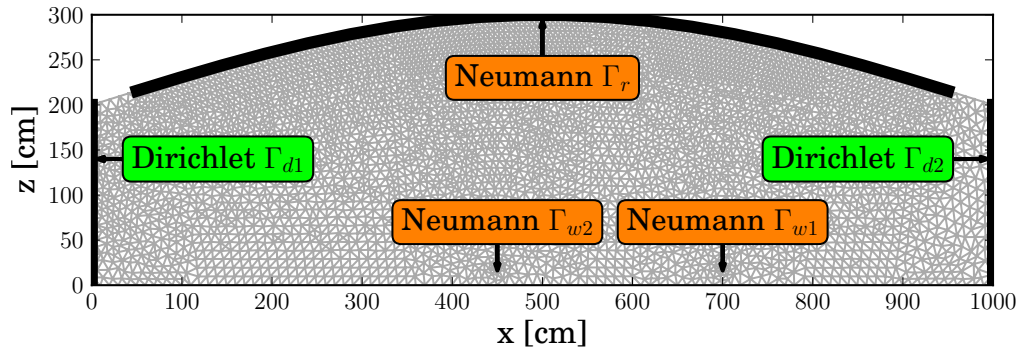


Figure 4.11: Numerical case study mesh with  $\approx 5000$  node points. The parabolic upper surface represents the soil topography

$$K_s = 0.35 \text{ cm/hr}$$

$$m = 0.4$$

$$h_b = 20 \text{ cm} \tag{4.15}$$

$$\varepsilon_s = 0.4$$

$$\varepsilon_r = 0$$

The high-fidelity (or full FE) simulation is executed with FreeFem++ (Hecht (2012)), using hourly time-steps and a convergence criteria of  $10^{-3}$  using the Picard iteration method, described in section 2.2.4. The model has  $\approx 5000$  nodes, and the simulation requires  $\approx 140$  s on a 2.8GHz processor, yielding a pressure head response at each FE node for 100 consecutive time steps.

The initial pressure head distribution is shown in Fig. 4.12a and the final pressure head distribution is shown in Fig. 4.12b. The effect of the two Neumann boundaries that are extracting water is evident, with the water table dropping to 80 cm.

The response from the simulation is stored as  $\mathbf{H} = \{\mathbf{h}_1, \dots, \mathbf{h}_{100}\}$ . The stored responses serves as the snapshot set, and no special snapshot selection is done. After doing a PCA on the entire snapshot set, a set of eigenvectors are obtained, three of which are shown in Fig. 4.13. The full model is approximated using a chosen number  $m^h$  of eigenvectors to recreate the problem using the Galerkin POD approach.

## 4.2.2 Time and accuracy results

After creating a ROM using the GPOD method, the clock time required by the ROM to replicate the original simulation is obtained. The time per simulation is



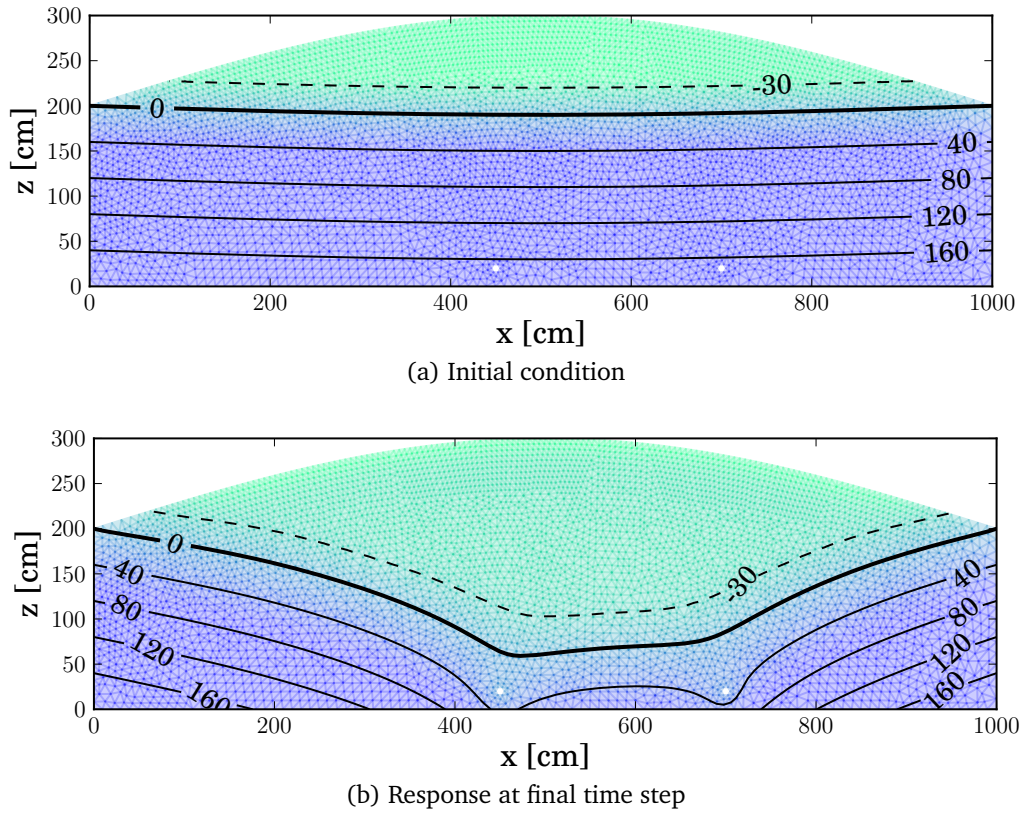


Figure 4.12: Initial and final head pressure given in centimetres

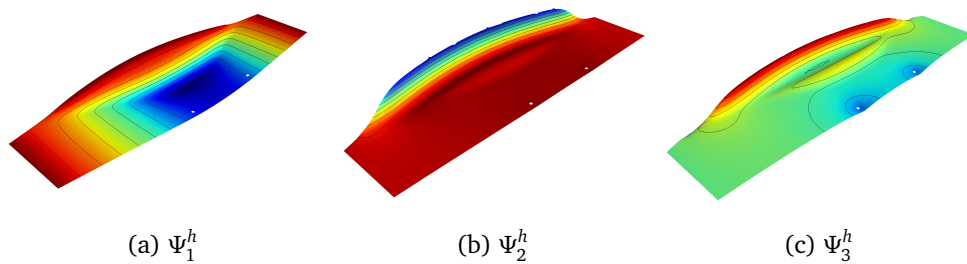


Figure 4.13: The first three pressure head eigenvectors derived from the snapshot set, showing the principal variations in the system occurring around the Neumann boundaries. The figures correspond to the problem in Fig. 4.12, and are plotted at an angle to view the variation.

plotted in Fig. 4.14, showing an increase in time with an increase in the number of shape functions. The error also decreases with an increase in shape functions, up to a RRMS < 1% at  $m = 12$ . However, the simulation using the ROM requires more time than the full model when using  $m \geq 7$ , indicating that the GPOD

approach is not feasible for application to the Richards equation.

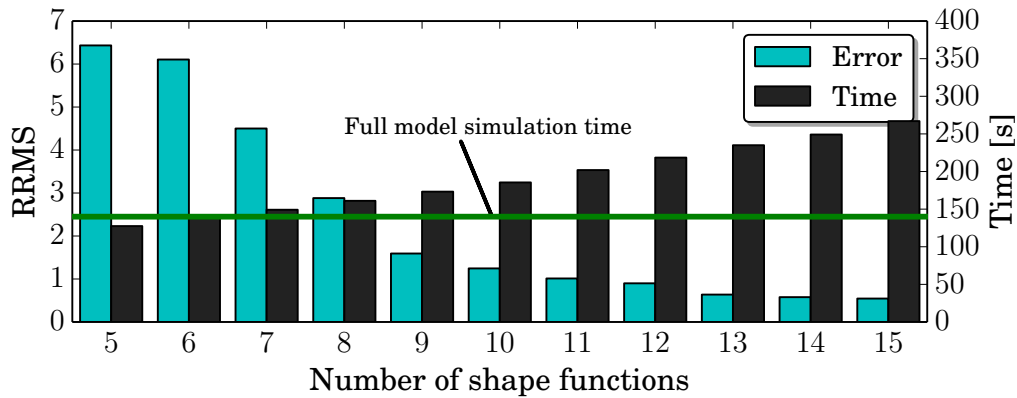


Figure 4.14: Time and error vs. the number of shape function per ROM evaluation using the GPOD approach, where RRMS is the relative root mean squared error (Eq. 3.40). The original simulation requires 140 s.

In creating the ROM, the emphasis shifts from inverting the full matrix to the creation of a reduced matrix. In this case, the shift does not result in a reduction in the computational time per simulation.

To better understand the computational expense required, the time for different steps of the two models are analysed. In Fig. 4.15 the time partitioning is shown for the different segments of the full and reduced model simulations, taken from the example. In the pie charts, the matrix reduction  $\mathbf{U}'\mathbf{M}\mathbf{U}$  is seen to take more time than the inversion of  $\mathbf{M}$  when using only 10 shape functions. However, the computational expense of calculating  $\boldsymbol{\alpha} = \hat{\mathbf{M}}^{-1}\hat{\mathbf{V}}$  is seen to take an insignificant time when compared to the entire simulation.

### 4.3 Conclusion

The application of the GPOD approach is found to be very effective when applied to the linear saturated flow equations in transient simulations. While the GPOD method is not tested on steady state simulations, similar results can be expected. The good interpolative qualities of the GPOD method are highlighted and demonstrated on the complex transient case study of the Kogelberg site.

In contrast, the GPOD approach does not yield promising results when applied to the non-linear Richards equation. The lack of efficiency can be explained when looking at the complexity of the different parts of the simulation. The inversion of the sparse finite element matrices is of complexity  $O(n^{3/2})$  per iteration (Pechstein (2012)), where  $n$  is the number of nodes in the numerical model. In contrast, the matrix reduction ( $\mathbf{U}'\mathbf{M}\mathbf{U}$ ) at each time step requires nearly full matrix operations, with a complexity of  $O(mn^2 + nm^2)$ , where  $m$  is the number of global shape functions, resulting in no time gain in non-linear problems. In comparison, the

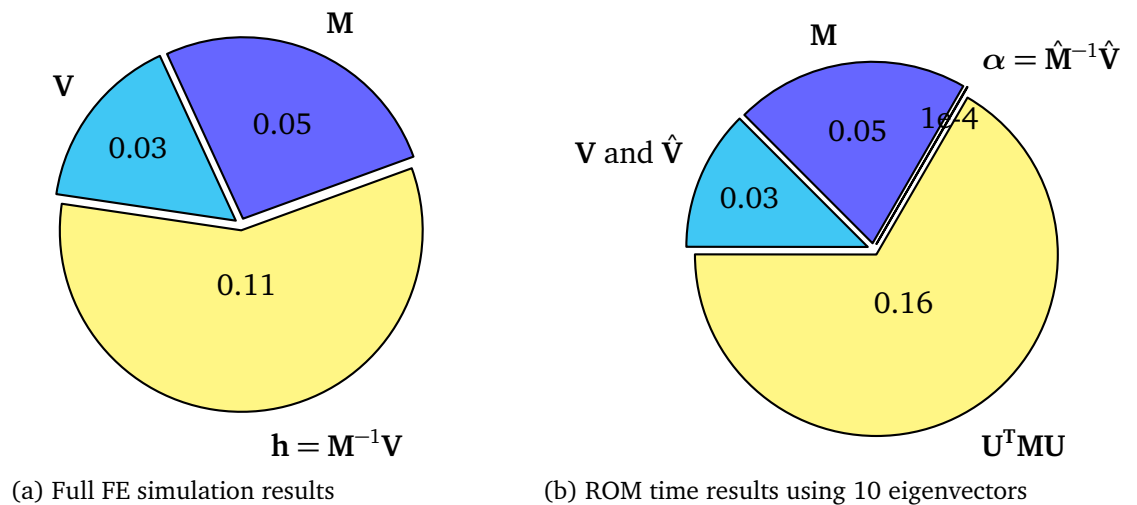


Figure 4.15: Pie charts showing time, in seconds, required per function evaluation using 10 shape functions. The computational expense shifts from inverting the matrix to creating and reducing the matrix when creating the ROM.

GPOD ROM applied to linear problems has a complexity of  $O(m^2)$ , resulting in significant speed-up times (Siade *et al.* (2010)). In the following section, we look at two alternative approaches to creating ROM's that reduce the computational complexity significantly.

## Chapter 5

# Improving CPU performance using the Hybrid and Linearised-POD methods

**I**N the previous chapter, the Galerkin Proper Orthogonal Decomposition (GPOD) approach is found to be infeasible in terms of time for non-linear problems, due to the computational complexity of creating and reducing the matrices at each time step. In this section, we consider alternative approaches of creating a POD-based Reduced Order Model (ROM). The alternative approaches that are proposed and developed in this chapter build on the idea for a Hybrid Reduced Order Model (ROM), that is described in the unpublished paper by De Vuyst (2009). The Hybrid method is so called because of the use of both linearisation and interpolation in the ROM construction. In the paper by De Vuyst (2009), the Hybrid approach is suggested as a method for non-linear stationary problems, and it is not applied on a case study.

In this chapter, the method is developed for the use of transient non-linear problems, a detailed expansion of the equations is given, boundary values are treated and the method is applied to a case study. When applied to a 2D test problem in this research, the Hybrid method is shown to yield good speed-up times ( $\approx 50$ ), where the response is based on the Richards' equation. However, some limitations were identified in the application of the Hybrid method when applied to a transient problem, such as the decoupling of the non-linear functions from the pressure head response, and the increasing complexity of an interpolative metamodel in a high-dimensional parameter space, also known as the curse of dimensionality.

The limitations, coupled with the goal of avoiding interpolative techniques, led to the development of the Linearised GPOD (LGP) approach in this research. The LGP is so named because of the way the non-linear terms are linearised during the execution of the ROM. The LGP approach yields similar speed-up times to the Hybrid approach when using a small reduced basis set when applied to the 2D test problem. However, the LGP displays better speed-up times with a

larger reduced basis set. The Hybrid and LGP methodologies are explained in this chapter and compared on the 2D example of section 4.2.

## 5.1 Hybrid approach to treatment of non-linear terms

In the paper by De Vuyst (2009), a hybrid method is proposed to approximate non-linear stationary problems. The method proposes that the matrices containing the non-linear functions,  $C$  and  $K$ , be calculated by the use of a metamodel, such as kriging or radial basis functions. In this section, the method is explained briefly and adapted to the Richards' equation.

It can be recalled that in the formulation used by the GPOD method, the evaluation of full integral matrices is required at each iteration. The Hybrid method proposes a method of eliminating the evaluation of the full matrices required by the GPOD approach, as well as the subsequent reduction. This is done by linearising the matrices, and recreating them with the aid of a metamodel. In this section this approach is explained in detail.

### 5.1.1 Linearising the non-linear terms

In the non-linear Richards' equation, the mass and stiffness matrices are a function of the non-linear functions describing moisture content and conductivity. In order to linearise the matrices, we first linearise these non-linear functions. A representative snapshot set for the pressure head is generated in the same way described in chapter 3, where the snapshot set is chosen as:

$$\mathbf{H} = \{\mathbf{h}^*(\mathbf{p}_1), \mathbf{h}^*(\mathbf{p}_2), \dots, \mathbf{h}^*(\mathbf{p}_d)\}, \mathbf{H} \in \mathbb{R}^{n \times d} \quad (5.1)$$

where  $d$  represents the number of snapshots chosen,  $\mathbf{p}$  represents the parameters on which each pressure head response ( $h$ ) depends, and each snapshot  $\mathbf{h}_i^*$ ,  $i = 1, \dots, d$  is 0 on the Dirichlet, or exterior, nodes. In the case of the Hybrid method, snapshots of each of the non-linear functions are also collected from the full simulation:

$$\mathbf{C} = \{C(\mathbf{h}(\mathbf{p}_1)), \dots, C(\mathbf{h}(\mathbf{p}_d))\}, \mathbf{C} \in \mathbb{R}^{n \times d} \quad (5.2)$$

and

$$\mathbf{K} = \{K(\mathbf{h}(\mathbf{p}_1)), \dots, K(\mathbf{h}(\mathbf{p}_d))\}, \mathbf{K} \in \mathbb{R}^{n \times d} \quad (5.3)$$

Since Dirichlet boundary conditions are not enforced directly on the non-linear functions, the snapshot set  $\mathbf{C}$  and  $\mathbf{K}$  do not have boundary, or exterior, nodes with values set to zero in the snapshot set, in contrast to the pressure head snapshot collection  $\mathbf{H}$ . A more detailed discussion on the selection and use of the parameters used in  $\mathbf{p}$  is given in section 5.1.2.

With the snapshot sets, the gravity centres of the snapshot sets,  $\mathbf{G}^C$  and  $\mathbf{G}^K$ , are found and the principal components of the snapshot sets are found with a PCA.

As in the linearisation of the pressure head function  $h$ , we have  $e$  Global Shape Functions (GSF) available with which to approximate  $C$  and  $K$ . We choose the number of GSF's for the  $K$  and  $C$  terms as  $m^K \leq e$  and  $m^C \leq e$  respectively, and the principal components of the snapshot sets are written as  $\mathbf{U}^K = \{\Psi_1^K, \dots, \Psi_{m^K}^K\}$  and  $\mathbf{U}^C = \{\Psi_1^C, \dots, \Psi_{m^C}^C\}$ . The snapshot sets for  $C$  and  $K$  are shown in Fig.'s 5.1a and 5.1b, demonstrating again how each snapshot set is projected onto a plane that is represented by the principal components of the snapshot set.

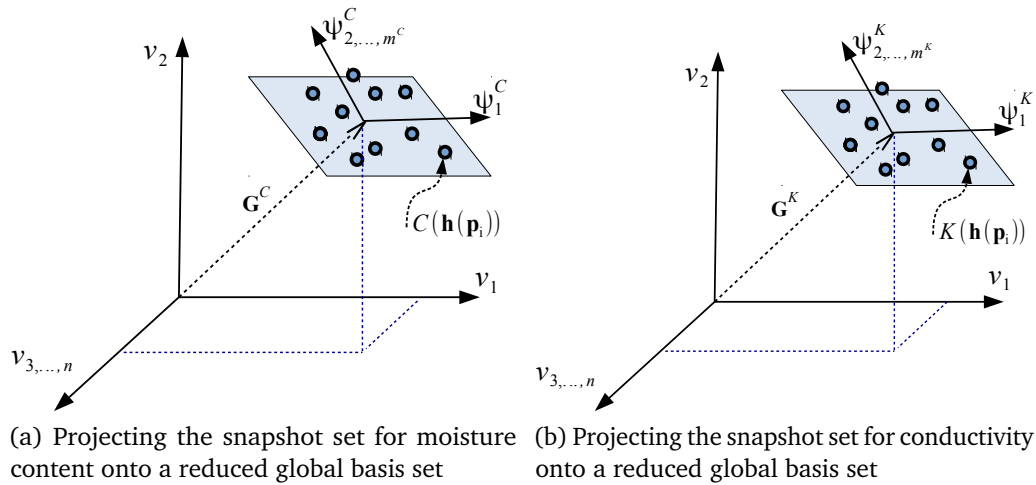


Figure 5.1: Projecting snapshots onto a reduced plane

As with the linearisation of the pressure head, the non-linear functions can also be approximated as a linear sum of the eigenvectors multiplied by weight values. A subtle difference between the way the true and approximated functions are calculated, is that each true function is evaluated as a response to a pressure head response, while each approximated function is evaluated as a response at a parameter point  $\mathbf{p}$ , by evaluating the weight values at the given parameter point, as follows:

$$C(h(\mathbf{p})) \approx \hat{C}(\mathbf{p}) = \sum_{i=1}^{m^C} \alpha_i^C(\mathbf{p}) \Psi_i^C + G^C \quad (5.4)$$

$$K(h(\mathbf{p})) \approx \hat{K}(\mathbf{p}) = \sum_{i=1}^{m^K} \alpha_i^K(\mathbf{p}) \Psi_i^K + G^K \quad (5.5)$$

where the unknowns are the weight functions  $\alpha_i^C(\mathbf{p})$ ,  $i = 1, \dots, m^K$  and  $\alpha_i^K(\mathbf{p})$ ,  $i = 1, \dots, m^K$ . The Hybrid method proposes that each weight function be calculated with a metamodel, as shown in Fig. 5.2. For each  $i^{\text{th}}$  global shape function, a separate metamodel needs to be created to evaluate the corresponding weight value. In the following section the use of metamodel in the Hybrid method is explained.

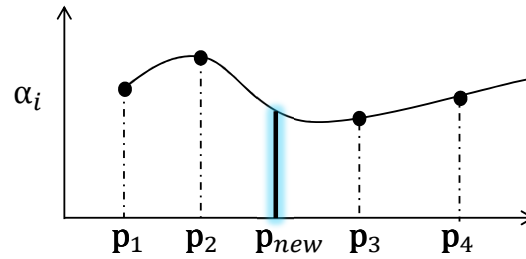


Figure 5.2: Interpolation of the  $i^{th}$  weight value between a series of training points, and the weight function approximation at a new parameter set  $\mathbf{p}_{new}$

### 5.1.2 Creating a metamodel for the non-linear terms

The Hybrid method uses a metamodel<sup>1</sup> to approximate the non-linear terms, while the pressure head response is approximated with a reduced basis set following the GPOD method in chapter 3. The combination of the two approaches leads to a hybrid approach, hence the name. A metamodel is constructed using a number of ‘true’ function responses at a specified design points, called a Design Of Experiments (DOE), in the parameter space. The parameter space in this research typically consists of the parameters used to describe the non-linear functions in Richards’ equation, as shown in Table 5.1. In this table, the parameters are shown as  $\mathbf{p}_i$ ,  $i = 1, \dots, ndoe$ , and  $ndoe$  is the number of design points in the DOE. Each  $\alpha_i^C$ ,  $i = 1, \dots, m^C$  function has a response over the parameter space, as does each  $\alpha_i^K$ ,  $i = 1, \dots, m^K$ , leading to the creation of  $(m^K + m^C)$  metamodels.

Table 5.1: Example of training inputs and outputs for the Hybrid method

	$\mathbf{t}$	$\mathbf{h}_b$	$\mathbf{K}_s$	$\boldsymbol{\varepsilon}_r$	$\boldsymbol{\varepsilon}_s$	$\mathbf{m}$	$\boldsymbol{\alpha}_i^K$	$\boldsymbol{\alpha}_i^C$
$\mathbf{p}_1$	$t_1$	$h_{b,1}$	$K_{s,1}$	$\varepsilon_{r,1}$	$\varepsilon_{s,1}$	$m_1$	$\alpha_{i,1}^K$	$\alpha_{i,1}^C$
$\mathbf{p}_2$	$t_2$	$h_{b,2}$	$K_{s,2}$	$\varepsilon_{r,2}$	$\varepsilon_{s,2}$	$m_2$	$\alpha_{i,2}^K$	$\alpha_{i,2}^C$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{p}_{ndoe}$	$t_{ndoe}$	$h_{b,ndoe}$	$K_{s,ndoe}$	$\varepsilon_{r,ndoe}$	$\varepsilon_{s,ndoe}$	$m_{ndoe}$	$\alpha_{i,ndoe}^K$	$\alpha_{i,ndoe}^C$

The creation of a suitable DOE is not discussed in the article by De Vuyst (2009), and the paper does not develop a Hybrid model for transient ROM’s either. We propose a simple adaptation to the method by adding time to the parameter space (Table 5.1), and using standard space-filling Latin-Hypercube (see Bates *et al.* (2004)) for the DOE. The addition of time in this manner presents a significant problem, because the calculation of  $C$  and  $K$  is now completely decoupled from the pressure head response  $h$ . This means that if the ROM response is inexact, the approximated non-linear functions will not correspond

<sup>1</sup>Indicating the use of interpolation with a technique like kriging

to the approximated pressure head. A solution to this problem is presented in section 5.2.

In order to create a metamodel to approximate the response function  $\alpha^C$  and  $\alpha^K$ , De Vuyst (2009) proposes the use of Radial Basis Functions (RBF's). Typically, in the use of a RBF metamodel, each training point  $\mathbf{p}_1$  is assigned a Radial Basis Function, as shown in Fig. 5.3a. A common radial basis function is the exponential Radial Basis kernel:

$$k(\mathbf{p}_1, \mathbf{p}_2) = \exp\left(\frac{-\|\mathbf{p}_1 - \mathbf{p}_2\|^2}{2\sigma^2}\right) \quad (5.6)$$

where  $\sigma$  is a scalar value that determines the radius of influence of the RBF. Each training point is assigned an amplitude (or weight function)  $A$ , so that the true response  $\alpha^f(\mathbf{p})$  at an unknown parameter  $\mathbf{p}$  can be approximated as:

$$\hat{\alpha}^f(\mathbf{p}) = \sum_{i=1}^{ndoe} A_i k(\mathbf{p}_i, \mathbf{p}) \quad (5.7)$$

The amplitude terms  $\mathbf{A} = [A_1, A_2, \dots, A_{ndoe}]$  can be calculated by solving the system:

$$\mathbf{A} = \mathbf{Q}^{-1}\mathbf{F} \quad (5.8)$$

where the matrix  $\mathbf{Q}$  and vector  $\mathbf{F}$  are represented as:

$$\mathbf{Q} = \begin{bmatrix} k(\mathbf{p}_1, \mathbf{p}_1) & k(\mathbf{p}_1, \mathbf{p}_2) & \dots & k(\mathbf{p}_1, \mathbf{p}_{ndoe}) \\ k(\mathbf{p}_2, \mathbf{p}_1) & k(\mathbf{p}_2, \mathbf{p}_2) & \dots & k(\mathbf{p}_2, \mathbf{p}_{ndoe}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{p}_{ndoe}, \mathbf{p}_1) & k(\mathbf{p}_{ndoe}, \mathbf{p}_2) & \dots & k(\mathbf{p}_{ndoe}, \mathbf{p}_{ndoe}) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{ndoe} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \alpha^f(\mathbf{p}_1) \\ \alpha^f(\mathbf{p}_2) \\ \vdots \\ \alpha^f(\mathbf{p}_{ndoe}) \end{bmatrix} \quad (5.9)$$

With the computed amplitudes, the true function can be approximated according to Eq. 5.7, as shown in Fig. 5.3b. Now using the RBF approach, it is possible to create metamodels for each  $\alpha_i^C$  and  $\alpha_i^K$ . However, before the metamodels can be created, the training points at each design point need to be generated. These points are generated by projecting the snapshot set for  $\mathbf{C}$  and  $\mathbf{K}$ , derived from a full simulation, onto their respective Reduced Basis Sets  $\mathbf{U}^C$ ,  $\mathbb{R}^{n \times m^C}$  and  $\mathbf{U}^K$ ,  $\mathbb{R}^{n \times m^K}$  respectively. The non-linear snapshot sets for the non-linear functions are represented as a function of a parameter point as follows:

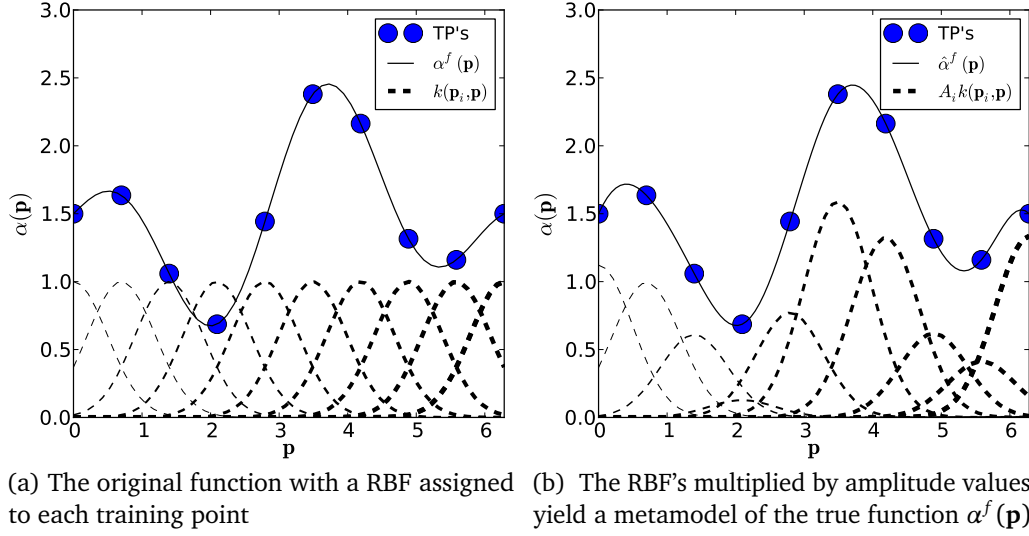
$$\mathbf{C} = \{C(h(\mathbf{p}_1)), \dots, C(h(\mathbf{p}_{ndoe}))\} \in \mathbb{R}^{n \times ndoe} \quad (5.10)$$

$$\mathbf{K} = \{K(h(\mathbf{p}_1)), \dots, K(h(\mathbf{p}_{ndoe}))\} \in \mathbb{R}^{n \times ndoe} \quad (5.11)$$

The training points for the metamodels are generated by projecting the normalised snapshot sets onto their respective RBS's as follows:

$$\alpha^C = \mathbf{U}^{C'} \bar{\mathbf{C}}, \quad \text{where } \alpha^C \in \mathbb{R}^{m^C \times ndoe} \quad (5.12)$$




 Figure 5.3: Approximating a weight function  $\alpha^f$  with Radial Basis Functions

$$\alpha^K = \mathbf{U}^{K'} \bar{\mathbf{K}}, \quad \text{where } \alpha^K \in \mathbb{R}^{m^K \times ndoe} \quad (5.13)$$

where each column  $\bar{\mathbf{C}}_i = \mathbf{C}_i - \mathbf{G}^C$ , and  $\bar{\mathbf{K}}_i = \mathbf{K}_i - \mathbf{G}^K$ , Now each  $i^{th}$  row of the  $\alpha^{K,C}$  matrices are used as training points used to generate the  $i^{th}$  weight value surface (see Table 5.1). Once the metamodels have been created, the weight values can be approximated at a new parameter point ( $\mathbf{p}_{new}$ ) at each ROM iteration, by searching the weight values from the metamodels as follows:

$$\alpha_i^K(\mathbf{p}_{new}) \approx \hat{\alpha}_i^K(\mathbf{p}_{new}), \quad i = 1, 2, \dots, m^K \quad (5.14)$$

$$\alpha_i^C(\mathbf{p}_{new}) \approx \hat{\alpha}_i^C(\mathbf{p}_{new}), \quad i = 1, 2, \dots, m^C \quad (5.15)$$

where  $\hat{\alpha}_i^K$  and  $\hat{\alpha}_i^C$  represent the metamodels approximating the weight values for the  $i^{th}$  weight function surface. The non-linear functions are now calculated using Eq.'s 5.4 and 5.5. Using the metamodels for non-linear functions, it is now possible to linearise the matrices.

### 5.1.3 Matrix reduction with the Hybrid approach

Using the approximations for the non-linear functions, the integral matrices used in the GPOD method (Eq. 3.16) can be linearised. As a reminder, we restate the

Richards' equation in its integral form:

$$\begin{aligned}
 \forall i, \sum_{j=1}^{m^h} \left( \int_{\Omega} C \Psi_i (\alpha_j^{\tau+1} \Psi_j) + \delta t \int_{\Omega} K \nabla \Psi_i \cdot \nabla (\alpha_j^{\tau+1} \Psi_j) \right) = \\
 \int_{\Omega} C \Psi_i \hat{h}^{\tau} + \delta t \int_{\Gamma_n} \Psi_i f_n \cdot \mathbf{n} - \delta t \int_{\Omega} K \frac{\partial \Psi_i}{\partial z} \\
 - \underbrace{\int_{\Omega} C \Psi_i G^h - \delta t \int_{\Omega} K \nabla \Psi_i \cdot \nabla G^h}_{\hat{\mathbf{V}}_i^g} \\
 - \underbrace{\int_{\Omega} C \Psi_i b^{h,\tau+1} - \delta t \int_{\Omega} K \nabla \Psi_i \cdot \nabla b^{h,\tau+1}}_{\hat{\mathbf{V}}_i^d}
 \end{aligned} \tag{5.16}$$

where the equation is represented as a system of ODE's in matrix form as:

$$\hat{\mathbf{M}}^s(h) \alpha^{\tau+1} + \delta t \hat{\mathbf{M}}^p(h) \alpha^{\tau+1} = \hat{\mathbf{V}}^s(h) + \delta t [\hat{\mathbf{V}}^n(f_n) - \hat{\mathbf{V}}^p(h)] - \hat{\mathbf{V}}^g(h) - \hat{\mathbf{V}}^d(h) \tag{5.17}$$

The full mass and stiffness matrices ( $\mathbf{M}^s$  and  $\mathbf{M}^p$ ), that are originally developed in section 2.2.3, can now be linearised. The mass matrix is written in terms of  $C$ , when using the local basis set  $v_i$ ,  $i = 1, \dots, n$  as follows:

$$\mathbf{M}^s(C) = \begin{bmatrix} \int_{\Omega} C v_1 v_1 & \dots & \int_{\Omega} C v_1 v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} C v_n v_1 & \dots & \int_{\Omega} C v_n v_n \end{bmatrix} \tag{5.18}$$

When using the reduced basis set, the full mass matrix is reduced to:

$$\hat{\mathbf{M}}^s(C) = \mathbf{U}^{h'} \mathbf{M}^s(C) \mathbf{U}^h \tag{5.19}$$

Similarly, the stiffness matrix is written in terms of  $K$  as follows:

$$\mathbf{M}^p(K) = \begin{bmatrix} \int_{\Omega} K \nabla v_1 \cdot \nabla v_1 & \dots & \int_{\Omega} K \nabla v_1 \cdot \nabla v_n \\ \vdots & \ddots & \vdots \\ \int_{\Omega} K \nabla v_n \cdot \nabla v_1 & \dots & \int_{\Omega} K \nabla v_n \cdot \nabla v_n \end{bmatrix} \tag{5.20}$$

and in its reduced form it is presented as:

$$\hat{\mathbf{M}}^p(K) = \mathbf{U}^{h'} \mathbf{M}^p(K) \mathbf{U}^h \tag{5.21}$$

Now, if we substitute the approximation for  $C$  from Eq. 5.4 into Eq. 5.18, the mass matrix can be calculated, by using the metamodels to calculate the weight values at an unknown parameter point  $\mathbf{p}_{new}$ , as follows:

$$\mathbf{M}^s(\hat{C}(\mathbf{p}_{new})) = \mathbf{M}^s \left( G^C + \sum_{i=1}^{m^C} \Psi_i^C \hat{\alpha}_i^C(\mathbf{p}_{new}) \right) = \mathbf{M}^s(G^C) + \sum_{i=1}^{m^C} \mathbf{M}^s(\Psi_i^C) \hat{\alpha}_i^C(\mathbf{p}_{new}) \tag{5.22}$$

Using the reduced formulation, the mass matrix can be written as:

$$\begin{aligned}\hat{\mathbf{M}}^s(\hat{\mathbf{C}}(\mathbf{p}_{new})) &= \hat{\mathbf{M}}^s\left(G^C + \sum_{i=1}^{m^C} \Psi_i^C \hat{\alpha}_i^C(\mathbf{p}_{new})\right) = \hat{\mathbf{M}}^s(G^C) + \sum_{i=1}^{m^C} \hat{\mathbf{M}}^s(\Psi_i^C) \hat{\alpha}_i^C(\mathbf{p}_{new}) \\ &= \mathbf{U}^{h'} \left[ \mathbf{M}^s(G^C) + \sum_{i=1}^{m^C} \mathbf{M}^s(\Psi_i^C) \hat{\alpha}_i^C(\mathbf{p}_{new}) \right] \mathbf{U}^h\end{aligned}\quad (5.23)$$

The complex operation of linearising the mass matrix is graphically demonstrated in Fig. 5.4. The matrices now only need to be calculated once, and added according to the weight values derived from the metamodels each time the matrix need to be approximated. Using the same approach as that used for the mass

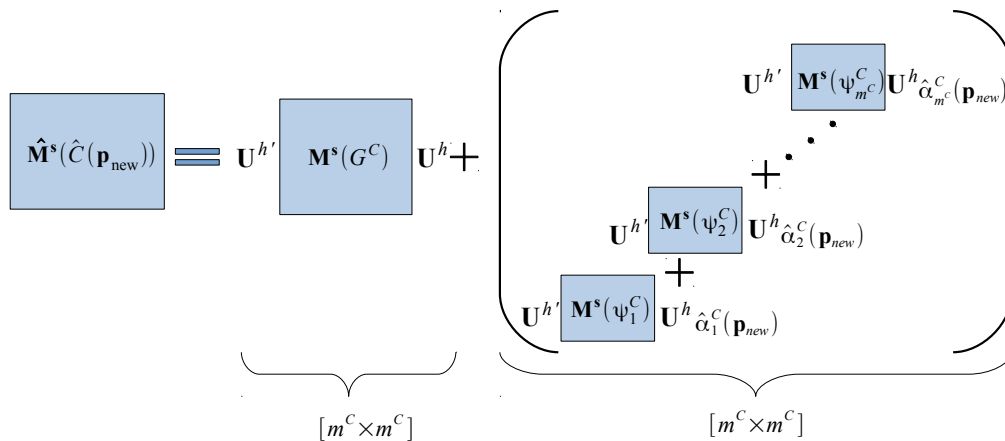


Figure 5.4: The linearisation of the mass matrix, where the weight value vector  $\hat{\alpha}^K$  is the unknown

matrix, the stiffness matrix can be approximated as:

$$\hat{\mathbf{M}}^p(\hat{\mathbf{K}}(\mathbf{p}_{new})) = \mathbf{U}^{h'} \left[ \mathbf{M}^p(G^K) + \sum_{i=1}^{m^K} \mathbf{M}^p(\Psi_i^K) \hat{\alpha}_i^K(\mathbf{p}_{new}) \right] \mathbf{U}^h \quad (5.24)$$

In the creation of the ROM, all these integral matrices do not change because the global shape functions do not vary with a change in parameters. Furthermore, due to the small size of the reduced matrices, very little storage space is required to store the reduced matrices. When approximating the mass and stiffness matrices, only the  $\alpha_i^C$  and  $\alpha_i^K$  values now need to be calculated from metamodels during each ROM execution. In a similar manner, the vector terms can now be reduced.

#### 5.1.4 Reducing vector terms using the Hybrid approach

The reduction of the vectors follows the same approach used to reduce the matrices. The non-linear functions in the integral vectors are again approximated

by the linear sum of the non-linear principal components. The first vector term in Eq. 5.17 is  $\mathbf{V}^s$ , which can be represented in terms of  $\mathbf{M}^s$  in Eq. 5.22 as follows:

$$\mathbf{V}^s(\mathbf{p}_{new}) = \mathbf{M}^s(\hat{C}(\mathbf{p}_{new}))\hat{\mathbf{h}}^\tau \quad (5.25)$$

The reduced form of  $\mathbf{V}^s$  is reduced by projecting it onto the reduced basis set:

$$\hat{\mathbf{V}}^s(\mathbf{p}_{new}) = \mathbf{U}^{h'}\mathbf{V}^s(\mathbf{p}_{new}) = \mathbf{U}^{h'}\mathbf{M}^s(\hat{C}(\mathbf{p}_{new}))\hat{\mathbf{h}}^\tau \quad (5.26)$$

The reduced form can be further reduced by linearising  $\hat{\mathbf{h}}^\tau$ :

$$\hat{\mathbf{V}}^s(\mathbf{p}_{new}) = \mathbf{U}^{h'}\mathbf{M}^s(\hat{C}(\mathbf{p}_{new})) \left[ \sum_{i=1}^{m^c} \hat{\alpha}_i^\tau \Psi_i + \mathbf{G}^h + \mathbf{b}^{h,\tau} \right] \quad (5.27)$$

where  $\mathbf{b}^{h,\tau}$  is the Dirichlet vector at the previous time step. Dropping the argument  $\hat{C}(\mathbf{p}_{new})$ , the vector is finally rewritten as:

$$\begin{aligned} \hat{\mathbf{V}}^s(\mathbf{p}_{new}) &= \mathbf{U}^{h'}\mathbf{M}^s\mathbf{U}^h\alpha^\tau + \mathbf{U}^{h'}\mathbf{M}^s\mathbf{G}^h + \mathbf{U}^{h'}\mathbf{M}^s\mathbf{b}^{h,\tau} \\ &= \hat{\mathbf{M}}^s\alpha^\tau + \mathbf{U}^{h'}\mathbf{M}^s\mathbf{G}^h + \mathbf{U}^{h'}\mathbf{M}^s\mathbf{b}^{h,\tau} \end{aligned} \quad (5.28)$$

The next term contains the Neumann boundary conditions. This term is calculated using the same approach as the one used in the GPOD method in section 3.2.5. The Neumann boundary term is calculated as:

$$\hat{\mathbf{V}}^n(f_n) = \mathbf{U}^{h'}\mathbf{V}^n(f_n) = \mathbf{U}^{h'} \begin{bmatrix} \int_{\Gamma_n} f_n v_1 \cdot \mathbf{n} \\ \vdots \\ \int_{\Gamma_n} f_n v_n \cdot \mathbf{n} \end{bmatrix} \quad (5.29)$$

The following vector,  $\mathbf{V}^p$ , is linearised similarly to the matrix linearisation. Following the procedure used in the matrix linearisation, the vectors are redefined as a function of  $K$  as follows:

$$\mathbf{V}^p(K) = \begin{bmatrix} \int_{\Omega} K \frac{\partial v_1}{\partial z} \\ \vdots \\ \int_{\Omega} K \frac{\partial v_n}{\partial z} \end{bmatrix} \quad (5.30)$$

The vector is reduced by projecting it onto the global basis set  $\mathbf{U}^h$ :

$$\hat{\mathbf{V}}^p(K) = \mathbf{U}^{h'}\mathbf{V}^p(K) \quad (5.31)$$

Now the vector term  $\hat{\mathbf{V}}^p$  is calculated by substituting the linearisation for  $K$  (Eq. 5.4) into Eq. 5.31:

$$\hat{\mathbf{V}}^p(\hat{K}(\mathbf{p}_{new})) = \mathbf{U}^{h'} [\mathbf{V}^p(\hat{K}(\mathbf{p}_{new}))] = \mathbf{U}^{h'} \left[ \mathbf{V}^p(G^K) + \sum_{i=1}^{m^K} \mathbf{V}^p(\Psi_i^K) \hat{\alpha}_i^K(\mathbf{p}_{new}) \right] \quad (5.32)$$

where the vector is reduced to size  $[m^K \times 1]$ . As is the case with the matrix linearisation, the weight values  $\hat{\alpha}^K(\mathbf{p}_{new})$  are evaluated with the metamodels for each weight function.

The final two vectors in Eq. 5.16 are functions of the mass and stiffness matrix. The Dirichlet vector term can be derived in the same way as in the GPOD approach in section 3.2.5, with the difference being that the Hybrid matrix  $\mathbf{M}^s(\hat{C}(\mathbf{p}_{new}))$  and  $\mathbf{M}^p(\hat{K}(\mathbf{p}_{new}))$  from Eq.'s 5.24 and 5.23 are used instead of the full matrices, as follows:

$$\hat{\mathbf{V}}^d = \mathbf{U}^{h'}[\mathbf{M}^s + \delta t \mathbf{M}^p] \mathbf{b}^{h,\tau+1} \quad (5.33)$$

Since  $\mathbf{b}^{h,\tau+1}$  is only non-zero at the Dirichlet nodes, only the matrix columns corresponding to these nodes need to be stored. The gravity centre term is calculated in the same way:

$$\hat{\mathbf{V}}^d = \mathbf{U}^{h'}[\mathbf{M}^s + \delta t \mathbf{M}^p] \mathbf{G}^h \quad (5.34)$$

With all the matrices and vectors terms used in the Hybrid method defined, the terms can now be assembled to approximate the pressure head response.

### 5.1.5 Solving for pressure head with the Hybrid approach

The system of ODE's in Eq. 5.17 can now be solved for the pressure head weight values  $\alpha$ . Using the implicit Euler technique and defining the parameter vector at the next time step as  $\mathbf{p}^{\tau+1}$ , the left hand terms in Eq. 5.17 can be expanded as:

$$\hat{\mathbf{M}}_{HYB} \alpha^{\tau+1} = [\hat{\mathbf{M}}^s(C(\mathbf{p}^{\tau+1})) + \delta t \hat{\mathbf{M}}^p(C(\mathbf{p}^{\tau+1}))] \alpha^{\tau+1} \quad (5.35)$$

The vector terms in the Hybrid POD model can also be assembled, by expanding the terms that are a function of the mass and stiffness matrices. For clarity, the matrix and vector arguments are dropped where suitable. The vector terms are added together and manipulated as follows:

$$\begin{aligned} \hat{\mathbf{V}}_{HYB} &= \hat{\mathbf{V}}^s + \delta t[\hat{\mathbf{V}}^n - \hat{\mathbf{V}}^p] - \hat{\mathbf{V}}^p - \hat{\mathbf{V}}^g \\ &= \hat{\mathbf{M}}^s \alpha^\tau + \mathbf{U}^{h'} \mathbf{M}^s \mathbf{G}^h + \mathbf{U}^{h'} \mathbf{M}^s \mathbf{b}^{h,\tau} + \delta t[\hat{\mathbf{V}}^n - \hat{\mathbf{V}}^p] - \mathbf{U}^{h'}[\mathbf{M}^s + \delta t \mathbf{M}^p][\mathbf{G}^h + \mathbf{b}^{h,\tau+1}] \\ &= \hat{\mathbf{M}}^s \alpha^\tau + \mathbf{U}^{h'} \mathbf{M}^s \mathbf{b}^{h,\tau} + \delta t[\hat{\mathbf{V}}^n - \hat{\mathbf{V}}^p] - \delta t \mathbf{U}^{h'} \mathbf{M}^p \mathbf{G}^h - \mathbf{U}^{h'}[\mathbf{M}^s + \delta t \mathbf{M}^p] \mathbf{b}^{h,\tau+1} \\ &= \hat{\mathbf{M}}^s \alpha^\tau + \mathbf{U}^{h'} \mathbf{M}^s [\mathbf{b}^{h,\tau} - \mathbf{b}^{h,\tau+1}] + \delta t[\hat{\mathbf{V}}^n - \hat{\mathbf{V}}^p] - \delta t \mathbf{U}^{h'} \mathbf{M}^p [\mathbf{G}^h + \mathbf{b}^{h,\tau+1}] \end{aligned} \quad (5.36)$$

All the terms can be stored in a reduced form for later access by the ROM. However, a detailed explanation of the optimal storage of the non-varying terms is beyond the scope of this text. The pressure head weight vector  $\alpha$  at the next time step can now be calculated as:

$$\alpha^{\tau+1} = \hat{\mathbf{M}}_{HYB}^{-1} \hat{\mathbf{V}}_{HYB} \quad (5.37)$$

and the ROM pressure head response at the next time step is calculated as:

$$\hat{\mathbf{h}}^{\tau+1} = \mathbf{U}^h \alpha^{\tau+1} + \mathbf{G}^h \quad (5.38)$$

One major advantage in using the Hybrid approach is that a convergence scheme, such as the Picard iteration scheme, is no longer necessary due to the non-linear functions and matrices being decoupled from the pressure head. The removal of a convergence scheme considerably reduces the number of iterations. The Hybrid approach is explained concisely in Algorithm 1, where each step is briefly recalled.

The application of this method is shown on a case study at the end of this chapter, in which good speed up-times are yielded, when compared to the full model. While this is already a good speed up time, an alternative approach is explored in the following section to find better speed-up times, as well as a method to avoid the use of explicit interpolation techniques when evaluating the weight values for the non-linear functions.

---

**Algorithm 1** The Hybrid approach
 

---

- 1: **Training point/points:** Create a set of high fidelity responses at a DOE. (e.g. Table 5.1)
  - 2: **Principal Component Analysis:** Use snapshot set and extract global shape functions for  $h$ ,  $C$  and  $K$  with a PCA. (Eq.'s 5.10 and 5.11)
  - 3: **Create reduced matrices:** With a choice of  $m^C$  and  $m^K$ , integrate full matrices  $\hat{\mathbf{M}}^C(\hat{K})$ ,  $\hat{\mathbf{M}}^C(\hat{C})$  and store matrices. (Eq.'s 5.23 and 5.24)
  - 4: **Create reduced vector terms:** Create and store vector terms in Eq. 5.36.
  - 5: Setup initial conditions
  - 6: Evaluate  $\alpha_{t=0}$ ,  $\alpha_{t=0}^K$ ,  $\alpha_{t=0}^C$  by projecting initial conditions onto their GSF's.
  - 7: **procedure** HYBRID:
  - 8:   **for**  $k = 0 : nts - 1$  **do** # Iterate through time
  - 9:     Evaluate  $\hat{\alpha}^K(\mathbf{p}^{\tau+1})$ ,  $\hat{\alpha}^C(\mathbf{p}^{\tau+1})$  using the metamodels
  - 10:    Evaluate  $\hat{\mathbf{M}}^s(\mathbf{p}^{\tau+1})$  and  $\hat{\mathbf{M}}^p(\mathbf{p}^{\tau+1})$  from Eq.'s 5.23 and 5.24
  - 11:    Evaluate vector terms in Eq. 5.36
  - 12:    Find  $\alpha^{\tau+1}$  from Eq. 5.37
  - 13:    Find  $\hat{\mathbf{h}}^{\tau+1}$  from Eq. 5.38
  - 14:   **end for**
  - 15: **end procedure**
- 

## 5.2 The Linearised POD approach

In this section, we propose an alternative approach to evaluating the non-linear weight values that avoids the creation of metamodels in the Hybrid method. As in the Hybrid approach, representative snapshot sets  $\mathbf{H}$ ,  $\mathbf{C}$  and  $\mathbf{K}$  are assembled and the reduced basis sets are found by performing a PCA on the snapshot set, to yield  $\mathbf{U}^h$  and  $\mathbf{G}^h$ ,  $\mathbf{U}^C$  and  $\mathbf{G}^C$ , and  $\mathbf{U}^K$  and  $\mathbf{G}^K$ . These Reduced Basis Sets, and gravity centre terms, are used to create the linearised matrices (Eq.'s 5.23 and 5.24) and vectors (Eq. 5.36) in the same way as the Hybrid approach. The main

difference in this approach is that the non-linear weight values are found by projecting the non-linear functions directly onto their respective reduced basis sets at each iteration in the ROM simulation, instead of using a metamodel. This approach is called the Linearised GPOD (LPG) in order to distinguish it from the Hybrid approach, and to emphasise the further linearisation of the non-linear terms.

### 5.2.1 Finding weight vectors

In order to find the weight values, we first define  $\hat{\mathbf{h}}_i \in \mathbb{R}^n$  as the ROM pressure head response at the  $i^{\text{th}}$  time step. The unknown weight vectors  $\hat{\alpha}^C$  and  $\hat{\alpha}^K$  can be found at each iteration of the ROM, by first evaluating  $C(\hat{\mathbf{h}}_i)$  and  $K(\hat{\mathbf{h}}_i)$ , by normalising and projecting these functions onto their respective reduced basis sets,  $\mathbf{U}^C$  and  $\mathbf{U}^K$ , as follows:

$$\hat{\alpha}^C(\hat{\mathbf{h}}_i) = \mathbf{U}^{C'} [C(\hat{\mathbf{h}}_i) - \mathbf{G}^C] \quad (5.39)$$

and

$$\hat{\alpha}^K(\hat{\mathbf{h}}_i) = \mathbf{U}^{K'} [K(\hat{\mathbf{h}}_i) - \mathbf{G}^K] \quad (5.40)$$

Now using these responses, the matrices and vectors are set up in the same way as the Hybrid method. By projecting the non-linear functions onto their RBS, the need for a metamodel is eliminated, but the full evaluation of the non-linear functions is required at each iteration. The full evaluation, and the subsequent projection of these functions onto their RBS's is the most significant time expense in each iteration.

### 5.2.2 Solving for pressure head with the LGP method

The time term in Eq. 5.17 is expanded and modified to be solved with Picard iteration scheme using the matrices and vectors defined for the Hybrid approach as follows:

$$\hat{\mathbf{M}}_{LGP} \alpha^{\tau+1} = \hat{\mathbf{V}}_{LGP} \quad (5.41)$$

where

$$\hat{\mathbf{M}}_{LGP} = \hat{\mathbf{M}}^s(C(\hat{h}^{\tau+1})) + \delta t \hat{\mathbf{M}}^p(K(\hat{h}^{\tau+1})) \quad (5.42)$$

The mass and stiffness matrices are linearised in the same way that the Hybrid approach used in Eq.'s 5.23 and 5.24, except that the weight values are found by direct projection onto the reduced basis sets in Eq.'s 5.39 and 5.40. The same vector terms that are developed for the Hybrid approach in Eq. 5.36 are used again for the LGP method, by adapting the arguments to the different terms:

$$\begin{aligned} \hat{\mathbf{V}}_{LGP} = & \mathbf{U}^{h'} \mathbf{M}^s(C(\hat{h}^{\tau+1})) \mathbf{U}^{h'} \alpha^\tau + \mathbf{U}^{h'} \mathbf{M}^s(C(\hat{h}^{\tau+1})) [\mathbf{b}^{h,\tau} - \mathbf{b}^{h,\tau+1}] \\ & + \delta t [\hat{\mathbf{V}}^m(K(\hat{h}^{\tau+1})) - \hat{\mathbf{V}}^p(K(\hat{h}^{\tau+1}))] - \delta t \mathbf{U}^{h'} \mathbf{M}^p(K(\hat{h}^{\tau+1})) [\mathbf{G}^h + \mathbf{b}^{h,\tau+1}] \end{aligned} \quad (5.43)$$

where the vector  $\hat{\mathbf{V}}^p$  and the matrices  $\mathbf{M}^s$  and  $\mathbf{M}^p$  are again approximated using the linearisation approach, with weight values found from Eq.'s 5.39 and 5.40. The  $\alpha$  values can now be calculated as:

$$\alpha^{\tau+1} = \hat{\mathbf{M}}_{LGB}^{-1} \hat{\mathbf{V}}_{LGB} \quad (5.44)$$

and the final response can be calculated with:

$$\hat{\mathbf{h}}^{\tau+1} = \mathbf{U}^h \alpha^{\tau+1} + \mathbf{G}^h \quad (5.45)$$

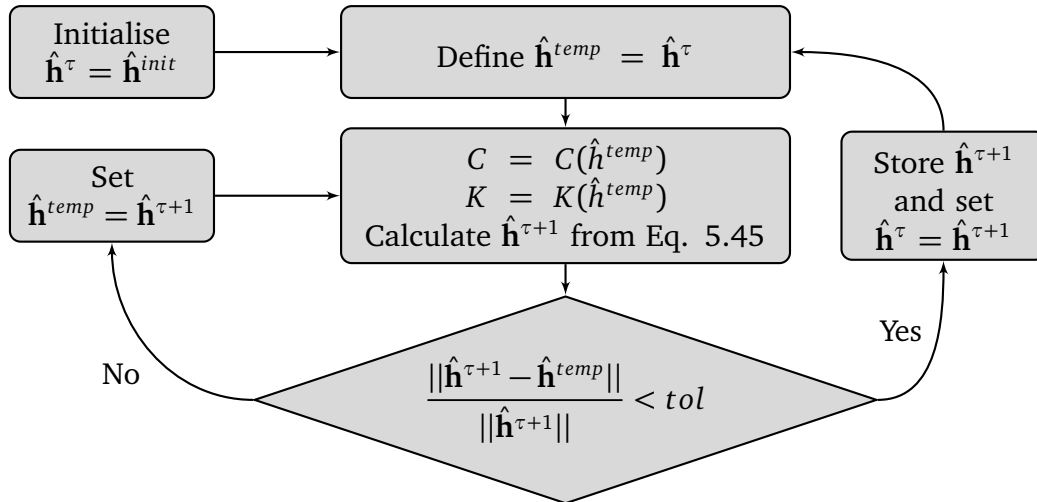


Figure 5.5: Fixed point form method for iterative convergence

In the application of the Picard iteration scheme, each  $\hat{h}^{\tau+1}$  is approximated with a temporary value  $\hat{h}^{temp}$ . Following the Picard scheme shown in Fig 5.5, Eq. 5.45 is run iteratively until the solution converges to a predefined tolerance. In Algorithm 2, the LGP method is shown showing the differences to the Hybrid method. The most significant changes to the Hybrid algorithm are shown on line 10 and 11.

In the following section the Hybrid approach is compared to the LGP approach, as well as the GPOD approach, to determine time and accuracy characteristics of each model. While these methods are applied to the Richards' equation, the approach can be modified to suit many different discretisation problems with parameterised non-linear functions.



---

**Algorithm 2** The proposed Linearised Galerkin POD approach

---

- 1: **Training point/points:** Run a high fidelity simulation/set of simulations at a design point or design points.
  - 2: **Principal Component Analysis:** Select snapshots saved from simulations, and extract global shape functions with a principal component analysis. (See section 3.2 )
  - 3: **Create reduced matrices:** With a choice of  $m^C$  and  $m^K$ , integrate and store the matrices required to recreate  $\hat{\mathbf{M}}^C(\hat{K})$ ,  $\hat{\mathbf{M}}^C(\hat{C})$ . (Eq.'s 5.23 and 5.24)
  - 4: **Create reduced vector terms:** Create and store static vector terms in Eq. 5.43.
  - 5: Setup initial conditions
  - 6: Evaluate  $\alpha_{\tau_0}, \alpha_{\tau_0}^K, \alpha_{\tau_0}^C$  by projecting initial conditions onto their GSF's.
  - 7: **procedure** LGP:
    - 8:     **for**  $\tau = \tau_1 : \tau_{final}$  **do** # Iterate through time
    - 9:         **while**  $Err < 0.001$  **do** # Check convergence
    - 10:             Evaluate non-linear functions  $C(\hat{h}^{\tau+1})$  and  $K(\hat{h}^{\tau+1})$  (Table 2.1)
    - 11:             Evaluate  $\hat{\alpha}^{K, \tau+1}, \hat{\alpha}^{C, \tau+1}$  with Eq.'s 5.39 and 5.40
    - 12:             Evaluate  $\hat{\mathbf{M}}^S(\hat{C}^{\tau+1})$  and  $\hat{\mathbf{M}}^P(\hat{K}^{\tau+1})$  from Eq.'s 5.23 and 5.24
    - 13:             Evaluate vector terms in Eq. 5.43
    - 14:             Find  $\alpha^{\tau+1}$  from Eq. 5.44
    - 15:             Find  $\hat{\mathbf{h}}^{\tau+1}$  from Eq. 5.45
    - 16:             Determine error (Fig. 5.5):  $Err = \frac{\|\mathbf{h}^{\tau+1} - \mathbf{h}^{temp}\|}{\|\mathbf{h}^{\tau+1}\|}$
    - 17:             **end while**
    - 18:         **end for**
    - 19:     **end procedure**
-

### 5.3 Example evaluation of Hybrid and LGP approaches

In order to evaluate the Hybrid and LGP methods, we use the same example problem introduced in section 4.2. The Hybrid and LGP methods are implemented in Python, while the full simulation is done in FreeFEM++. In order to gain insight into the time and accuracy of the two reduced models, the time and accuracy is evaluated for different values for  $m^h$ . The errors created by the reduced model responses are then compared to the response of the system in a full FE simulation. The domain is shown again in Fig. 5.6, and is subject to the Richards' equation and boundary conditions shown in Eq. 5.46. The example problem used in section 4.2 is recalled:

$$\begin{aligned}
 C(h) \frac{\partial h}{\partial t} &= \nabla \cdot [K(h) \nabla (h + z)] \\
 h(x, z, 0) &= h_{init} \\
 f_n(x, z, t) &= -1 \text{ cm/hr on } \Gamma_{w1} \\
 f_n(x, z, t) &= -1 \text{ cm/hr on } \Gamma_{w2} \\
 f_n(x, z, t) &= q_{rain}(t) \cdot \mathbf{n} \text{ cm/hr on } \Gamma_r \\
 h(x, z, t) &= (180 - z) \text{ cm on } \Gamma_{d1}, \Gamma_{d2} \\
 t &\in [0, 500 \text{ hr}]
 \end{aligned}
 \quad
 \begin{aligned}
 K_s &= 0.35 \text{ cm/hr} \\
 m &= 0.4 \\
 h_b &= 20 \text{ cm} \\
 \theta_s &= 0.4 \\
 \theta_r &= 0 \\
 S_s &= 0
 \end{aligned}
 \tag{5.46}$$

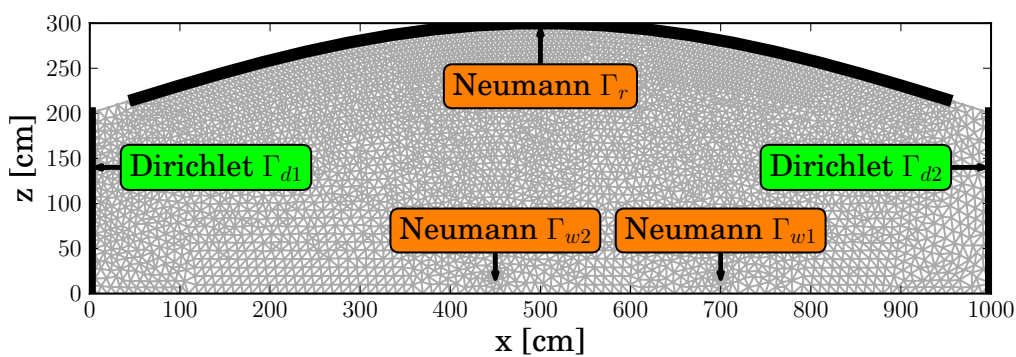


Figure 5.6: Numerical case study mesh with  $\approx 5000$  node points

### 5.3.1 Weight values, eigenvalues and eigenvectors

The eigenvalues, the mean values and the eigenvectors are derived after applying PCA to the snapshot set generated from the example. In Fig. 5.7 the eigenvectors are shown. It is seen that the eigenvalues corresponding to the eigenvectors decrease exponentially, indicating that the response can be replicated with a small number of GSF's.

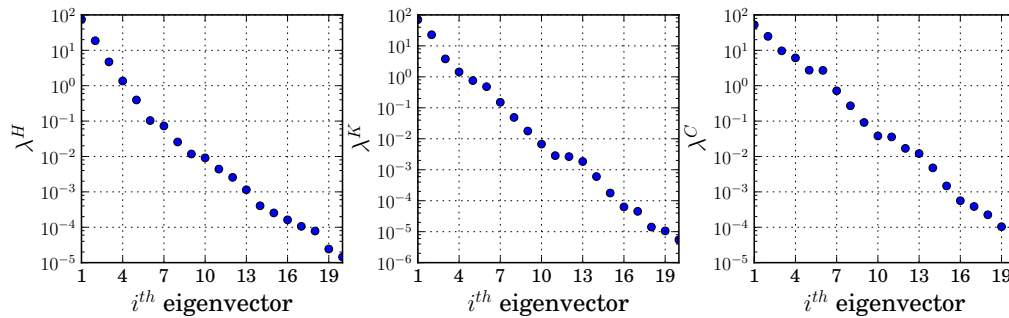


Figure 5.7: Exponentially decreasing eigenvalues resulting from the PCA on the normalised snapshot sets

In Fig 5.8 the first three GSF's derived from the pressure head response snapshot set are shown, and in Fig.'s 5.9 and 5.10 the first three eigenvectors derived from the non-linear functions are plotted. The eigenvectors show variation principally at the water table, which is expected because of the non-linear functions having a transition zone around  $h = 0$ .

The behaviour of the weight values can be seen in Fig. 5.11. These weight values are found by projecting the true response onto their respective eigenvectors. Due to the time-varying infiltration, these weight values do not reach a steady state. One interesting observation is that the behaviour of  $\alpha$ ,  $\alpha^C$  and  $\alpha^K$  does not appear to be directly linked. Using these GSF's, the different ROM's are now applied.

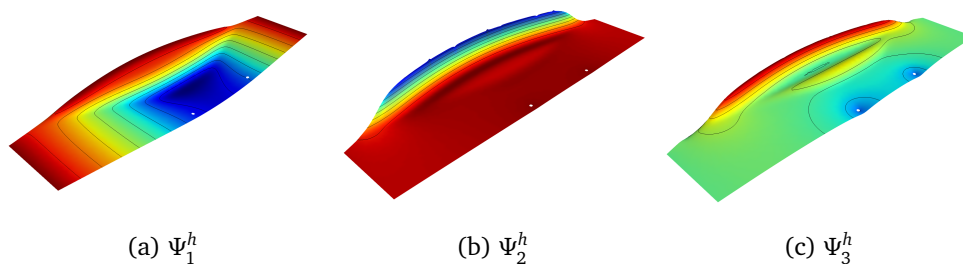


Figure 5.8: First three pressure head eigenvectors

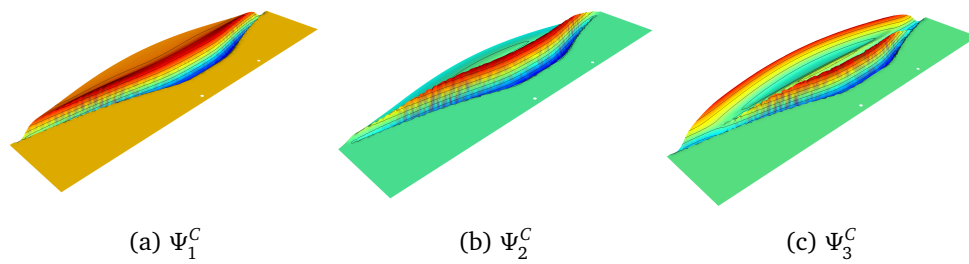


Figure 5.9: First three moisture content eigenvectors

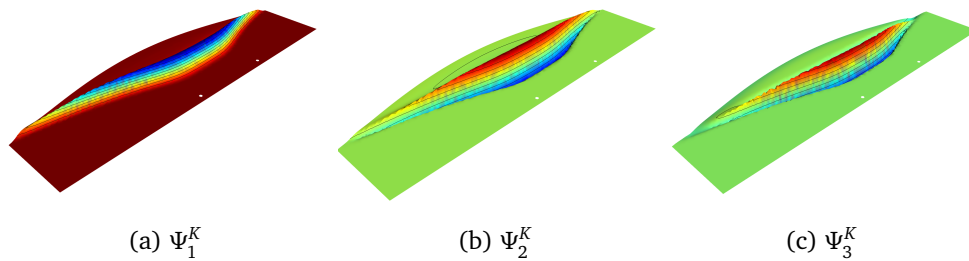


Figure 5.10: First three conductivity eigenvectors

### 5.3.2 Model reduction using the Hybrid method

The Hybrid approach is implemented using radial basis functions to create the metamodels, and follows the procedure described in algorithm 1. In this example, the case study presented above is replicated using the Hybrid method. The only parameter that varies is time, allowing for the use of a very simple DOE. The snapshots for pressure head and the non-linear function responses are collected at 5 hr intervals, resulting in a set of 100 snapshots of each. After conducting a PCA on these snapshot sets, the metamodels are created for each of the weight functions derived from the non-linear functions.

The training points for the metamodels are found by projecting the snapshots onto the reduced basis set derived from the PCA. The weight values for the first 3 basis functions are shown in Fig.'s 5.11b and 5.11c. The weight functions have a high number of training points, and in the metamodels can be considered exact. These weight values correspond to the global basis functions shown earlier in Fig.'s 5.9 and 5.10. The RBF metamodels are created using the RBF function in the SciPy interpolation library (Jones *et al.* (2001)) in the Python coding

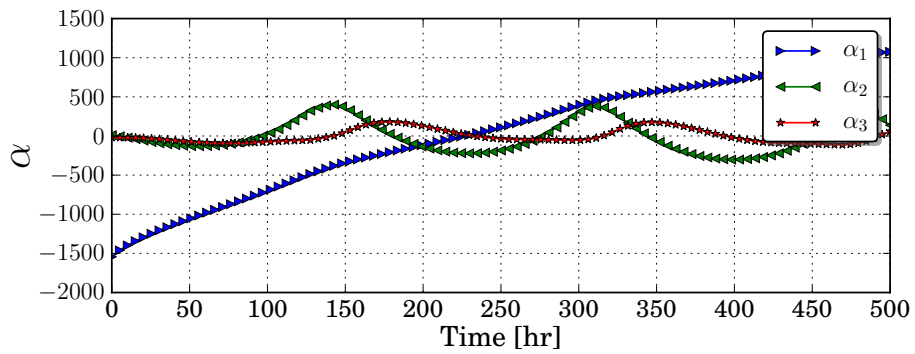
environment. In this case, the default kernel parameters in the RBF function was used without any adaptation.

The speed-up in the Hybrid ROM when applied to the Richards' equation is constrained by the number of metamodel function calls, as well as the time required per metamodel function call. During the execution of the ROM, no iterative scheme is required since the approximated  $C(h^{\tau+1})$  and  $K(h^{\tau+1})$  functions are evaluated directly from the metamodel, and do not need to be determined iteratively. If we define the number of time steps as  $nts$  and we choose the same number of shape functions for pressure head and the non-linear functions by setting  $m = m^h = m^K = m^C$ , a total number of metamodel calls  $n_{fc} = 2 \cdot nts \cdot m$  need to be executed. In the example problem, the number of time steps are  $nts = 100$ . If we choose the number of shape functions as  $m = 15$ , each simulation requires a total number of weight-function calculations  $n_{fc} = 3000$ , which means 3000 metamodel function calls. Each metamodel evaluation has a computational complexity of  $O(n_{tp})$ , where  $n_{tp}$  is the number of training points used by the metamodel. In spite of this large number of metamodel function calls, a ROM speed up time of  $\approx 50$  is achieved when compared to the time required to execute the full simulation. The time per ROM execution for different sized basis sets is shown in Fig. 5.12a, and the corresponding RRMS (discussed in section 5.3.4) error is shown in Fig. 5.12b. The same problem is also analysed using the LGP method.

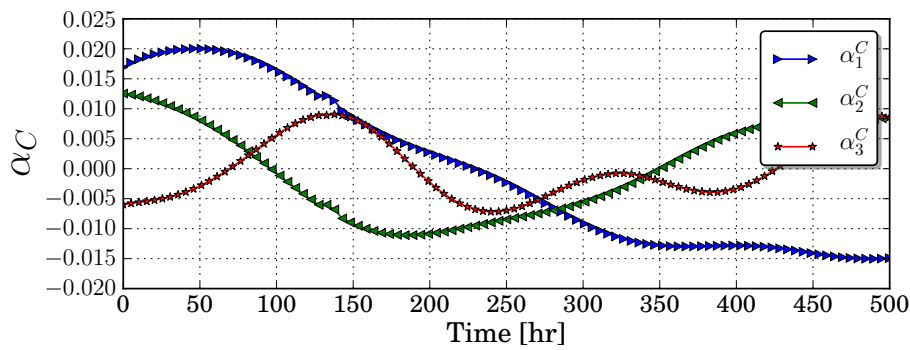
### 5.3.3 Model reduction using the LGP method

In contrast to the Hybrid approach, the function calls are not to metamodels. Instead, the non-linear functions are evaluated using the pressure head approximated with the ROM. Also, the number of function calls are not dependent on the number of global shape functions used, but rather on the number of time steps and iterations to convergence per time-step. If the average number of iterations to convergence per time-step is  $n_{it} = 5$ , the LGP requires  $2 \cdot nts \cdot n_{it} = 1000$  functional calls to evaluate  $C$  and  $K$  at  $\hat{h}$ , and to project these functions onto their reduced basis sets.

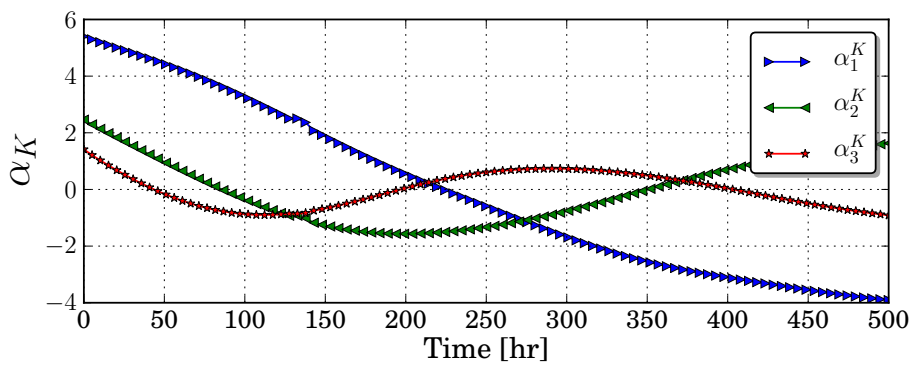
The LGP method also shows further improved speed-up times over the Hybrid method, with an average speed-up of  $\approx 70$  over the values of  $m$  tested, as shown in Fig. 5.12a. It is interesting to note that there is a very small increase in time with each increase in the number of GSF's. This is due to a small increase in computational complexity brought about by the projection of the non-linear functions onto slightly augmented reduced basis sets, and small increases in the reduced matrix size. The corresponding RRMS error is evaluated and plotted in Fig. 5.12b. In the following section, the time and accuracy of the different POD-based ROM's is discussed.



(a) Weight values for the head response



(b) Weight values for the storativity function



(c) Weight values for the conductivity function

Figure 5.11: Weight values over time corresponding to the eigenvectors. These values are found by projecting the true response onto the derived eigenvectors directly.

### 5.3.4 Accuracy and time results

Three POD-based ROM's have been developed and implemented in the previous sections and chapter. These methods are called Galerkin POD, the Hybrid and the LGP methods. In order to compare the different methods, we first evaluate time and accuracy of the three methods, when applied to the case study in this chapter, in order to gain insight into each method. The accuracy of the reduced models are determined by evaluating the Relative Root Mean-Square Error (RRMS) used by Vermeulen *et al.* (2004) as follows:

$$\text{RRMS}(\mathbf{H}, \hat{\mathbf{H}}) = 100 \cdot \frac{1}{n_t} \sum_{i=1}^{nts} \sqrt{\frac{\|\mathbf{h}_i - \hat{\mathbf{h}}_i\|^2}{\|\mathbf{h}_i - \mathbf{G}^h\|^2}} \quad (5.47)$$

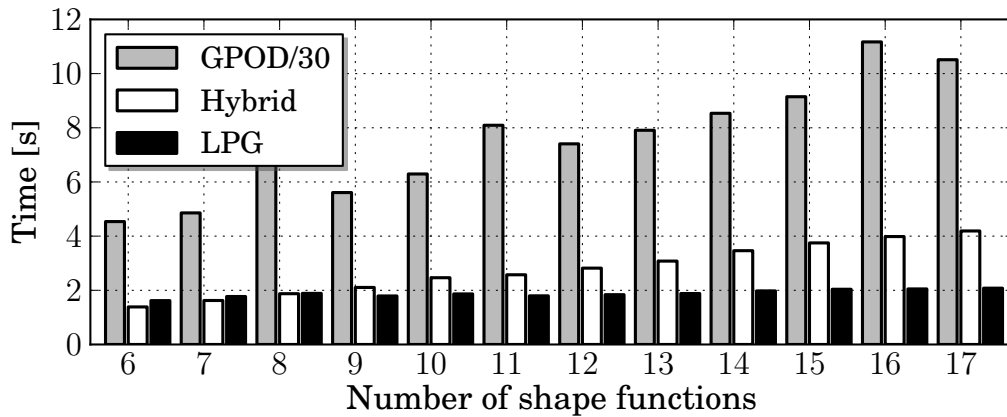
where  $\mathbf{H} = \{\mathbf{h}_1, \dots, \mathbf{h}_{100}\}$  is the FE response, and  $\hat{\mathbf{H}} = \{\hat{\mathbf{h}}_1, \dots, \hat{\mathbf{h}}_{100}\}$  is the response from one of the ROM approaches.

The RRMS for different number values of shape-functions  $m$  is shown in Fig. 5.12b. The graph shows an exponentially decreasing error of the GPOD, the Hybrid and LGP techniques with a corresponding increase in the number of GSF's, which makes all three ROM approaches very appealing. However, in Fig. 5.12a, it is seen that the GPOD method does not gain in time over the original model. In contrast, the Hybrid and LGP method achieve significant speed up times. For example, using 12 GSF's, the LGP technique achieves a speed-up of  $> 70$ , with an RRMS  $< 1\%$ .

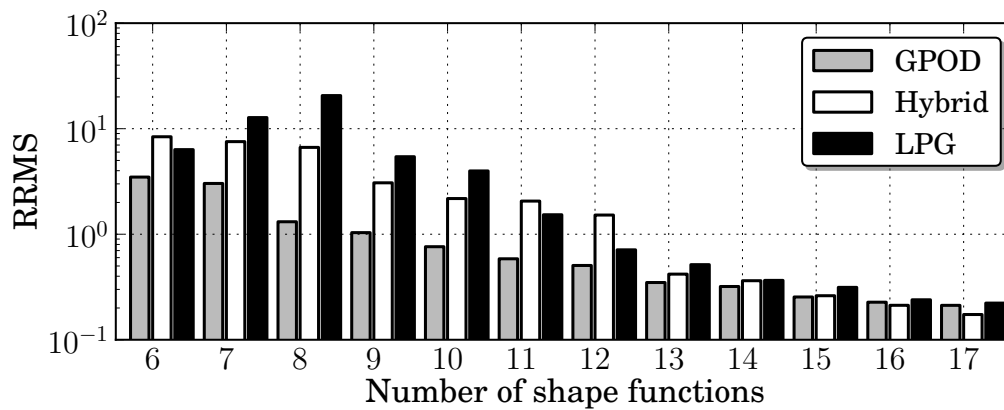
In Fig. 5.12a, the time per simulation required by the GPOD method is divided by 30 to fit into the graph scale. It is clear that the LGP approach performs best in terms of time, with a slightly lower accuracy than the other two methods. The time required per simulation for the LGP approach is seen to increase very slowly with an increase in RBS's. This is principally due to the evaluation and projection of  $C(\hat{h})$  and  $K(\hat{h})$  onto the RBS requiring  $> 90\%$  of the simulation time, when using 10 global shape functions. In the Hybrid approach, the total time to evaluate the weight values does not increase with an increase in GSF's. In contrast, the time to evaluate the weight functions increases linearly with an increase in GSF's in the Hybrid approach.

## 5.4 Discussion on the Hybrid and LGP approaches

The RRMS values for the three approaches are similar once a sufficient number of eigenvectors are used. The GPOD method consistently achieves the lower error for  $m^h \leq 15$ , while the Hybrid method exhibits the greatest accuracy for  $m^h > 15$ . The accuracy in the Hybrid method can be attributed to the fact that the metamodels approximate the exact non-linear functions at each step, and do not require a convergence scheme. This allows for a very accurate approximation, providing the ROM and the metamodels are accurate. However, since the non-linear function evaluations in the Hybrid method are completely decoupled from



(a) Time per simulation



(b) The RRMS error with increasing number of global shape functions

Figure 5.12: Comparison of time and accuracy using an increasing number of GSF's

the pressure head response, this approach can lead to significant errors when the approximated non-linear functions and approximated pressure head response do not match.

In this example, the LPG method exhibits fast run-times per simulation, making it the most attractive method to use in the creation of an intrusive POD-based ROM's for the Richards' equation. The LPG method also avoids the complexity of creating metamodels of the non-linear functions that is required in the Hybrid method, allowing for a faster implementation of a ROM.

The LPG method exhibits RRMS errors very similar to the GPOD and Hybrid methods once sufficient global shape functions have been used, but is generally slightly less accurate than the other two methods. This can be attributed to the fact that the weight values in the LPG method are derived from the projection of  $C(\hat{h})$  and  $K(\hat{h})$  onto their RBS's. Since  $\hat{h}$  is already approximated, this results in a compounded loss in accuracy.

The storage requirements for the Hybrid method is slightly higher than that



of the LGP method, since the  $2m^h$  metamodels need to be stored in addition to the linearised matrices, vectors and reduced basis sets. In contrast, the GPOD method only required the storage of the reduced basis sets.

The Hybrid method can be significantly accelerated by exploring the use of single metamodel that can yield multiple vector results. This is demonstrated in the DACE software in Matlab (Lophaven *et al.* (2002)), and can lead to a further acceleration of  $2m^h$ . However, the time per metamodel function call will increase significantly when more parameters are added. However, since the further use of data-driven methods has been explored extensively in the literature, it is not further explored in this research.

## 5.5 Conclusion

In order to improve on the time characteristics of the GPOD applied to the Richards' equation, two alternative POD-based approaches are developed in this chapter. The methods are implemented on a 2D numerical example, where the LGP method is found to be most efficient in terms of time, while the GPOD method is found to have the highest accuracy. However, the GPOD method exhibits no time-gain in the application. The error from all methods show a decreasing trend with an increase in the number of eigenvectors used, as expected. While the Hybrid method shows acceptable increases in time, it is limited by the high number of metamodel calls when evaluating the weight values. The LGP method shows the best combined characteristics when applied to the Richards' equation.

Thus far, the use of POD-based ROM's have been used to replicate full models. However, the use of a ROM is of considerable interest in the field of surrogate modelling and optimisation, due to the reduction in time in individual simulation times and inverse modelling studies. In the next chapter we develop a method to use the LGP method as a surrogate model in optimisation and inverse modelling studies.

## Chapter 6

# Inverse modelling and updating strategies

THE use of a Reduced Order Model (ROM) is of particular interest in the field of inverse modelling, where input parameters are varied until the numerical response best matches the measured response, or in direct optimisation, where some input parameters are varied until an optimal operating condition is found. Since the ROM's typically have significant speed-ups when compared to a full model, inverse modelling and optimisation problems also exhibit corresponding speed-up times.

The use of Proper Orthogonal Decomposition (POD) based ROM's have been employed in numerous inverse modelling studies, where the ROM's have been shown to be able to accurately approximate a true function response at a parameter point away from the design points used in the creation of the ROM. A number of articles, including those by De Vuyst (2009), Burkardt *et al.* (2006) and Audouze *et al.* (2009), make allusion to the strong interpolation and extrapolation qualities of POD based ROM's. In all references to extrapolation in this chapter, extrapolation refers to a bounded region of the parameter space in which the model exhibits an acceptably low error, but is beyond the region in which the ROM training points are taken. In this chapter we explore two techniques that capitalize on these characteristics, based on the Linearised Galerkin POD (LGP) method developed in the previous chapter.

In order to evaluate the interpolation/extrapolation characteristics, it is necessary to verify that the POD-based ROM has an acceptably low error, at least in the parameter region from which the training points are taken. A ROM that can produce an accurate response only in the region under consideration is said to be locally accurate. To test the ROM interpolation/extrapolation characteristics, snapshots are selected from a number of full simulation responses taken at a number of training parameter points. These snapshots are used to create the ROM, and a second set of parameters are used to create a set of responses, where the second set of responses are used to validate the ROM, and so determine the accuracy of the ROM across the design space.

The use of locally accurate ROM's in inverse modelling can become a necessity, especially when the design space limits are undefined, and the ROM needs to be continually updated. Furthermore, locally accurate ROM's are necessary, for example, when the snapshots from different locations in the parameter spaces create Global Shape Functions (GSF's) that are destructive (or non-representative of local behaviour), instead of constructive. Destructive GSF's can be generated, for example, using snapshots from turbulent behaviour (e.g. high velocity) in fluid flow and snapshots from a laminar flow response (e.g. low velocity). In this case, including snapshots from the turbulent response will degrade the ROM when approximating laminar flow, and vice-versa. Having a locally accurate POD based ROM ensures that only snapshots describing behaviour in the region of interest will be used to create the ROM.

The use of locally accurate models is also proposed by Fahl (2000), who argues in favour of iterative updating, stating that each POD model has limited degrees of freedom. As a result, some system variations cannot be captured and the POD model has to be updated with new information.

A further justification of research into locally accurate ROM's is that high-fidelity models evaluated across a parameter domain can produce large volumes of data, that are difficult or infeasible to process. The use of a locally accurate ROM in numerical optimisation studies allows the iterative creation of a ROM based on high-fidelity snapshots taken from a relevant region of the full parameter space.

The first method of iterative updating investigated in this chapter is known as the Trust-Region-POD (TRPOD) method Fahl (2000), which is an adaptation of the trust region method by Alexandrov *et al.* (1997) to ROM's. The TRPOD has been applied in a number of applications, notably in a fluid flow problems by Bergmann and Cordier (2008) and by Kragel (2005). One significant limitation in the trust region method is the requirement to perform high-fidelity simulations during optimisation.

A second method is proposed and developed in this thesis that eliminates the need for high-fidelity simulations as the ROM moves through the parameter space. These type of models are known as 'standalone' models. The developed standalone model uses Support Vector Classification (SVC) (Cortes and Vapnik (1995)) to define a zone of high accuracy in the ROM model. The method is similar to the TRPOD method, in that the optimiser is allowed to move inside the parameter region in which the ROM has an acceptable accuracy. The proposed method is called the SVC-POD method.

## 6.1 Trust region approach

The trust-region approach is an approach that uses low-fidelity models to approximate some function responses based on information taken from a high-fidelity model. Here low-fidelity refers to a model of lower accuracy, which typically has a lower computational expense, and a high-fidelity model refers to the original

model, also called a full or true model. The trust-region approach has been proven by Alexandrov *et al.* (1997) to converge to the high-fidelity solution. In this section we show the application of the trust-region method, as developed by Alexandrov *et al.* (1997), to a general low-fidelity model, and subsequently apply it to ROM's.

Let us define some high-fidelity response  $F(\mathbf{p})$  where  $\mathbf{p} = (p_1, \dots, p_n)$  is a vector containing the design variables. The trust region method iteratively creates a quadratic approximation of the true function  $F(\mathbf{p})$ . In using the trust region approach, we define the optimisation problem at iteration  $k$  with a corresponding parameter set  $\mathbf{p}_k$  as:

$$\begin{aligned} \text{Minimise:} \quad & Q_k(\mathbf{p}_k + \mathbf{s}) \\ \text{Subject to:} \quad & \|\mathbf{s}\| \leq \delta^k \end{aligned} \quad (6.1)$$

where  $Q_k$  is the quadratic approximation of  $F$  at  $\mathbf{p}_k$ ,  $\mathbf{s}$  is a vector indicating the step towards the optimisation goal of minimising  $Q_k$ , where the step size is bounded by the trust region constraint  $\|\mathbf{s}\| \leq \delta^k$ . The starting point of the next iteration is updated only if the function decreases:

$$\mathbf{p}_{k+1} = \begin{cases} \mathbf{p}_k + \mathbf{s} & \text{if } F(\mathbf{p}_{k+1}) < F(\mathbf{p}_k) \\ \mathbf{p}_k & \text{otherwise.} \end{cases} \quad (6.2)$$

The iteration continues until the optima is found in the  $k^{\text{th}}$  trust radius, before updating the trust radius. An outline of the method is given in Fig. 6.1, where the full model is used to create an approximate model that is linked to the optimiser. The quadratic approximation and trust region updating are discussed in the following sections.

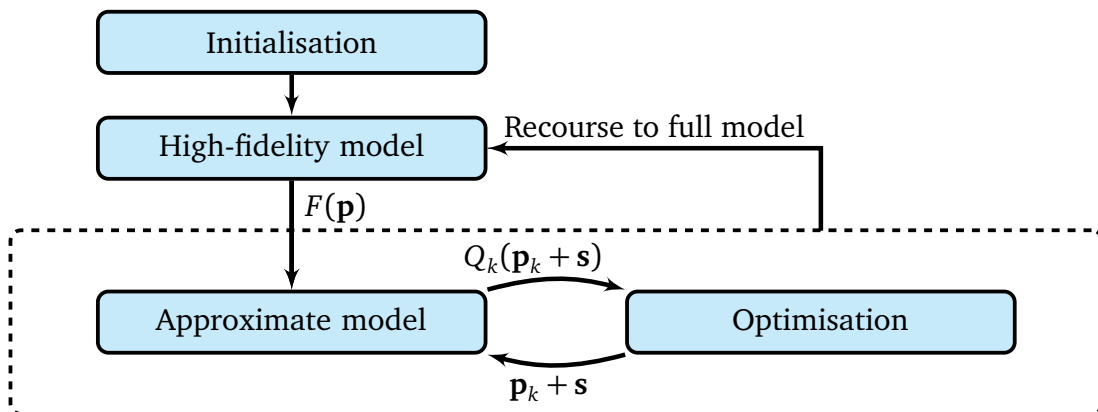


Figure 6.1: Model updating using the trust region POD approach

### 6.1.1 Creating a quadratic approximation

The quadratic model approximates the high-fidelity response  $F$  at the point  $\mathbf{p}_k + \mathbf{s}$  with a quadratic approximation as follows:

$$F(\mathbf{p}_k + \mathbf{s}) \approx Q_k(\mathbf{p}_k + \mathbf{s}) = F(\mathbf{p}_k) + \nabla F(\mathbf{p}_k)' \mathbf{s} + \frac{1}{2} \mathbf{s}' \mathbf{B} \mathbf{s} \quad (6.3)$$

where  $\mathbf{B}$  approximates the Hessian matrix containing the second order gradient information of  $F$  at  $\mathbf{p}_k$ . More information on the Hessian approximations can be obtained in Vanderplaats (2005).

However, before the new optimum point is found, the direction and magnitude of the optimal search vector  $\mathbf{s}$  must be found. In this research we find the search vector using a gradient based approach (Appendix A) such as Sequential Quadratic Programming (SQP) or the Levenberg-Marquardt method. The limitation  $\delta_k$  on the magnitude of search vector indicates the region in which  $Q_k$  is considered to have a sufficiently high accuracy.

### 6.1.2 Updating the trust region

Since the trust radius does not cover the entire design space, the  $k^{\text{th}}$  search radius is constantly adapted according to the accuracy of the quadratic model at the previous iteration  $Q_{k-1}$  within the previous search radius  $\delta_{k-1}$ . The accuracy at iteration  $k$  is determined by a ratio as follows:

$$\rho_k = \frac{F(\mathbf{p}_k) - F(\mathbf{p}_k + \mathbf{s}_k)}{F(\mathbf{p}_k) - Q_k(\mathbf{p}_k + \mathbf{s}_k)} \quad (6.4)$$

The ratio is positive when both functions are decreasing from the same point of origin. We can now use the information from this ratio to update the radius for the next approximation  $Q_{k+1}$ , which will be created at  $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{s}_k$ .

The ratio  $\rho_k$  gives information on the quality of the quadratic approximation  $Q_k$  at  $\mathbf{p}_k + \mathbf{s}_k$ . The 'quality' of this decrease determines whether the subsequent trust region,  $\rho_{k+1}$ , will be greater or smaller than  $\rho_k$ . The procedure to determine the quality of  $Q_k$ , taken from Alexandrov *et al.* (1997), is outlined in Algorithm 3.

---

**Algorithm 3** Trust region updating
 

---

 1: **Choose parameters:**

- $0 < \eta_1 < \eta_2 < 1$ , where  $\eta_1 = 0.25$  and  $\eta_2 = 0.75$
- $0 < \gamma_1 < \gamma_2 < 1 < \gamma_3$ , where  $\gamma_1 = 0.25$ ,  $\gamma_2 = 0.75$ ,  $\gamma_3 = 2$

 2: **if**  $\rho_k \geq \eta_2$  **then:**

- Set  $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{s}_k$  and  $\delta_{k+1} \in [\delta_k, \gamma_3 \delta_k]$

 3: **else if**  $\eta_1 \leq \rho_k < \eta_2$  **then:**

- Set  $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{s}_k$  and  $\delta_{k+1} \in [\gamma_2 \delta_k, \delta_k)$

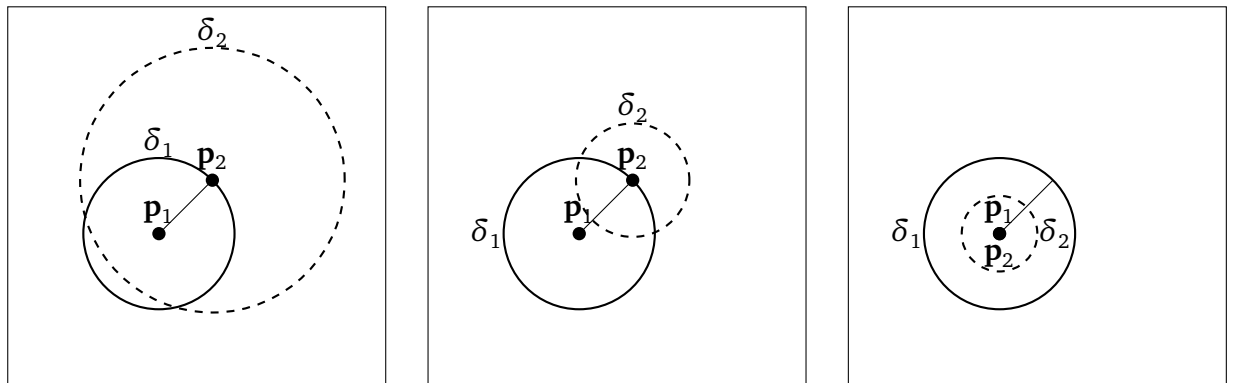
 4: **else**  $\rho_k < \eta_1$ :

- Set  $\mathbf{p}_{k+1} = \mathbf{p}_k$  and  $\delta_{k+1} \in [\gamma_1 \delta_k, \gamma_2 \delta_k]$

 5: **end if**


---

The changes in trust radius, corresponding to the model accuracy in Algorithm 3, is illustrated in Fig. 6.2. Here the three possible changes in trust radius are shown, and the choice of the centre location of the  $(k+1)^{th}$  quadratic approximation. In Fig. 6.2a, the initial point  $\mathbf{p}_1$  moves to point  $\mathbf{p}_2$ , the optimal point within the corresponding trust radius ( $\delta_1$ ), where the low-fidelity model shows a good trend when compared to the high-fidelity model, and so the trust radius is increased to  $\delta_2$ . Similarly, Fig.'s 6.2b and 6.2c show the change in trust radius and initial point to different accuracies reflected by Eq. 6.4.



(a) With  $\rho_k \geq \eta_2$ , the initial point moves and trust radius increases  
 (b) With  $\eta_1 \leq \rho_k < \eta_2$ , the initial point moves, but trust radius decreases  
 (c) With  $\rho_k < \eta_1$ , the initial point does not move and the trust radius decreases

Figure 6.2: Changes in the trust radius in the parameter space from  $\delta_1$  to  $\delta_2$ , and changes in quadratic model initial value, moving from  $\mathbf{p}_1$  to  $\mathbf{p}_2$ .

As an example, we present a 1D example where  $F(p) = \cos(p)^2$ , with a quadratic approximation  $Q_k(p)$  near the origin as shown in Fig. 6.3a. Close to the origin, the approximation is very good ( $\rho > .75$ ), as shown in Fig. 6.3b. If the optimum was found in this area, the trust radius is increased, up to double the previous radius. However, further from the origin the ratio is worse, with ( $0.25 \leq \rho < 0.75$ ). This means the radius can be decreased by up to  $\eta_2 = 0.75$ , should the optimum be found in this area. Lastly, if the optimum is found in the  $\rho < .25$  region, the optimal point becomes the starting point, and the trust region in the following iteration has to be reduced as follows:  $0.25\delta_k \leq \delta_{k+1} \leq 0.75\delta_k$ . This approach is expanded to the application on POD based ROM's in the following section.

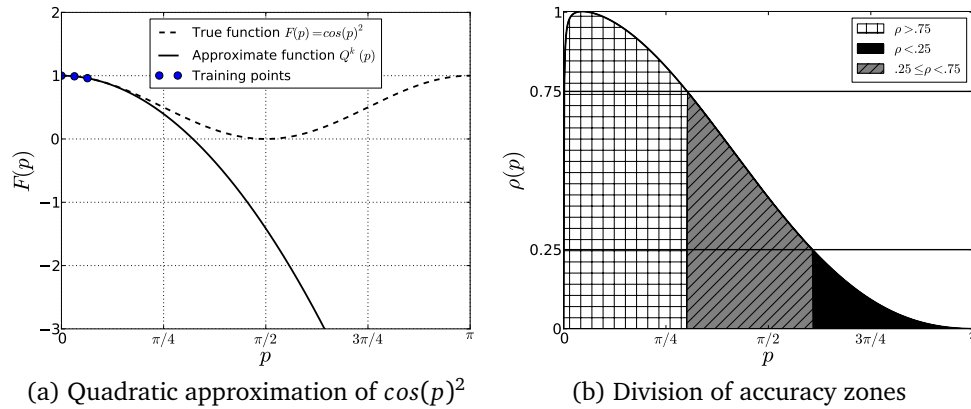


Figure 6.3: One dimensional example of the trust region method

### 6.1.3 Trust region POD

The trust region can be generalised to include approximations that are not of a quadratic nature. An algorithm was proposed by Fahl (2000), where the POD ROM serves as the low-fidelity model. In this approach, the true response is denoted as  $F(\mathbf{p})$ . The POD model will be denoted as  $F^M(\mathbf{p})$ , where  $M$  refers to the number of POD modes used. Fahl proposes that the ratio be determined as the ‘true’ decrease versus the ‘approximate’ decrease in the objective function, so that the ratio  $\rho$  is redefined as follows:

$$\rho_k = \frac{F(\mathbf{p}_k) - F(\mathbf{p}_k + \mathbf{s}_k)}{F^M(\mathbf{p}_k) - F^M(\mathbf{p}_k + \mathbf{s}_k)} \quad (6.5)$$

This permits a bias error in the POD model. Since the trust region is moving through the design space and executing new high-fidelity simulations, the ROM can be updated with the new high-fidelity results at the end of every iteration. The use of the TRPOD method is presented in Algorithm 4.

---

**Algorithm 4** Updating algorithm with the TRPOD method (adapted from Fahl (2000))

---

- 1: **Initialise:** From a DOE run a series of high-fidelity simulations. Choose trust parameters as defined in Algorithm 3.
  - 2: **POD ROM:** Use responses from high-fidelity simulations to build a LGP ROM  $F^M(\mathbf{p}_k)$ , explained in Algorithm 2.
  - 3: **Search direction:** Find the search vector  $\mathbf{s}_k$  to minimise the ROM  $F^M$ , subject to  $\|\mathbf{s}_k\| \leq \delta_k$ , to yield  $F^M(\mathbf{p}_k + \mathbf{s}_k)$
  - 4: **Compute true response:** Evaluate  $F(\mathbf{p}_k + \mathbf{s}_k)$  and evaluate the ratio  $\rho_k$  from Eq. 6.5
  - 5: **procedure** UPDATE TRUST REGION:
  - 6:     **if**  $\rho_k \geq \eta_2$  **then**
  - 7:         Set new starting point  $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{s}_k$
  - 8:         Store  $F(\mathbf{p}_{k+1}) = F(\mathbf{p}_k + \mathbf{s}_k)$
  - 9:         Enlarge  $\delta_{k+1} \in [\delta_k, \gamma_3 \delta_k]$ , set  $k = k + 1$  and go to step 3
  - 10:     **else if**  $\eta_1 \leq \rho_k < \eta_2$  **then**
  - 11:         Set new starting point  $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{s}_k$
  - 12:         Store  $F(\mathbf{p}_{k+1}) = F(\mathbf{p}_k + \mathbf{s}_k)$
  - 13:         Reduce  $\delta_{k+1} \in [\gamma_2 \delta_k, \delta_k]$ , set  $k = k + 1$  and go to step 3
  - 14:     **else**
  - 15:         Keep old starting point  $\mathbf{p}_{k+1} = \mathbf{p}_k$
  - 16:         Reduce  $\delta_{k+1} \in [\gamma_1 \delta_k, \gamma_2 \delta_k]$ , set  $k = k + 1$  and go to step 2
  - 17:     **end if**
  - 18: **end procedure**
- 

In this approach, the ROM is updated only when  $\rho_k \leq \eta_1$ . This allows the expansion of the trust region until the ROM becomes too inaccurate. In each step  $k$ , the movement of the search vector is limited to a spheroid  $\|\mathbf{s}_k\| = \delta_k$ . Once a new point  $F(\mathbf{p}_k + \mathbf{s}_k)$  is evaluated, the results are stored and used to upgrade the ROM once it becomes too inaccurate ( $\rho_k \leq \eta_1$ ). When this happens, the previous high-fidelity responses are used to create a new POD basis. One option is to perform a new PCA on the responses yielded by the high-fidelity simulations. Another option is to enrich the current POD basis set by adding eigenvectors containing system variations from high-fidelity simulations that have not been included in the current basis set. An enrichment method is presented in the following section.

## 6.2 Enrichment of POD basis set

Initially, a locally accurate ROM exists that has been created with a set of eigenvectors  $\mathbf{U}^{orig} \in \mathbb{R}^{n \times m^h}$ , derived from the high-fidelity responses  $\mathbf{H}^{orig} \in \mathbb{R}^{n \times d}$  which have been evaluated at a set of DOE's  $\mathbf{p}_i$ ,  $i = 1, \dots, n_{DOE}$ . As the optimisation moves across the parameter space and out of the parameter space, the low-fidelity



model accuracy is evaluated after each iteration, and if needs be, updated (for example when  $\rho_k \leq \eta_1$ ). Here we consider an approach to enriching a ROM by adding a response, or set of responses, obtained from a new high-fidelity response.

### 6.2.1 Enrichment from one new response

The POD basis can be enriched to consider system variations in some new centred and normalised response  $\bar{F}(\mathbf{p}_i)$ . To include the system variations missing from this point, we follow the Gram-Schmidt orthogonalisation procedure (De Vuyst (2009)) to find a new GSF as follows:

$$\Psi_{m+1} = \bar{F}(\mathbf{p}_i) + \text{Proj}_{\mathbf{U}^{orig}} \bar{F}(\mathbf{p}_i) \quad (6.6)$$

The new GSF  $\Psi_{m+1}$  is the error in the projection of the new point onto the original basis set  $\mathbf{U}^{orig} = \{\Psi_1, \dots, \Psi_m\}$ . Now the ROM basis set is augmented to include the new eigenvector  $\mathbf{U}^{aug} = \{\Psi_1, \dots, \Psi_m, \Psi_{m+1}\}$ , and an upgraded ROM can be created using  $\mathbf{U}^{aug}$ .

### 6.2.2 Enrichment from a new set of responses

Instead of using one new response, it is often desirable to add new system variations from a set of centred, normalised responses  $\bar{\mathbf{Y}}^{new}$  (introduced in section 3.2) to the current POD basis set  $\mathbf{U}^{orig}$ . In order to do this, we first determine which system variations in the new set of responses are poorly represented by  $\mathbf{U}^{orig}$ . Firstly, the projection error is calculated as follows:

$$\mathbf{E} = \bar{\mathbf{Y}}^{new} - \text{Proj}_{\mathbf{U}^{orig}} \bar{\mathbf{Y}}^{new} \quad (6.7)$$

where the projection on the basis set can simply be calculated as:

$$\text{Proj}_{\mathbf{U}^{orig}} \bar{\mathbf{Y}}^{new} = \mathbf{U}^{orig} \cdot [(\mathbf{U}^{orig})^T \cdot \bar{\mathbf{Y}}^{new}] \quad (6.8)$$

Now  $\mathbf{E}$  represents system variations that have not been captured by  $\mathbf{U}^{orig}$ , and so again we can use a PCA on the matrix  $\mathbf{E}$  to optimally add new system variations to  $\mathbf{U}^{orig}$ . As explained in section 3.2, the covariance matrix of  $\mathbf{U}_E$  constructed:

$$\mathbf{C}_m = \frac{1}{n} \mathbf{E}^T \mathbf{E} \quad (6.9)$$

and a new set of eigenvectors  $\mathbf{U}^{new}$  are found by a SVD. Finally, the original eigenvector set is augmented by adding a certain number  $m^{new}$  eigenvectors from the set  $\mathbf{U}^{new}$  to the set  $\mathbf{U}^{orig}$  as follows:

$$\mathbf{U}^{aug} = \left\{ \mathbf{U}^{orig}, \Psi_1^{new}, \dots, \Psi_{m_{new}}^{new} \right\} \in \mathbf{R}^{n \times (m+m_{new})} \quad (6.10)$$

Using the enrichment procedure explained here is significantly faster than performing a PCA on the augmented snapshot set. However, the enrichment process

does not guarantee the same accuracy as a PCA would on the combined sets of original and new snapshots, since the GSF's from the original snapshot set are automatically used, meaning that non-representative system variations might be preserved from the original snapshot set.

In the following section, an alternative approach to enrichment is proposed, in which snapshot sets from a region of the design space are used in an attempt to avoid the problem of non-representative system variations in the enrichment process, as well as the necessity of the TRPOD method of executing high-fidelity model execution during runtime.

### 6.3 Defining a confidence region with SVC

One limitation of the TRPOD method is that the need for high-fidelity simulations during optimisation is not eliminated. In order to eliminate the need for high-fidelity simulations during optimisation, a method is proposed in this section that executes the high-fidelity simulations across the design space before optimisation, where each high-fidelity simulation is run at a point defined by a design of experiments. The use of a low-fidelity model that does not require high-fidelity simulations during an optimisation study is known as a stand-alone model. The stand-alone model is typically built on results obtained by previously executed high-fidelity simulations.

In this section we propose an approach of creating accurate low-fidelity stand-alone models that do not require high-fidelity models to be run during an optimisation study, in contrast to the TRPOD method. The proposed approach is similar to the TRPOD approach since it also creates approximations that are locally accurate. In this proposed method, the locally accurate ROM's are created from responses derived from a region of the entire parameter subspace. This means that only a subset of all the responses derived from the DOE are used. Using this approach means that a percentage of the high-fidelity responses will not be used at all to create ROM's during optimisation. However, there are three reasons that justify the development of locally accurate ROM's that are developed from a subset of responses acquired from previously run high-fidelity simulations, namely:

- The creation of a metamodel, such as kriging, requires the high-fidelity simulations to be done prior to the creation of the metamodel,
- The high-fidelity simulations can be executed in parallel,
- If POD-based ROM's do have the good interpolation and extrapolation qualities alluded to in previous research papers, a low number of design points can be used to create the design of experiments.

In the case studies undertaken in this research, the latter point is found to be true, and the responses from a very small number of DOE's are needed to capture system

variance. The region of the design space that is considered to have a sufficiently high accuracy is defined as a confidence region. In Fig. 6.4 an outline of the proposed method is given, where the ROM is built on high-fidelity simulations run prior to the optimisation. In the following sections, the demarcation of the confidence region is explained, using Support Vector Classification.

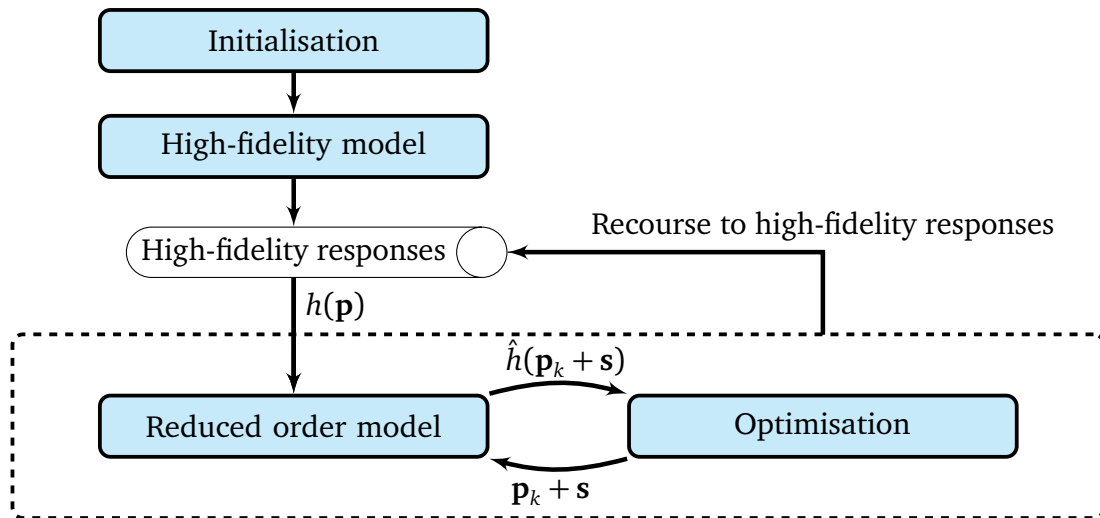


Figure 6.4: Model updating using stored high-fidelity responses to create a locally accurate ROM. In this flow-chart,  $\hat{h}$  refers to the approximated head pressure response from the ROM.

### 6.3.1 Using SVC to define a confidence region

The use of Support Vector Classification (SVC) (Cortes and Vapnik (1995), Schölkopf (1997)) was developed in Bell Labs, with a focus on image recognition. Since then it has been applied on many classification problems, and here we use SVC to classify the parameter space into two zones: A zone in which the ROM is ‘accurate’ and the zone in which the ROM is ‘inaccurate’. As such we call it the SVC-POD method.

This method depends on the responses generated at a number of points over a determined DOE. Methods have been proposed by Siade *et al.* (2010) and Haasdonk (2011) to create a DOE on which to build ROM’s for transient responses. The methods are based on the assumption that the response reaches a steady-state over time. However, in the problems presented in this research, the boundaries are often varying with time, which results in a response that does not reach a steady state.

A more generic approach is therefore needed, and the space filling latin-hypercube (Bates *et al.* (2004)) is chosen with which to create the DOE. All parameters are normalised to lie between limits of  $[0, 1]$ . Once the DOE is

created, a high-fidelity simulation is run at each design point and the results obtained over some specified time are stored.

The method developed in this section proposes the creation of a ROM by using the high-fidelity responses obtained at a subset of the DOE's. The responses derived in this way are used to derive the snapshot set from which to construct the ROM. The DOE subset is typically very small ( $\approx 4$ ) in the applications tested in this paper. Once the ROM has been set up, the training points are evaluated with the ROM. The number of eigenvectors  $m^h$  in the POD basis set is augmented until the training points have a  $RRMS_{m^h} < tol$ , where  $tol$  is some specified tolerance, typically 1%.

Once the ROM is sufficiently accurate at the training points, the response at all the points defined in the DOE are evaluated using the same ROM. Finally, the ROM accuracy at all the points are found by comparing the ROM responses to the high-fidelity results, using the RRMS error defined in Eq. 5.47. All points with a  $RRMS_{m^h} < tol$  are considered as 'accurate', and all points with a  $RRMS_{m^h} \geq tol$  are considered as inaccurate. Finally, we use SVC to create a separation between the accurate and inaccurate points, effectively classifying the two regions. In the following section the SVC approach is introduced, after which the SVC-POD method is outlined concisely in Algorithm 5.

### 6.3.2 Support Vector Classification: Linear case

The simplest use of SVC is the linear separation of two sets of data, with each set being defined with a different label. Let us define a typical data set:

$$D = \{(\mathbf{p}_1, y_1), \dots, (\mathbf{p}_l, y_l)\} \in \mathbb{R}^2, y \in \{-1, 1\} \quad (6.11)$$

with each  $\mathbf{p}_i$ ,  $i = 1, \dots, l$  the parameter set, and each  $y_i$ ,  $i = 1, \dots, l$  labelled either 1 or  $-1$ . An example is shown in Fig. 6.5, where a linear hyperplane is found between the points labelled  $-1$  in the lower left quadrant, and the points labelled  $+1$  in the upper right quadrant. The dotted line represents the hyperplane intersect at  $+1$  and  $-1$ . The solid line represents the separation plane where, in this case, the contour of the separation is 0.

The decision function (defined in Schölkopf (1997)) that is used to classify the different regions is given as:

$$f(\mathbf{p}) = \text{sgn}(\langle \mathbf{w}, \mathbf{p} \rangle + b_{svc}) \quad (6.12)$$

where  $\mathbf{w}$  is the gradient of the linear hyperplane separating two different regions, and  $b_{svc}$  is the bias. To find  $\mathbf{w}$  the following optimisation problem Schölkopf (1997) needs to be solved:

$$\begin{aligned} &\text{Minimise} && \frac{1}{2} |\mathbf{w}|^2 \\ &\text{Subject to:} && y_i (\langle \mathbf{w}, \mathbf{p}_i \rangle + b_{svc}) \geq 1, \quad i = 1, \dots, l \end{aligned} \quad (6.13)$$

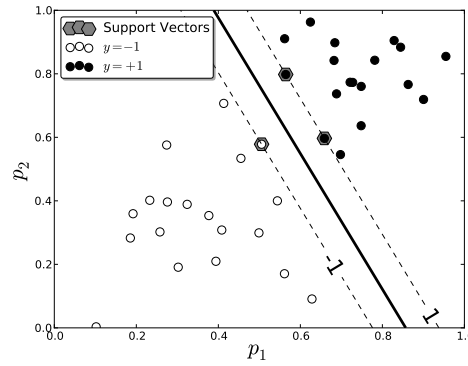


Figure 6.5: Creating a linear separating hyperplane between two data sets

The constraint ensures that each design point is correctly classified, while minimising the gradient. However, it is often not possible to separate all the points, as shown in Fig. 6.6, and so slack variables are introduced as:

$$\xi_i \geq 0, \quad i = 1, \dots, l \quad (6.14)$$

where each  $\xi_i$  gives a measure of the misclassification error. The slack variables are included in the optimisation problem as a penalty in the objective function, and serve to relax the constraint in the original formulation. The optimisation problem now becomes:

$$\text{Minimise:} \quad \frac{1}{2} |\mathbf{w}|^2 + C_{svc} \sum_{i=1}^l \xi_i \quad (6.15)$$

$$\text{Subject to:} \quad \begin{cases} y_i (\langle \mathbf{w}, \mathbf{p}_i \rangle + b_{svc}) \geq 1 - \xi_i, & i = 1, \dots, l \\ \xi_i > 0, & i = 1, \dots, l \end{cases} \quad (6.16)$$

where  $C_{svc}$  is a penalty value used to scale the penalisation of the slack variables. With a higher  $C_{svc}$  value, more training points will become support vectors, while a lower value will typically make use of a lower number of support vectors in inseparable problems.

### 6.3.3 Non-linear mapping using kernels

So far we have only considered linear classification problems. However, many problems require a non-linear separation as shown in Fig. 6.6a. A brief outline is given here of the use of non-linear kernels to map the points to a higher dimensional plane in which the points are *linearly* separable. Firstly, we consider the gradient  $\mathbf{w}$ . From the derivation of the Lagrangian (Schölkopf (1997) Gunn (1998)) of Eq.'s 6.15 and 6.16, we can express the gradient as follows:

$$\mathbf{w} = \sum_{i=1}^l \rho_i y_i \mathbf{p}_i \quad (6.17)$$

where  $\varrho_i$ ,  $i = 1, \dots, l$  are the Lagrange multipliers. Typically, only a percentage of the total training points are used to define the separation. These points are called support vectors. The decision function used to classify the different regions is calculated as:

$$f(\mathbf{p}) = \sum_{i=1}^l \varrho_i y_i \langle \mathbf{p}_i, \mathbf{p} \rangle + b_{svc} \quad (6.18)$$

Here the dot product  $\langle \mathbf{p}_i, \mathbf{p} \rangle$  can be defined as a linear kernel:

$$k(\mathbf{p}_i, \mathbf{p}) = \langle \mathbf{p}_i, \mathbf{p} \rangle \quad (6.19)$$

The linear kernel allows only a linear separation between points. It is possible to replace the linear kernel with a suitable non-linear kernel Schölkopf (1997). One of the most common kernels used is SVC is the exponential Radial Base kernel:

$$k(\mathbf{p}_i, \mathbf{p}) = \exp\left(\frac{-\|\mathbf{p} - \mathbf{p}_i\|^2}{2\sigma^2}\right) \quad (6.20)$$

One immediate problem with non-linear kernels is the choice of parameters, in this case the  $\sigma$  value. The greater the value of  $\sigma$ , the smaller the radius of influence of the non-linear kernel. In this research the RBF kernel is used in the examples, and parameters are adjusted manually to obtain acceptable values in the normalised space, using the *Sklearn* software package by Pedregosa *et al.* (2011).

### 6.3.4 Dual optimisation problem

By taking the Lagrangian of the optimisation problem posed in Eq. 6.15, the optimisation is developed to the following formulation:

$$f(\mathbf{p}) = \text{sgn}\left(\sum_{i=1}^l \varrho_i y_i k(\mathbf{p}_i, \mathbf{p}) + b_{svc}\right) \quad (6.21)$$

$$\text{Minimise} \quad \left\{ \begin{array}{l} \sum_{i=1}^l \varrho_i + \sum_{i,j=1}^l \varrho_i \varrho_j y_i y_j k(\mathbf{p}_i, \mathbf{p}_j) \end{array} \right. \quad (6.22)$$

$$\text{Subject to} \quad \left\{ \begin{array}{l} \sum_{j=1}^l y_j \varrho_j = 0 \\ 0 \leq \varrho_i \leq C_{svc}, \quad i = 1, \dots, l \end{array} \right. \quad (6.23)$$

which is effectively a quadratic optimisation problem (Gunn (1998)). The function in Eq. 6.21 is called the decision function, and is used to classify the two regions as positive or negative. This function is continuous and typically has a value of +1 at the positive support vectors and -1 at the negative support vectors. The SVC is applied on a separable case using the radial base kernel in Fig. 6.6b, where the contours of the decision function at +1 and -1 are shown. The plot shows the decision function derived from Eq. 6.21. The solid dividing line is the zero contour of the decision function. The dotted lines represent the +1 and -1 contours of the decision function.

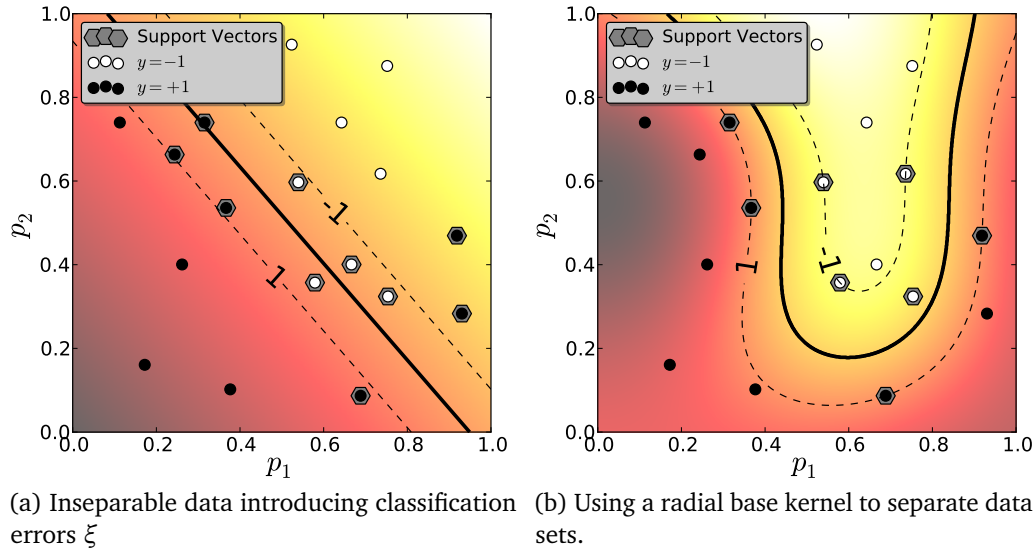


Figure 6.6: Comparison of the use of a linear and radial base kernel for the same set of data that is not linearly separable

### 6.3.5 Dealing with the discontinuous constraint boundary

When dealing with constraints, the gradient based optimiser such as SQP (Appendix A), requires gradient information from a constraint function when the constraints are violated. This gradient information allows it to move out of the constrained region. However, since SVC is used with the goal of classifying a feasible/non-feasible region, the only information given is the classification of the region in which a design point is evaluated. This means that once the SQP optimiser violates the confidence region constraint, no gradient information is given by the SVC. The lack of gradient information means that the SQP algorithm cannot minimise the constraint violation. A simple proposition to overcome this problem is to use the decision function in Eq. 6.21 to produce this information as follows:

$$f(\mathbf{p}) = \sum_{i=1}^l \varrho_i y_i k(\mathbf{p}_i, \mathbf{p}) + b_{svc} \quad (6.24)$$

Typically, the optimiser takes finite difference steps to calculate the gradient. This provides a solution for small constraint violations because of the clear gradient information at the classification boundary shown in Fig. 6.6b. Let us define the region in which the ROM is accurate as the region classified as +1, which corresponds to the white dots and lighter region in Fig. 6.6b. Once a design point is evaluated on the other side of the separating contour line, the SVC returns -1 as a classification value, which signifies a constraint violation. Once the constraint is violated, the function evaluation of the constraint will return a -1 to indicate a violation of the constraint. In order to enable the optimiser to

calculate gradient information to move out of the violated region, we recommend that Eq. 6.24 be used, since the function is smooth and continuous, satisfactory gradient information is given for small constraint violations. Using SVC, we turn our attention to the procedure involved in creating a locally accurate ROM.

### 6.3.6 Implementation of the SVC-POD

The use of SVC can be simply adapted to classifying the parameter space into accurate and inaccurate regions. In this section an alternative approach to TRPOD is developed, in which a confidence region<sup>1</sup> is created using support vector classification to classify a region in which the accuracy of the ROM is acceptable. This method is called SVC-POD, in which a confidence region is defined using SVC, instead of a trust radius used by the TRPOD method. The confidence region is updated similarly to the approach used for TRPOD (Algorithm 3), where the ROM is iteratively updated once the optimiser reaches the bounds of a confidence region.

The SVC-POD method has a number of advantages over TRPOD. Most significantly, the need for high-fidelity simulations during optimisation is eliminated. Instead, the ROM is built from a subset of responses taken from high-fidelity simulations performed, prior to the optimisation, at a set of DOE's across the parameter space. Another advantage of the SVC-POD method is that the SVC confidence region demarcates the parameter region in which the ROM has an acceptable accuracy. By accurately classifying the confidence region with SVC, the iterative adaptation to the trust radius that is necessary in the TRPOD method is avoided.

The SVC-POD is used in conjunction with an optimisation routine, where the optimiser moves from some initial point until some optima is found. In using a locally accurate methodology, the optimiser will be called iteratively, with each iteration finding an optimum within the region defined by the locally accurate model, as illustrated in Figure 6.7.

In this SVC-POD approach, which is concisely defined in Algorithm 5, we propose that a DOE be done over the entire design space, and that a high-fidelity simulation be done at each point in the DOE. The locally accurate ROM is then constructed from a selected number of DOE responses. The selected responses are selected in the proximity of the initial point in the parameter space in order to ensure that local system variations are captured. After creating the ROM, the accuracy of the ROM is determined at each point in the DOE, by comparing the ROM response to the high-fidelity response. Typically, the ROM will have a low error at the training points, and a higher error at the remaining points.

Using the RRMS error of the ROM at each DOE point, it is possible to classify regions of acceptable, and unacceptable, regions of accuracy for the optimiser to move in. All RRMS errors below a certain error tolerance are defined as

---

<sup>1</sup>The name 'confidence region' is used in place of 'trust region' to differentiate the TRPOD and SVC-POD approaches



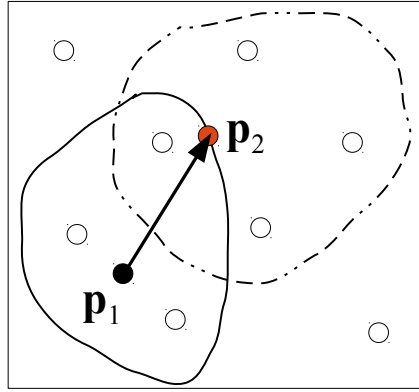


Figure 6.7: Example in which a starting point  $\mathbf{p}_1$  has a SVC boundary defined as  $\rho_1^{SVC}$ . Within the boundary, an optimum  $\mathbf{p}_2$  is found. The ROM is then recreated at  $\mathbf{p}_2$ , and a new confidence region  $\rho_2^{SVC}$  is defined.

acceptable, and the rest are defined as unacceptable. Using SVC, the two zones are separated. Now the boundary of the acceptable region is used as a constraint for the optimiser, and a new ROM is created once the optimiser reaches a SVC boundary. The new ROM again uses the responses generated at a selected number of DOE's in its proximity. Again, the RRMS error is calculated at each point and the SVC boundary is redrawn. Consequently, the SVC-POD method differs to the TRPOD method in that the model boundary is defined by the ROM accuracy (using Eq. 3.40), as opposed to the ratio of descent in the TRPOD method (Eq. 6.4).

The feasible portion of the parameter space is defined as the region bounded by the SVC decision function, explained in section 6.3.2, The optimisation problem is given as:

$$\text{Minimise } F_k^M(\mathbf{p}_k + \mathbf{s}_k) \quad (6.25)$$

$$\text{Subject to } \begin{cases} \mathbf{p}_i^l \leq \mathbf{p}_i \leq \mathbf{p}_i^u & i = 1, \dots, n_{par} \\ (\mathbf{p}_k + \mathbf{s}_k) \in \rho_k^{SVC} \end{cases} \quad (6.26)$$

where  $n_{par}$  is the number of parameter values in the parameter vector,  $F_k^M$  represents the objective function derived from the reduced order model at the  $k^{th}$  iteration. The optimisation problem is subject to the constraint that the problem lies within the confidence region determined at the  $k^{th}$  iteration,  $\rho_k^{SVC}$ , and that the optimum lies between upper and lower limits, denoted by  $\mathbf{p}_i^l$  and  $\mathbf{p}_i^u$  respectively. The goal here is similar to the TRPOD optimisation problem, where the optimiser seeks to minimise the function  $F$  by varying the parameters contained in  $\mathbf{p}$  and the response derived from the ROM. The optimiser is allowed to move within the limits imposed by the SVC boundary. This method allows both problems mentioned in the introduction to this chapter to be avoided, namely non-constructive mode-shapes and over-sized data sets. A comprehensive implementation of this method is given in the following chapter.

**Algorithm 5** Implementation of the SVC-POD method

- 
- 1: **DOE:** Create a LH DOE (Appendix A.4) and run a series of high-fidelity simulations
  - 2: **Initialise:** Select starting point for optimisation as  $\mathbf{p}_{start}$  from the high-fidelity response that has the smallest  $F^M$
  - 3: **Create ROM:** Select the  $n_{close}$  closest points to  $\mathbf{p}_{start}$  in the DOE. These  $n_{close}$  points are used as training points to create a LGP ROM (Algorithm 2) using  $m^h$  global shape functions.
  - 4: **Ensure accuracy:** Evaluate the RRMS error at each training point. If the RRMS at any training point is greater than a specified tolerance, augment the basis set  $m^h = m^h + 1$ , and go to step 3
  - 5: **Global evaluation:** Evaluate the RRMS error at all remaining points in the DOE using the ROM.
  - 6: **Classify points:** For points in which the  $RRMS < tol$ , set the corresponding point in the classification vector  $y_i = -1$ . All other points are set to 1 (Section 6.3.2), and the parameter space is classified into accurate and inaccurate regions using SVC
  - 7: **Search direction:** Find the optimal parameter change  $\mathbf{s}_k$  to satisfy Eq. 6.25, so that  $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{s}_k$
  - 8: **if**  $\|\mathbf{p}_k - \mathbf{p}_{k+1}\| < \delta_{terminate}$  **then**  $\mathbf{p}_k$  is the final optimum. Iterations end
  - 9: **else** Start new iteration from step 3, and setting  $\mathbf{p}_{start} = \mathbf{p}_{k+1}$
  - 10: **end if**
- 

## 6.4 Conclusion

In this chapter, two methods of using locally accurate models in inverse modelling and optimisation studies are introduced. The TRPOD method is a method with a proven convergence, but requires high-fidelity simulations during optimisation. With TRPOD, the parameter region that is accurately approximated by the ROM is denoted by a trust radius around a point.

In an attempt to eliminate the necessity of high-fidelity simulations during the optimisation, while at the same time identifying the parameter region in which the ROM yields a low error approximation, the SVC-POD method is introduced. While the SVC-POD method does not have a proven convergence like the TRPOD (Fahl (2000)), and needs to work in a predefined parameter space, it has the aforementioned advantages over the TRPOD method. In order to evaluate these two methods, the methods are demonstrated on different test case studies in the following chapter.

## Chapter 7

# Optimisation case studies

**T**HIS chapter focuses on the application of the Support Vector Classification Proper Orthogonal Decomposition (SVC-POD) and Trust Region POD (TR-POD) methods as local updating approaches to optimisation problems. In order to gain insight into the two methods, they are both applied to the same 2D problem presented in section 4.2. However, in this case the Reduced Order Model (ROM) seeks to approximate the high-fidelity response across the design space, instead of simply approximating one high fidelity response at one parameter point. In these examples, the Linearised Galerkin POD (LGP) method (section 5.2) is used to approximate the true response across the parameters space.

Two optimisation studies are undertaken. The first problem seeks to maximise the extraction of water using two sinks from the ground with a constraint on the level of the water table. This problem is demonstrated using both the SVC-POD and TRPOD methods. In the application of these two methods, some advantages and disadvantages of each method are highlighted.

The second problem is an inverse-modelling problem in which we seek to vary the soil and infiltration parameters until the ROM response matches or approximates some known response. The inverse problem is a problem of calibrating 5 parameters, of which 4 describe the soil type. In this section the interdependence between the parameters is briefly investigated.

Due to the intrusive nature of the LGP method, the ROM set-up time is still significant, but only requires a fraction of the time of a high-fidelity simulation. The case study on which the two locally accurate techniques are applied is developed in the following section.

### 7.1 Problem description

The case study considers a vertical cross section of soil subject to different soil parameters and boundary values, shown in Fig. 7.1. A Neumann boundary condition on the upper slope  $\Gamma_{nr}$  considers infiltration as an input. Two other Neumann boundaries  $\Gamma_{nw}$  act as sinks, allowing for removing of water from the system. Lastly, two Dirichlet boundaries are applied, approximating streams at each  $\Gamma_d$

positioned on the right and left sides of the domain. The problem is defined as follows:

$$\begin{aligned}
 C(h) \frac{\partial h}{\partial t} &= \nabla \cdot [K(h) \nabla (h + z)] \\
 f_n(t) &= 0.01 + 0.02 \sin(6\pi t / t_{nts}) \text{ cm/hr on } \Gamma_{nr} \\
 f_n(t) &= -p1 \text{ cm/hr on } \Gamma_{nw1} \\
 f_n(t) &= -p2 \text{ cm/hr on } \Gamma_{nw2} \\
 h(x, z, t) &= (200 - z) \text{ cm on } \Gamma_{d1} \text{ and } \Gamma_{d2} \\
 h(x, z, 0) &= h_{init} \text{ on } \Omega
 \end{aligned} \tag{7.1}$$

where  $f_n$  is the Neumann boundary condition used in the weak formulation in Eq. 2.49. In the direct optimisation problems we maximise the pumping rates with the soil parameters known, while in the inverse problem the pumping rates are known and the soil parameters are sought.

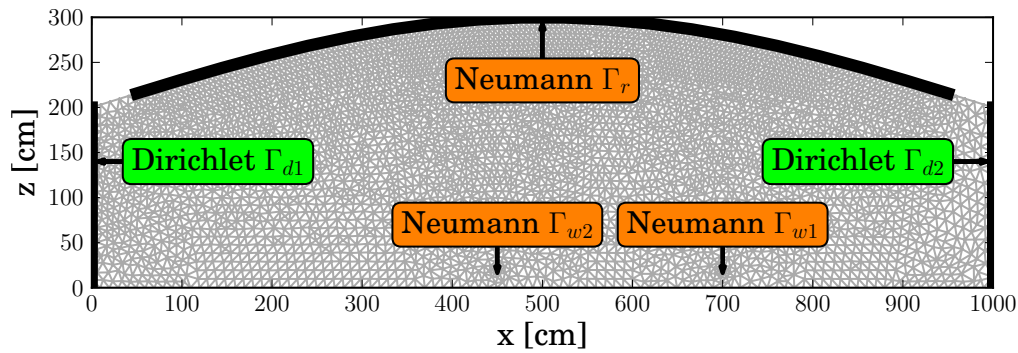


Figure 7.1: Mesh and location of boundary conditions of case study

In the following sections the TRPOD and the SVC-POD methods are applied to an optimisation problem based on this example. Both methods use a set of training points to create a locally accurate metamodel. The principal differences in these two methods lie in the creation of a confidence region, and updating criteria.

## 7.2 Example: Direct optimisation with SVC-POD

In this section we apply the SVC-POD method to a direct optimisation problem. The objective of this optimisation problem is to maximise the extraction rates

$\mathbf{p} = [p_1, p_2]$ , subject to three constraints:

$$\begin{aligned} \text{Maximise :} \quad & F(\mathbf{p}) = p_1 + p_2 \\ \text{Subject to :} \quad & 0 \leq p_i \leq 1, \quad i = 1, 2 \\ & h_{wt} \geq 100 \text{ cm} \\ & (\mathbf{p}_k + \mathbf{s}_k) \in \rho_k^{SVC} \end{aligned} \quad (7.2)$$

where the first constraint is a bound on the design parameters  $\mathbf{p}$ , the second constraint  $h_{wt} \geq 100$  cm requires the water table to be kept above 100 cm, and the third constraint is a sub-problem requiring the sum of the pumping rate ( $\mathbf{p}_k$  and movement  $\mathbf{s}_k$ ) to lie within the confidence region defined  $\rho_k^{SVC}$  defined with SVC.

The SVC-POD technique iteratively creates a locally accurate ROM, using high-fidelity responses obtained at a subset of the points defined by a Design of Experiments (DOE), as explained in Algorithm 5. In this example, a DOE of 16 points is created using a Latin Hypercube space filling algorithm (see Bates *et al.* (2004)), where the distribution across the design space is shown in Fig. 7.2. The two axes represent the extraction rates of the two sinks.

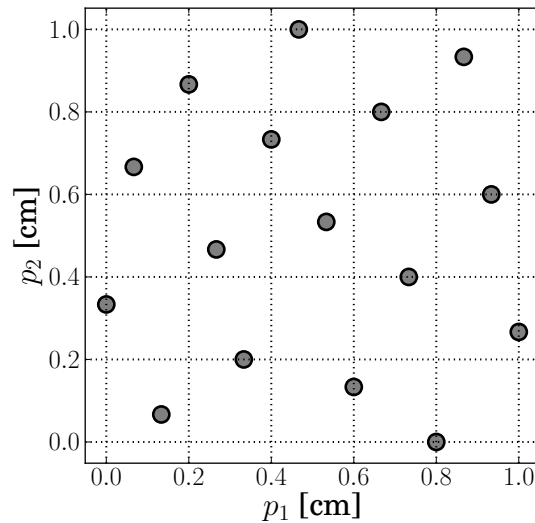


Figure 7.2: The two-parameter Design of Experiment points across the design space, using a latin hypercube generated in DOT Vanderplaats (2001)

### 7.2.1 Extrapolation with SVC-POD

To demonstrate the use of the SVC-POD as a metamodel with the potential to create a bounded extrapolation beyond the training points used to create the ROM (shown in chapter 6), let us choose the 4 points in proximity to the vector  $\mathbf{p} = [1, 0]$  in Fig. 7.2. Using these 4 points, we create a reduced model according

to Algorithm 2. The 4 design points have an error of  $RRMS \leq 1\%$  when  $m^h = 20$  mode shapes are used in the SVC-POD model. Subsequently, the accuracy of the ROM is evaluated at all the other design points.

Finally, all points with  $RRMS \leq 1\%$  are classified as  $-1$ , while all points with  $RRMS > 1\%$  are classified as  $+1$ . Using these classified points, a region of accuracy is defined with SVC. In Fig. 7.3a we show the training points, as well as the classification of each point in the DOE. The contour with value 0 defines the classification line between the two regions, while the contours at  $-1$  and  $+1$  show the contours of the decision function from Eq. 6.21 used to separate the two regions. The software *Sklearn* by Pedregosa *et al.* (2011) is used to demarcate the two regions.

The bar chart in Fig. 7.3b shows the magnitude of the error at each design point. The response of one design point outside of the training set is seen to have the  $RRMS < 1\%$ . The confidence region, shown in Fig. 7.3c, is seen to extrapolate beyond the initial training points. Should the tolerance on the LGP error be relaxed to  $RRMS \leq 3\%$ , the constrained region will extend further into the design space as shown in Fig. 7.3. In this case roughly half of the design space is included in the confidence region.

## 7.2.2 Optimisation with SVC-POD

In the optimisation problem defined in Eq. 7.2 the goal is to maximise the pumping rates while keeping the water table above 100 cm. The constraint  $\rho_k^{SVC}$  is the boundary imposed by SVC, effectively allowing the optimiser to move in the region defined by the SVC. In this case the SVC is defined as the region in which the DOE points have an error smaller than some tolerance, which is set to 1% in this example.

Following the method described in Algorithm 2, the high-fidelity responses at a subset of a predefined DOE are evaluated. We give a starting point as  $\mathbf{p}_0 = [0, 0]$  cm/hr. A subset of size  $n_{close} = 4$  is chosen. The high-fidelity responses to the  $n_{close}$  design points nearest to  $\mathbf{p}_0$  are used to build a LGP ROM. The ROM is then used to evaluate the training points in Fig. 7.4 (shown as hexagons). Should the ROM not have a  $RRMS \leq 1\%$ , the model should be enriched until the required tolerance is met at the training points. Once the ROM is sufficiently accurate at the training points, the error of the reduced model is evaluated at all design points. A high-accuracy region ( $RRMS > 1\%$ ) and a low accuracy region with  $RRMS < 1\%$  is defined using SVC. The feasible region  $\rho_k^{SVC} < 1\%$  is shown in Fig. 7.4.

Once a locally accurate ROM has been created and a confidence region has been defined, the local optimum movement  $\mathbf{s}_k$  is found to satisfy to Eq. 7.2. The optimiser used is the SQP optimiser implemented by Vanderplaats (2001). In this example, the ROM does not accurately approximate the points that do not form part of the training set, and so no bounded extrapolation takes place. The interpolation between points is still considered to be accurate however. In the

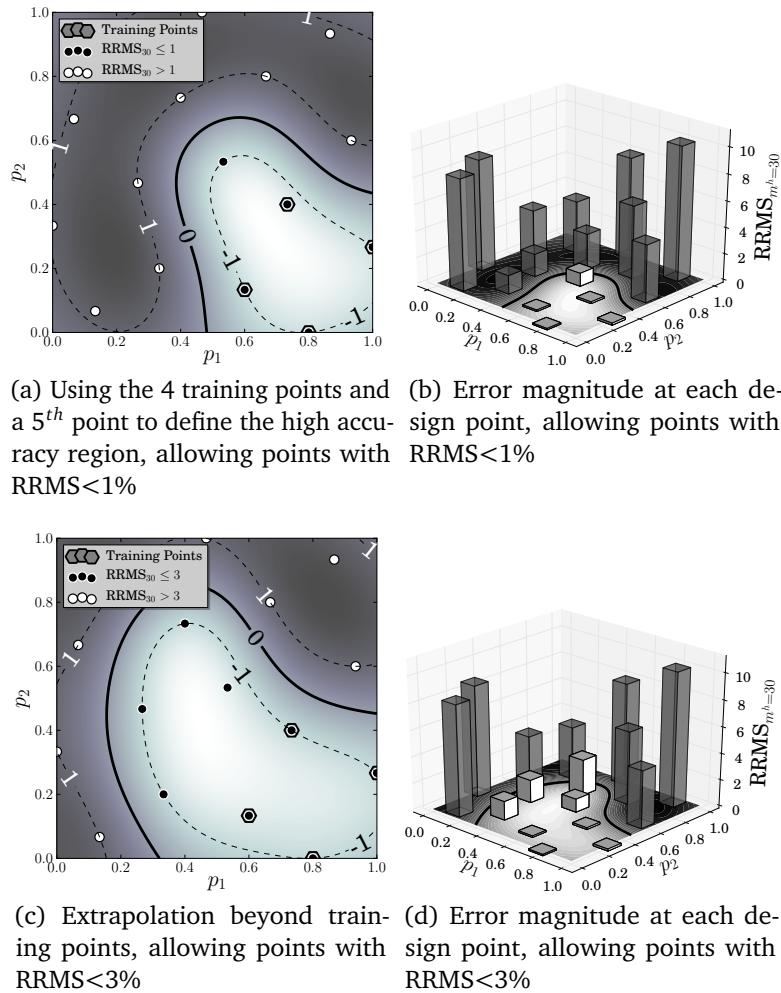


Figure 7.3: Classification of the regions of high and low accuracy

optimisation, the optimiser moves from the initial point to the first boundary as shown in Fig. 7.4a. Once the boundary is reached, a new locally accurate ROM and constraint boundary is created. This is repeated iteratively until the optimum is reached (step 8 in Alg. 5), as shown in Fig. 7.4d.

In this instance, 6 iterative updates are required to reach the optimum when using  $m^h = 20$  as tabulated in Table 7.1. The average time per iteration is  $t_{iter} \approx 90$  s, with each ROM call requiring  $t_{opt}/n_{opt} \approx 2$  s and each ROM requires  $\approx 50$  s to be created. A high-fidelity response is created for verification purposes, and the RRMS error of the ROM at the optimum is found to be 0.357%. The difference between the high-fidelity and ROM responses at the final time step are shown in Fig. 7.5 for a qualitative comparison. The error on the graph is seen to be very small ( $\leq 0.3$  cm) when compared to the magnitudes of head pressure in the range of  $-50 \text{ cm} < h < 200 \text{ cm}$ .

The number of simulations per iteration  $n_{calls}$  are also shown in Table 7.1, where the function calls per iteration range between 7 and 21. The low number

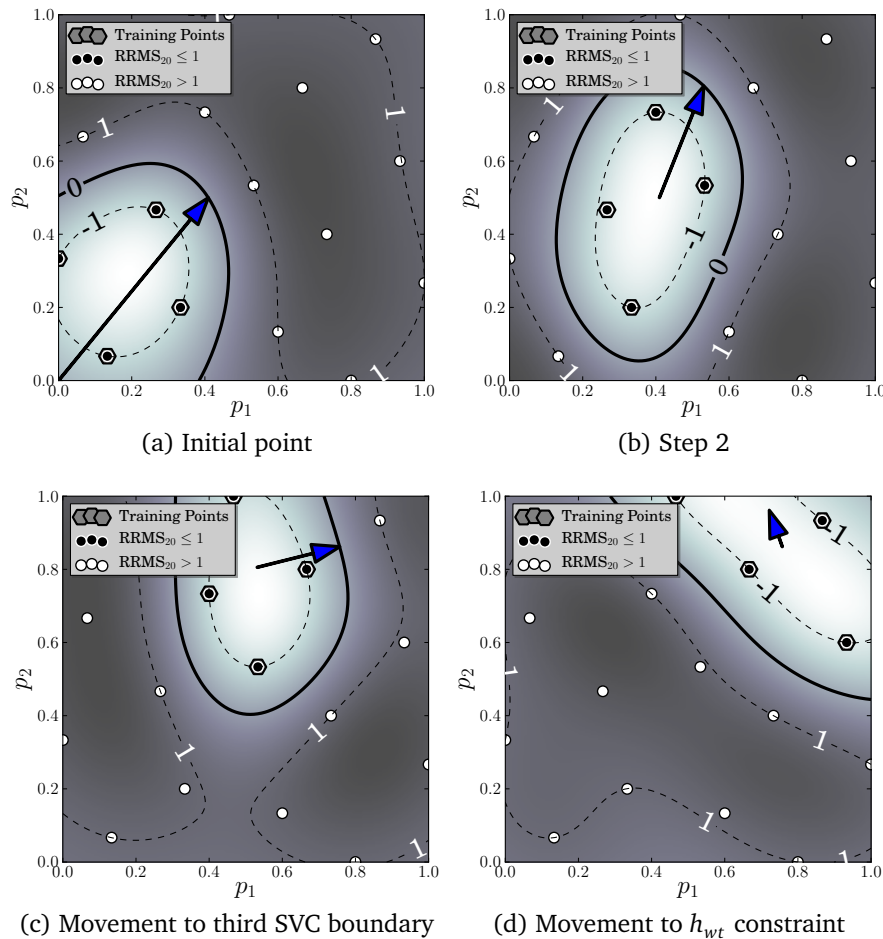


Figure 7.4: Example showing the iterative updating of the confidence region as the optimiser moves through the design space

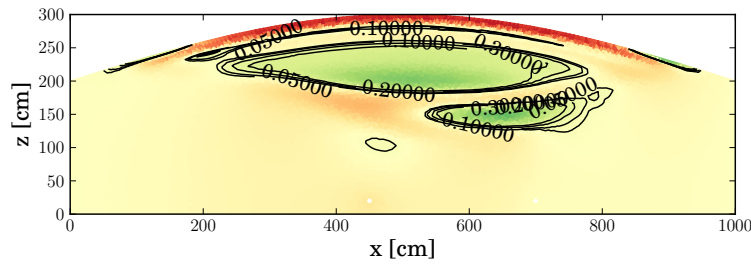
of function calls can be attributed in part to the low dimension of the parameter space. In addition, a criterion is added to the optimiser, where the optimiser is required to terminate the iteration once it moves within a specified distance of the constraint boundary, where the boundary defines the region in which the ROM has a sufficient accuracy. Terminating an iteration like this eliminates the need for the optimiser to search along a constraint boundary, reducing the number of function calls significantly.

In this example it is not necessary to enrich the basis set, since using  $m^h = 20$  gave an  $RRMS_{20} \leq 1\%$  at each of the training points. To use fewer iterations, the tolerance on the SVC boundary can be relaxed or more global shape functions can be used to create the ROM. The application of SVC-POD is seen to efficiently advance on the optimum, while iteratively creating locally accurate ROM's. The same example is now solved using the TRPOD method.



Table 7.1: SVC-POD results using 20 global shape functions

$k$	$p_1^*$	$p_2^*$	$n_{\text{calls}}$	$t_{\text{opt}}[\text{s}]$	$t_{\text{iter}}[\text{s}]$	$F(\mathbf{p}_k)$	$n_{TP}$	$h_{wt}$
0	0.41	0.5	14	27.43	87.25	0.91	4	159.09
1	0.53	0.81	12	24.76	85.59	1.3	4	145.91
2	0.76	0.86	7	14.59	79.15	1.34	4	103.79
3	0.72	0.96	21	37.45	105.27	1.68	4	100.35
4	0.73	0.95	21	38.09	101.53	1.68	4	100.33
5	0.72	0.96	12	21.12	79.75	1.68	4	100.34

Figure 7.5: Absolute error  $|\mathbf{h}_{nts} - \hat{\mathbf{h}}_{nts}|$  at the final time step

### 7.3 Example: Direct optimisation with TRPOD

In this section the TRPOD is applied to the numerical case study presented in section 7.1, as defined in the Algorithm 4. We redefine the water extraction optimisation problem as follows:

$$\begin{aligned}
 \text{Maximise :} & \quad F(\mathbf{p}) = p_1 + p_2 \\
 \text{Subject to :} & \quad 0 \leq p_i \leq 1, \quad i = 1, 2 \\
 & \quad h_{wt} \geq 100 \text{ cm} \\
 & \quad \|\mathbf{s}_k\| \leq \delta_k
 \end{aligned} \tag{7.3}$$

Here the optimiser iteratively solves a sub-problem where it is allowed to move a radius of  $\delta_k$  from a given centre  $\mathbf{p}_k$  at the  $k^{\text{th}}$  iteration. Some necessary adaptations to the optimisation problem are discussed in this section.

#### 7.3.1 Adaptation of the TRPOD method

The TRPOD approach needs to be modified to fit the optimisation problem, since the extraction rates  $\mathbf{p} = [p_1, p_2]$  are design parameters and not a system response. The TRPOD requires some true response  $F$  and some corresponding approximate response  $F^M$  to determine the change in the trust region. In this problem we evaluate the ratio  $\rho$  by considering the lowest level of the water table  $h_{wt}$  as

an indicator of our model accuracy, where  $h_{wt}$  denotes the lowest point in the water table at the final time step. The value of  $h_{wt}$  for the high-fidelity model is denoted as  $h_{wt}$  and the reduced model as  $h_{wt}^M$ . Now the reduced model accuracy is calculated as:

$$\rho_k = \frac{h_{wt}(\mathbf{p}_k) - h_{wt}(\mathbf{p}_k + \mathbf{s}_k)}{h_{wt}^M(\mathbf{p}_k) - h_{wt}^M(\mathbf{p}_k + \mathbf{s}_k)} \quad (7.4)$$

where  $\mathbf{s}_k$  is the change in pumping rates so that the optimal pumping rate is  $\mathbf{p}_k + \mathbf{s}_k$  after the  $k^{th}$  iteration. The first ROM is created with the response at one design point, shown in Fig. 7.6a. The initial trust region is set to  $\delta_0 = 0.15$ , and the number of eigenvectors used by the ROM is  $m^h = 20$ . The optimiser is called using the LGP as the ROM. At each  $k^{th}$  optimum, the ROM response is stored and compared to the high-fidelity response at that point. Once the ROM needs to be updated, snapshots from the high-fidelity response at the previous design point are used to create the next ROM, according to algorithm 4.

### 7.3.2 TRPOD results

The first ROM is constructed using the response at only one design point. The first optimum is found at the edge of the  $\delta_0$ , as shown in Fig 7.6a. The subsequent movement of the optimiser through the parameter space is shown in Fig.'s 7.6b to 7.6c. Each circle represents the extent of the trust region at  $k^{th}$  iteration. The lighter circles represent previous trust regions. The different trust regions  $\delta_i$  are adjusted according to the ratio defined in Eq. 7.4. Each circle represents the extent of the trust region at  $k^{th}$  iteration, and the lighter circles represent previous trust regions.

The results from each iteration are shown in Table 7.2. In the table, the time required to per iteration is given as  $t_{iter}$ , and the time used for the optimisation routine is denoted as  $t_{opt}$ . A total of 10 iterative updates of the trust region are required to get to the optimum, with a total time of  $\approx 25$  minutes. The high-fidelity simulation is seen to take  $\approx 80\%$  of the time in each iteration.

In Fig. 7.7  $h_{wt}^M$  is plotted against  $h_{wt}$ , showing the corresponding change in the trust radius. The trust region increases until the change in the ROM response is degraded, when it is updated according to algorithm 3. At the end of iteration 2 and 6, the ROM response diverges, resulting in a reduction of the trust radius and a new ROM. At the end of iteration 9 and 10, the ROM has a low accuracy, with  $h_{wt} < h_{wt}^M$ , indicating that the true model is violating the constraint of  $h_{wt}$ . At the final point, the evaluation of the ratio  $\rho_k$  does not indicate the need for an updated ROM, which permits a small error in the optimal point.

The termination criteria is chosen as  $\|\mathbf{p}_k - \mathbf{p}_{k-1}\| \leq 0.01$ , which is met at the end of the  $11^{th}$  iteration. Compared to the SVC-POD method, the TRPOD requires approximately the same number of ROM evaluation per iteration, but this example requires twice the number of iterations. The high number of iterations can be attributed to the ROM response diverging, resulting in a decrease in the

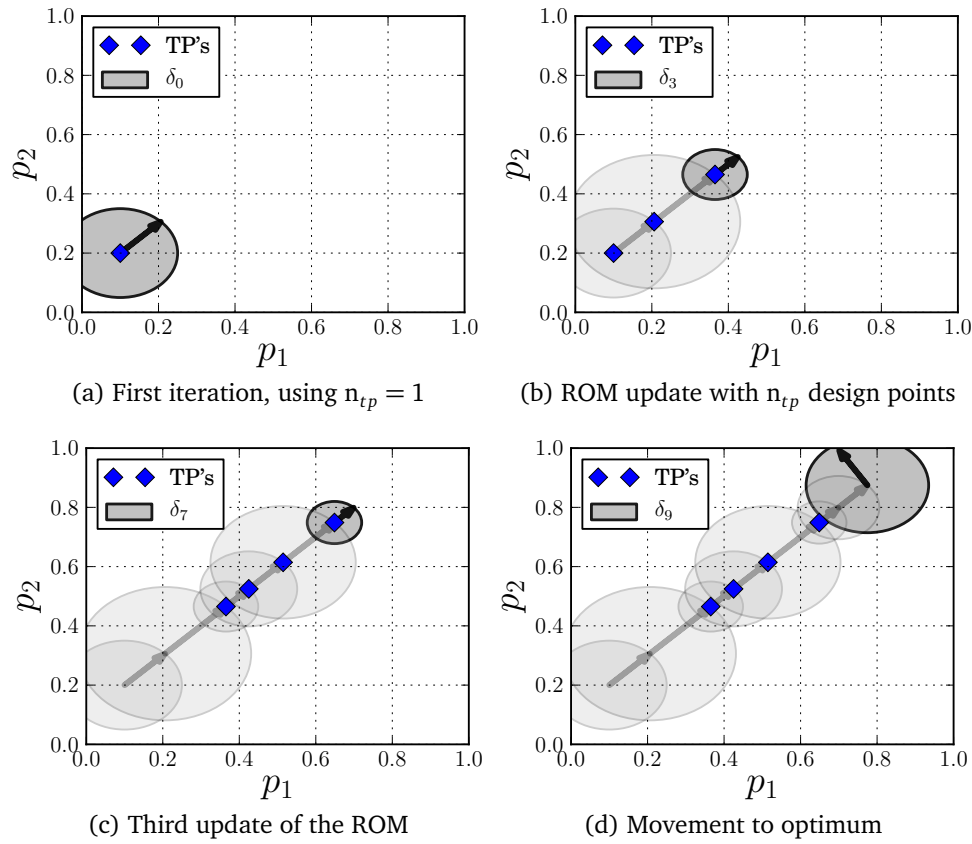


Figure 7.6: Movement of the trust region through the design space, showing the movements at iteration 1, 3, 7 and 9

Table 7.2: TRPOD results using 20 global shape functions

Iteration	$p_1^*$	$p_2^*$	$n_{calls}$	$t_{opt}[s]$	$t_{iter}[s]$	$\delta_{max}$	$h_{wt}$	$h_{wt}^M$
0	0.21	0.31	12	23.81	204.89	0.15	184.36	187.95
1	0.37	0.47	12	25.25	182.1	0.22	164.32	174.43
2	0.37	0.47	13	32.99	184.57	0.34	129.93	195.93
3	0.42	0.52	11	19.6	215.82	$8 \cdot 10^{-2}$	156.28	156.25
4	0.51	0.61	12	22.06	174.18	0.13	143.4	143.4
5	0.65	0.75	12	22.81	174.11	0.19	122.88	123.28
6	0.65	0.75	28	69.09	223.25	0.28	86.62	200
7	0.7	0.8	11	19.31	214.18	$7 \cdot 10^{-2}$	114.43	114.97
8	0.77	0.87	11	19.32	166.39	0.11	101.21	101.79
9	0.7	0.99	111	216.33	365.1	0.16	97.49	100.31
10	0.7	0.99	48	95.57	258.62	0.24	97.49	100.31

trust radius. In the following section, the SVC-POD is applied to an inverse problem.

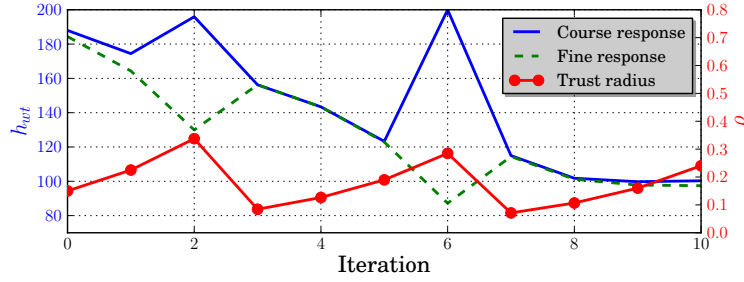


Figure 7.7: Water level approximations and corresponding adjustment in trust radius

## 7.4 Example: Inverse modelling with 5 design variables

With the SVC-POD method applied successfully on an optimisation problem in section 7.2, the method is now also applied to an inverse modelling problem. Only the SVC-POD approach is applied in this example, in order to explore the method in a higher dimensional parameter space. The inverse modelling is a problem with 5 unknown soil parameters, using the same geometry presented at the start of the chapter, in section 7.1. The unknown parameters are required to calculate  $C(h)$  and  $K(h)$ , defined in section 2.1.6 as well as the infiltration  $I$ , and so the vector of unknown parameters is defined as:

$$\mathbf{p} = [K_s, m, h_b, \varepsilon_s, I] \quad (7.5)$$

The residual saturation  $\varepsilon_r$  is assumed to be 0. The goal is to find the 5 unknown parameters by ‘matching’ a ROM response to a previously measured response by varying the parameter set  $\mathbf{p}$ . The optimisation problem is presented as follows:

$$\begin{aligned} \text{Min}_{\mathbf{p}} : & \quad \text{RRMS}(\mathbf{H}(\mathbf{p}^*, t), \hat{\mathbf{H}}(\mathbf{p}, t)) \\ \text{Subject to :} & \quad (\mathbf{p}_k + \mathbf{s}_k) \in \rho_k^{\text{SVC}} \\ & \quad \mathbf{p}^l \leq \mathbf{p} \leq \mathbf{p}^u \end{aligned} \quad (7.6)$$

where  $\mathbf{H}(\mathbf{p}^*, t)$  is a high-fidelity response calculated over a time period of  $t \in [0, 100]$  hr, where the optimal parameter set  $\mathbf{p}^*$  is given in Table 7.3 is predefined, and the SVC-POD is applied to find the optimum by varying  $\mathbf{p}$  until the RRMS is minimised. The first constraint is again a SVC confidence region  $\rho^{\text{SVC}}$ , as defined

in Alg. 5. The second constraint places upper and lower limits on the soil and infiltration parameters  $\mathbf{p}$ .

The parameters are chosen arbitrarily from the range of parameters given in Todd and Mays (2005). In the estimation of the parameters, the parameters are allowed to vary between a lower limit  $\mathbf{p}^l$  and an upper limit  $\mathbf{p}^u$  as shown in Table 7.3. In the table, the parameter  $\mathbf{p}^*$  is a chosen optimum that the SVC-POD must find by moving through the parameter space, until the ROM response evaluated at  $\mathbf{p}$  approximates the high-fidelity, or true, response at  $\mathbf{p}^*$ .

Table 7.3: The bounds of the design variables

	$\mathbf{p}^l$	$\mathbf{p}^u$	$\mathbf{p}^*$
$K_s$	0.2	0.4	0.3
$m$	0.2	0.4	0.3
$h_b$	20	60	30
$\varepsilon_s$	0.2	0.4	0.3
$I$	0.2	1	0.5

The Neumann boundary values in the problem Eq. 7.1 are assumed to be known in the example, which allowing us to focus on finding the soil parameters by inverse modelling. The sinks and infiltration Neumann boundary values are defined as:

$$\begin{aligned} f_n(t) &= I \times [0.03 + 0.06 \sin(6\pi t/t_{nts})] \text{cm/hr on } \Gamma_{nr} \\ f_n(t) &= -0.5 \text{cm/hr on } \Gamma_{nw1}, \Gamma_{nw2} \end{aligned} \quad (7.7)$$

In this optimisation problem, the LGP method is used as a ROM to evaluate the response  $\hat{\mathbf{H}}(\mathbf{p}, t)$ . Following the SVC-POD methodology in Alg. 5, we create a DOE again using 16 points to capture behaviour across the 5 dimensional design space. The small number is chosen deliberately to explore the limits of the extrapolation properties of the ROM.

In this example the initial point is taken as  $\mathbf{p}^l$  instead of at the high-fidelity response with the smallest RRMS error, in contrast to what is defined in Algorithm 2. The response at the nearest  $n_{close} = 3$  points in the DOE are used to create the first ROM, and the optimisation routine is started. The response at each iteration is shown in Table 7.4. The most noteworthy result here is that a large number of design points  $n_{SV}$  that fall within the  $RRMS < 1\%$  requirement. In the final iteration for example, 11/16 (= 68.75%) of the points are included in the confidence region  $\rho_3^{SVC}$  when using  $m^h = 20$  eigenvectors. This indicates very good extrapolation properties. It also indicates that the system ‘behaviour’ does not vary significantly when soil parameter values are changed, in contrast to the strong variation in behaviour seen in the example when Neumann boundary values are changed.

After 4 iterations, the SVC-POD finds the optimum  $\mathbf{p}^*$  after 4 iterations. During iteration 2, a very high number of function calls are executed. This can be attributed to the fact that the optimiser does not terminate against a constraint boundary, or that searching in a 5 dimensional space can require a very high number of function evaluations to find an optimum, but further adaptation of the optimisation approach should be done to limit excessive function calls. However, this short example shows that the SVC-POD can be applied very effectively to an inverse modelling problem, where some true response needs to be matched to a response derived from a ROM.

Table 7.4: Inverse optimisation results

Iteration	$K_s$	m	$h_g$	$\varepsilon_s$	I	$n_{\text{calls}}$	$t_{\text{opt}}[\text{s}]$	$t_{\text{total}}[\text{s}]$	RRMS	$n_{\text{SV}}$
0	0.25	0.31	33.8	0.25	0.54	27	41.58	85.94	46.81	4
1	0.29	0.31	35.16	0.27	0.55	30	43.17	88.35	11.14	7
2	0.3	0.3	29.96	0.3	0.5	306	429.55	475.13	0.47	10
3	0.3	0.3	29.96	0.3	0.5	27	46.07	89.58	0.67	11

## 7.5 Comparison between TRPOD and SVC-POD

Following the implementation of the TRPOD and SVC-POD approaches on the case studies, some comparisons can be done between the two methods. In the direct optimisation problem, it is seen that the TRPOD requires roughly 4 times the amount of time needed by the SVC-POD approach, but neglects the time required to set-up the SVC-POD high-fidelity responses prior to the optimisation. As expected, the SVC-POD approach is faster due to the time required to run high-fidelity simulations in the TRPOD method at each iteration.

In the inverse modelling problem, the good extrapolation properties of the SVC-POD method are highlighted by the large number of responses included in the SVC boundary. A total of 3 DOE responses are used, and the number of points included in the SVC boundary are between 4 and 11 of the 16. This allows the optimiser to move over a large percentage of the parameter space.

Both methods reach an optimum that is approximated by the ROM. However, only the TRPOD verifies the accuracy of the optimum at the end of each iteration, while the SVC-POD approach assumes a sufficient accuracy that is inferred from the training points, and the points used to create the confidence region. The final point is verified with a high-fidelity simulation to determine the final accuracy. In the SVC-POD approach, a number of validation points can be generated to ensure that the constraint boundary correctly defines the region of acceptable accuracy. In Table 7.5, a number of the principal differences between the two methods are explained.

Table 7.5: Discussion of differences between the TRPOD and SVC-POD methods

	TRPOD	SVC-POD
<b>Parameter domain</b>	The TRPOD can move through an unbounded parameter domain	The SVC-POD has a bounded domain.
<b>Design points</b>	Training points and high-fidelity responses are created during optimisation	Training points and high-fidelity responses are run prior to optimisation
<b>Optimisation constraints</b>	Each iteration is constrained by a spherical trust region, where the trust radius is determined by the accuracy of the previous response	Each iteration is constrained by the SVC boundary, where the boundary is constructed to include all sufficiently accurate points
<b>Time for optimisation</b>	The major time expense per iteration is the full model simulation, followed by the number of ROM function calls per iteration.	The major time expense is the number of ROM function calls per iteration.
<b>Optimum</b>	Finds a true optimum by verifying the sub-optima reached with the ROM at the end of each iteration	Finds an optimum approximated by the final ROM, where the ROM accuracy is inferred from neighbouring points.
<b>Overhead costs</b>	All overhead time costs, including high-fidelity simulations, are included in optimisation	The SVC-POD method has the overhead cost of having to find all high-fidelity responses prior to optimisation

## 7.6 Conclusion

In this chapter, the TRPOD and SVC-POD methods, explained in chapter 6, are applied to case studies to demonstrate the implementation and compare the advantages and disadvantages of each method. Both methods create locally accurate models, and the optimiser searches only in the region of the parameter space that is defined by the method. The use of locally accurate models is of particular importance to POD-based ROM's, where the quality of the ROM is dependent on the quality of the snapshots. Therefore, snapshots selected from a local region of the parameter space will yield a ROM that is more accurate in that local region, than if snapshots are used across the entire parameter space.

The TRPOD method significantly reduces the time that would be required by an optimiser using high-fidelity simulations, and it has a proved local convergence. However, the SVC-POD method is a very useful alternative to the TRPOD method, since it allows us to define a region of accuracy within which to search an

optimum, and allows us to eliminate the need for high-fidelity simulations during optimisation. The most significant compromise is that it does not have a proven convergence. However, in the examples studied in this chapter, the SVC-POD method finds the correct optima, and a significantly shorter time is required for the optimisation routine.

While the TRPOD allows the optimiser to explore freely through the parameter space, the SVC-POD approach is bounded by predefined limits on the design of experiments. However, the SVC-POD approach allows for extrapolation beyond the training points, with extrapolation permitted within bounds where the ROM accuracy can be verified.

The TRPOD and SVC-POD have been demonstrated on the optimisation of soil parameters and well extraction rates, but can be extended to parameters defining boundaries and even the geometry of the problem. However, for the purposes of this research report, both methods have been shown to be efficient, and can be used as tools in inverse modelling and optimisation studies.



# Chapter 8

## Conclusion

*I*N this thesis, the use of reduced order modelling is applied to the equations describing groundwater movement in saturated and unsaturated zones. The research is guided by a number of objectives.

### 8.1 Application of POD to groundwater models

The first objective was to develop a non-interpolative POD-based Reduced Order Model (ROM) on the Richards equation with attractive speed up times when compared to a high-fidelity model. The standard method for reduced modelling is the Galerkin Proper Orthogonal Decomposition (GPOD) method. This methodology is applied, but found to have little or no speed up times. An alternative method, called the Linearised GPOD (LGP) method, is developed. This method is demonstrated on a 2D example, and showed speed up times of  $\approx 70$  with a RRMS error  $< 1\%$  when compared to high-fidelity simulations in the single synthetic case study. It can be concluded that the objective is successfully met.

### 8.2 Extrapolation and optimisation with POD

A second objective was to investigate the extrapolation properties of the developed ROM and use it in optimisation and inverse modelling studies. The Trust-Region POD (TRPOD) method is an existing method that capitalises on the extrapolation properties within a bounded trust region. However, this method requires high-fidelity simulation during optimisation. To avoid this, a method called the SVC-POD is developed. This method iteratively defines a confidence region based on the accuracy of the LGP ROM at each DOE point, and subsequently defines the confidence region using Support Vector Classification (SVC).

The two methods were tested on a hypothetical unsaturated problem, in which the objective function strives to maximise extraction rates. The SVC-POD required half as many complete iterations when compared to the TRPOD method. This is due to the TRPOD method 'retreating' and restricting the trust region

when the ROM accuracy become too low. Furthermore, each TRPOD iteration required approximately twice as much time as the SVC-POD method due to the high-fidelity simulation being run at the end of each iteration. This comparison only considers the time required for optimisation, and excludes the time required to setup by the SVC-POD method to execute the full model at the initial training points.

The SVC-POD is subsequently demonstrated on a hypothetical inverse modelling problem, where the ROM interpolation properties, coupled with a verified confidence region, are shown to be very attractive. In the SVC confidence region,  $\approx 70\%$  of the design points are included in the confidence region when using only  $\approx 20\%$  of the design points to create the model. It is therefore concluded that the objective is met with a method that effectively exploits the good interpolation properties of the ROM approaches.

### 8.3 POD in coupled surface-groundwater problems

It is considered necessary to test the techniques developed on a true site. However, historic data on a suitable small-scale case study was not found, and so the Kogelberg site was used. The site is modelled as a saturated zone since it was considered too large to model with the Richards equation, and evapotranspiration properties are included in the numerical model.

A ROM was developed using the GPOD method, and speed up times of  $\approx 500$  (including overheads) were achieved. As in the 2D inverse modelling study, the ROM was found to have very good interpolation properties. In this case, the entire parameter domain was accurately approximated using the responses from 5 parameter sets to create a POD-based ROM with 40 eigenvectors.

The optimised parameters and recharge values are found to correspond well with those found in the literature. The results are of limited practical value, since the model was only calibrated at one site. However, the numerical and measured responses correspond well, and this can be considered as a preliminary study in the creation of a complete model approximation the entire Kogelberg site.

### 8.4 Local updating of POD models

With the high accuracy obtained using the LGP method in the hypothetical case studies, it can be concluded that this method supplies an efficient way of approximating non-linear PDE's. However, the added complexity means that the time for implementation is significantly longer than the GPOD method.

The use of explicit interpolation methods (kriging, RBF) are largely overlooked in this research, but have been shown to be able to create accurate POD-based ROM responses in the literature.

Although the method developed in this chapter are applied to the groundwater equations, these methods can be simply adapted to solve the large majority of PDE's that have been discretised with the Finite Element or other discretisation techniques.

## 8.5 Future work

In this research, not all possible ROM approaches could be investigated. However, some important points merit further research. One problem with the examples demonstrated is that the Richards equation can describe both the saturated and non-saturated zone. At the water-table, the non-linear functions are discontinuous, requiring a large number of eigenvectors to capture these functions since the water table moves in response to the boundary conditions. A way to circumvent this problem is having a separation between the two zones, and possibly adapting the mesh with the change in the water table.

The Greedy algorithm is a method to evaluate and upgrade a ROM across the entire design space. One of the steps in the algorithm is to evaluate the ROM residual  $R(\hat{h})$  to find the ROM accuracy at a different design point  $\mathbf{p}^*$ . This approach can be incorporated into the SVC-POD method, the residual error can be used to validate and refine the extents of the confidence region.

A next step in this research is to couple the Richards equation to the equations describing solute movement in a fluid. A ROM can then be created to approximate the water flow, and another ROM to calculate the subsequent solute convection and diffusion. The solute concentration of different elements is measured in many groundwater studies, and is an important value in inverse modelling studies.

Further future work includes the creation of a POD-based ROM that is based on different forms of the Richards' equation. One variation of the Richards' equation used in this research is known as the mixed formulation in which a system of equation for both the pressure head and the pressure dependent water content are solved. Another alternative formulation is the standard equation used in this thesis, but after applying a Kirchoff transform to it, effectively creating a partial linearisation of the equation.

Finally, the speed up times in the LGP method can be further improved with further optimisation of the research code. However, the current time gains and model errors are a good indication of what is possible, and are capable of significantly reducing computational time in optimisation studies.

# Appendices

# Appendix A

## Gradient based numerical optimisation

In this section we give a brief introduction into the concept of numerical optimisation, followed by a discussion of two gradient based optimisation techniques. The two techniques are Sequential Quadratic Programming (SQP) and the Modified Method of Feasible Direction (MMFD).

In this research document, the case studies are typically coupled to a reduced order model. Each reduced order model is dependent on the response at a set of training points. The choice of the location and quantity of these training points in the parameter space is known as a design of experiments. The use of the Latin Hypercube distribution is included at the end of this section to compliment the discussion on optimisation.

### A.1 Optimisation problem

A general approach to optimisation is introduced as follows:

$$\text{Minimise : } F(\mathbf{p}) \quad (\text{A.1})$$

$$\text{Subject to : } g_j(\mathbf{p}) \leq 0 \quad j = 1, \dots, m \quad (\text{A.2})$$

$$h_k(\mathbf{p}) = 0 \quad k = 1, \dots, l \quad (\text{A.3})$$

$$p_i^l \leq p_i \leq p_i^u \quad i = 1, \dots, n \quad (\text{A.4})$$

$$\text{where } \mathbf{p} = \{p_1, p_2, \dots, p_n\}' \quad (\text{A.5})$$

and  $\mathbf{p}$  represents the design variables. The approach states that the function  $F(\mathbf{p})$  needs to be minimised, subject to certain inequality constraints (Eq. A.2) and certain equality constraints (Eq. A.3). The side constraints in Eq. A.4 imply upper and lower limits that are imposed on the design variables, shown in Eq. A.5.

The use of a gradient based optimiser to find the optimum is typically a two step process. Firstly gradients are determined at some starting point, from which a search vector,  $\mathbf{s}$ , is derived. Once the search direction is found, a 1D search is

done in the search direction to find the minimum. The parameter set is updated according to the following equation:

$$\mathbf{p}_q = \mathbf{p}_{q-1} + \alpha^* \mathbf{s}_q \quad (\text{A.6})$$

where the term  $\alpha^* \mathbf{s}_q$  determines the optimal distance and direction from the previous design point  $\mathbf{p}_{q-1}$ . More detail can be found on the 1D search algorithm in Vanderplaats (2005). In the following section two techniques for finding the optimal search direction are discussed.

## A.2 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is a technique that finds the search direction  $\mathbf{s}$  by solving a quadratic sub-problem. In formulating the sub-problem for SQP the objective function is approximated by a quadratic Taylor series around some initial design point  $\mathbf{p}_0$  while the constraints are approximated by linear Taylor series:

$$\text{Minimise : } Q(\mathbf{s}) = F(\mathbf{p}_0) + \nabla F(\mathbf{p}_0)^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T \mathbf{B} \mathbf{s} \quad (\text{A.7})$$

$$\text{Subject to : } \begin{cases} \nabla g_j(\mathbf{p})^T \mathbf{s} + \delta_j g_j(\mathbf{p}) \leq 0 & j = 1, m \\ \nabla h_k(\mathbf{p})^T \mathbf{s} + \bar{\delta} h_k(\mathbf{p}) = 0 & k = 1, l \end{cases} \quad (\text{A.8})$$

The symbol  $\mathbf{B}$  represents a matrix that approximates the Hessian matrix (see Vanderplaats (2005)) at  $\mathbf{p}_0$ , and the two scalar parameters  $\bar{\delta}$  and  $\delta$  are used to regulate and prevent inconsistencies in the linearised constraints. The Hessian matrix is updated according to a Broydon-Fletcher-Goldfarb-Shanno update formula (see Vanderplaats (2005)). In the case of limited computer memory, the Fletcher-Reeves conjugate directions method (Vanderplaats (2001)) is used. Solving the sub-problem yields the search vector  $\mathbf{s}$ , and a 1D search can commence to find the minimum. The 1D search makes use of a penalty function to search along the search vector until a minimum is reached. This procedure is implemented iteratively until no further improvement can be made. The design point at which this minimum is found is evaluated and becomes the new point around which a linearisation is done. This procedure is repeated iteratively until a precise solution is reached, hence the term ‘sequential’.

## A.3 Modified Method of Feasible Directions (MMFD)

The objective in the MMFD technique is to find a direction that reduces the objective function as rapidly as possible without violating constraints. Once a constraint is encountered, the search direction is set tangent to the constraint boundary. For

a problem including only inequality constraints, the search direction is found by solving the following sub-problem (see Vanderplaats (2005)):

$$\text{Minimise : } \quad \nabla F(\mathbf{p})^T \mathbf{s} \quad (\text{A.9})$$

$$\text{Subject to : } \quad \begin{cases} \nabla g_j(\mathbf{p})^T \mathbf{s} \leq 0 & j \in J \\ \mathbf{s}^T \mathbf{s} \leq 0 \end{cases} \quad (\text{A.10})$$

where  $J$  is the number of active and violated constraints. A constraint is considered active when the current parameter point is at the constraint boundary. Should a constraint be violated during a 1D search, a Newton-step method is initiated to move back into the usable-feasible region. A search is followed to the point that no further improvement can be made in the objective function.

## A.4 Design of experiments

In the creation of metamodels (of which ROM's form a branch), a number of training points are required. The way in which these training points are chosen is known as a Design Of Experiments (DOE). The location and quantity of the DOE points can significantly affect the accuracy of the metamodel. In cases where no underlying model is assumed, it is generally advantageous to have a space filling DOE.

For a design of  $N$  design variables and  $M$  training points, the Latin Hypercube (LH) design (Bates *et al.* (2004)) divides each dimension into  $M$  equal levels. For each dimension, each level is occupied by only one point. Figure A.1a shows a typical implementation of a LH design in two dimensions. However, this design could potentially have very poor space filling characteristics, as shown in Fig. A.1b

The Optimum Latin Hypercube (OLH) design is an improved method of creating a space-filling DOE. The OLH essentially strives to maximise the minimum space between design points, while satisfying the LH requirements. In Fig. A.2 an example of an OLH design with 9 points is presented. Here the effective space filling design properties are evident. The disadvantage of the OLH is that it requires significantly more time to generate the design points.

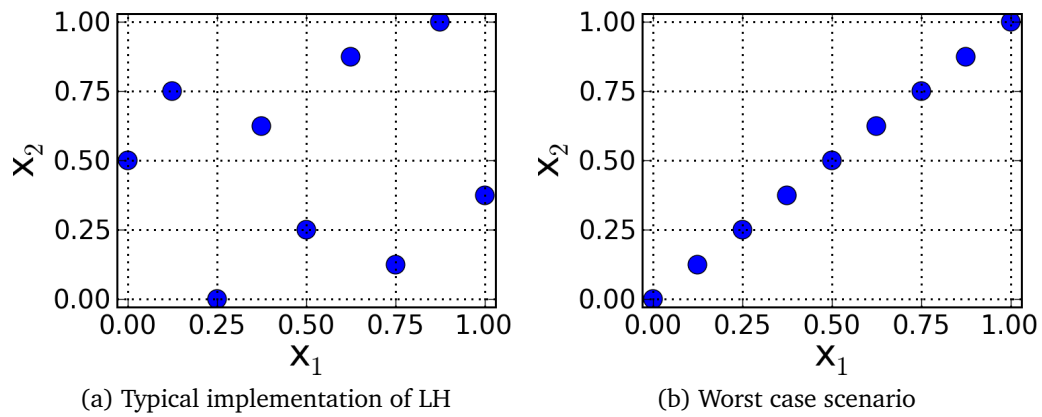


Figure A.1: Latin Hypercube designs using 9 design points

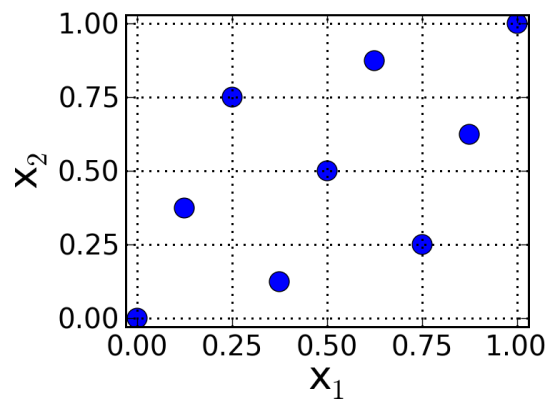


Figure A.2: Optimum Latin Hypercube design generated by VisualDOC (Vanderplaats (2001))



## Appendix B

### Code segments

```

//——Define triangular domain——//
fespace Vh(Th,P1);
Vh h,v, V;
... //Code excluded
// — Matrix — //
varf m(h,v) = int2d(Th)( Ch/dt h v
                      + Kh dx(h) dx(v) //  $\int_{\Omega} \nabla v \cdot K(h) \nabla v$ 
                      + on(Dirtop,h=bvt) // Soil surface — Dirichlet
                      + on(Dirbot,h=0); // Water table — Dirichlet
//——Vector——//
varf v1(h, v) = int2d(Th)( Ch/dt hn v — Kh dy(v))
                + on(Dirtop,h=bvt) // Soil surface — Dirichlet
                + on(Dirbot,h=0); // Water table — Dirichlet
// ——Iteration over time——//
for(int i=1;i<nts;i++){
  while(tol>0.01){
    ...
    M = m(Vh, Vh);
    V[] = v1(0,Vh);
    h[] = M^-1V[];
    ... //Code excluded
  }
}

```

Figure B.1: FreeFem++ code segment solving the Richards' equation

```
...
Uh = loadMAT(path+"Results/Uh.bb"); // Fetch eigenvector Uh
// ——Iteration over time——//
for(int i=1;i<nts;i++){
    while(tol>0.01){
        ...
        M = m1(Vh, Vh); // Create matrices
        Mhat = Uh' M; Mhat = Mhat Uh;
        V[] = m2(0,Vh);
        Vhat = Uh' V[]; // Reduce vectors
        alpha(:,i) = Mhat^-1Vhat;
        hhat[] = Uh alpha(:,i); // Final head
        ...
    }
}
```

Figure B.2: Implementation of the Galerkin-POD method in FreeFem++

## Appendix C

# Evapotranspiration

Evapotranspiration is the name given to the combined phenomena of evaporation and transpiration, and is an important boundary condition in recharge case studies. The Penman-Monteith (PM) method is the standard method recommended, especially for agricultural use. The method is used to *predict transpiration from a knowledge of available energy, temperature, specific humidity and aerodynamic and surface conductance* Allen (2000). In this section we briefly outline the parameters used in the PM method. For a complete approach the FAO manual (by Allen (2000)) must be consulted.

With the environmental factors listed above, the PM equation calculates the reference evapotranspiration (ET) of an unstressed hypothetical crop with assumed characteristics such as height and surface resistance. The complete equation to calculate the reference evapotranspiration ( $ET_o$ ) is:

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T + 273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0.34u_2)} \quad (C.1)$$

where the relevant parameters and units used are listed in Table C.1. Once the  $ET_o$  has been determined, the value is then calibrated to the crop under consideration. (See sections C.1 and C.2)

A significant amount of information is needed to calculate  $ET_o$  with the PM method. When all the climate data is not available, alternative methods are proposed to approximate  $ET_o$  in the FAO.

### C.1 Adjusting the $ET_o$ with a single crop factor

The  $ET_o$  gives information about the amount of evapotranspiration that is yielded by a hypothetical well-watered grass field. As such, it needs to be adjusted to match the specific crop type. This is done by multiplying  $ET_o$  by a crop factor  $K_c$ , shown in Fig. C.1. The crop factor is a comparison between the type of crop (or vegetation) under consideration and the reference well-watered grass field. The

Table C.1: Parameters in Penman-Monteith equation

Factor	Description	Units
$ET_o$	Reference evapotranspiration	[mm/day]
$R_n$	net radiation at the crop surface	[MJ m <sup>-2</sup> day <sup>-1</sup> ]
$G$	soil heat flux density	[MJ m <sup>-2</sup> day <sup>-1</sup> ]
$T$	air temperature at 2 m height	[° C]
$u_2$	wind speed at 2 m height	[m/s]
$e_s$	saturation vapour pressure	[kPa]
$e_a$	actual vapour pressure	[kPa]
$e_s - e_a$	saturation vapour pressure deficit	[kPa]
$\Delta$	slope vapour pressure curve	[kPa/°C]
$\gamma$	psychometric constant	[kPa/°C]

adjusted evapotranspiration is given as:

$$ET_c = K_c ET_o \quad (C.2)$$

The value of  $K_c$ , shown in Fig. C.2a typically varies between 0.1 and 1.2 depending on factors such as the vegetation growth stages and climatic conditions. An additional soil evapotranspiration reduction coefficient  $K_s$  considers environmental stress (such as limited water) and is used to adjust the  $ET_c$  value as follows:

$$ET_{c, adj} = K_s K_c ET_o \quad (C.3)$$

The  $K_s$  factor is plotted in Fig. C.2b. There is a linear decrease once the available water drops below the availability defined by as Readily Available Water (RAW), dropping to zero once the total available water (TAW) has been used up. While there is water readily available for evapotranspiration, there is no reduction in  $K_c$ . The TAW and RAW are dependent on the crop characteristics.

## C.2 Adjusting the $ET_o$ with dual crop factors

The true evapotranspiration can also be calculated by separating the factors to consider the basal crop coefficient  $K_{cb}$  and soil evaporation  $K_e$  as follows:

$$ET_c = (K_{cb} + K_e) ET_o \quad (C.4)$$

Using the dual factor approach, the soil evaporation under water stress is calculated as follows:

$$K_e = K_r K_{e, max} \quad (C.5)$$

where  $K_{e, max}$  is the maximum  $K_e$  value following a rainfall or irrigation event. The factor considering water stress,  $K_r$ , is introduced to limit evaporation when limited water is available at the surface. The factor decreases with a decrease

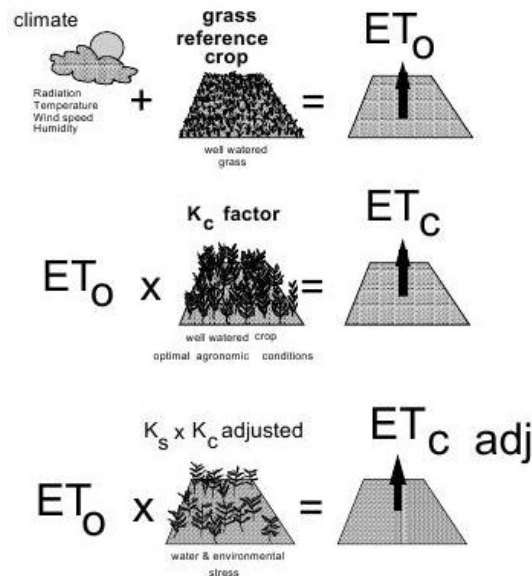
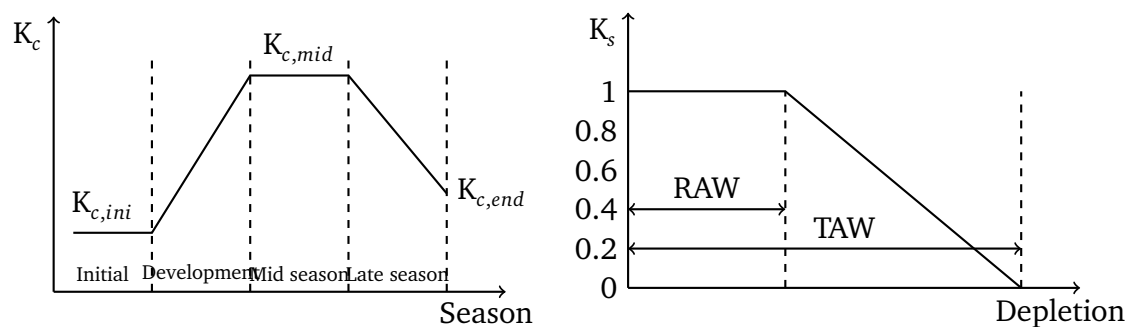


Figure C.1: Value of true evapotranspiration derived from  $K_c$  and  $K_s$  factors (Allen (2000))



(a) Typical crop factor during different growth phases

(b) Water stress coefficient Allen (2000)

Figure C.2: Crop factor and transpiration stress factor

in available surface water as shown in Fig. C.3. Similar to the evaluation of the evaporation coefficient  $K_s$ , the soil coefficient  $K_r$  decreases linearly to zero when the readily evaporable water (REW) has been used up. Once the total evaporable water (TEW) is used up,  $K_r$  becomes zero, thereby eliminating the soil evaporation coefficient.

The adjustment of the  $ET_0$  with the two stress factors is now calculated as:

$$ET_c = (K_s K_{cb} + K_e) ET_0 \quad (C.6)$$

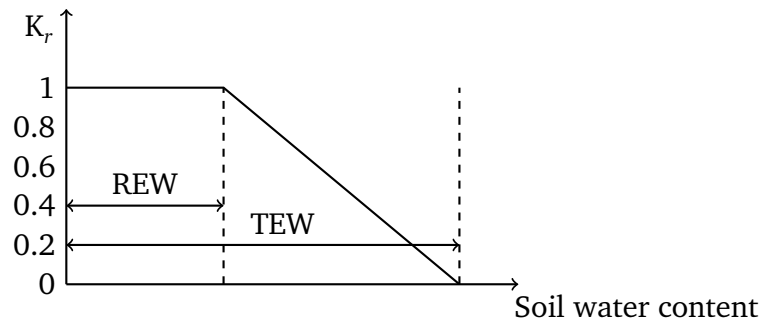


Figure C.3: Soil evaporation reduction coefficient

where  $K_e$  is calculated from Eq. C.5. In the case study in Chapter 4, an inverse estimation approach to find the evapotranspiration parameters on a study site is shown.

With the evapotranspiration parameters relevant to this research now defined, we move to the discretisation of the groundwater flow equations.

# List of References

- Alexandrov, N., Branch, M.D.O., Larc, N., Dennis, J.E., Michael, R., Icase, L. and Torczon, V. (1997). A Trust Region Framework for Managing the Use of Approximation Models in Optimization. Tech. Rep. 97, Institute for Computer Applications in Science and Engineering NASA Langley Research Center.
- Allen, R.D. (2000). F. A. O. Irrigation and Drainage Paper No. 56. *Crop Evapotranspiration (guidelines for computing crop water requirements)*.
- Arlen W. Harbaugh (2005). *MODFLOW-2005, The US Geological Survey Modular Ground-Water Model à the Ground-Water Flow Process MODFLOW-2005*.
- Astrid, P. (2004). *Reduction of Process Simulation Models: a proper orthogonal decomposition approach*. Ph.D. thesis, Technische Universiteit Eindhoven.
- Astrid, P., Weiland, S., Willcox, K. and Backx, T. (2008 November). Missing Point Estimation in Models Described by Proper Orthogonal Decomposition. *IEEE Transactions on Automatic Control*, vol. 53, no. 10, pp. 2237–2251. ISSN 0018-9286.
- Audouze, C., Vuyst, F.D. and Nair, P.B. (2009). Reduced-order modeling of parameterized PDEs using time-space-parameter principal component analysis. , no. October, pp. 1025–1057.
- Barrett, J.W. and Elliott, C.M. (1986). Finite Element Approximation of the Dirichlet Problem Using the Boundary Penalty Method. *Numerische Mathematik*, vol. 366, no. 49, pp. 343–366.
- Barrow, D. (2010). *Ground water Dependence of Ecological Sites Located in the Table Mountain Group*. Master's thesis, University of the Free State.
- Bates, S.J., Sienz, J. and Toropov, V.V. (2004). Formulation of the Optimal Latin Hypercube Design of Experiments Using a Permutation Genetic Algorithm. *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*.
- Bergmann, M. and Cordier, L. (2008 August). Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models. *Journal of Computational Physics*, vol. 227, no. 16, pp. 7813–7840. ISSN 00219991.
- Boulakia, M., Schenone, E. and Gerbeau, J.-F. (2011). Reduced-order modeling for cardiac electrophysiology. Application to parameter identification. , no. November. arXiv:1111.5926v2.

- Burkardt, J., Gunzburger, M. and Lee, H.-C. (2006 December). POD and CVT-based reduced-order modeling of Navier-Stokes flows. *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 1-3, pp. 337–355. ISSN 00457825.
- Celia, M.A. and Bouloutas, E.T. (1990). A General Mass-Conservative Numerical Solution for the Unsaturated Flow Equation. *Water Resources Research*, vol. 26, no. 7, pp. 1483–1496.
- Colvin, C., Riemann, K., Brown, C., Maitre, D.L., Mlisa, A., Blake, D., Aston, T., Maherry, A., Engelbrecht, J., Pemberton, C., Magoba, R., Soltau, L. and Prinsloo, E. (2009a). *Development in the Table Mountain Group ( TMG ) Aquifer System*. 1327. ISBN 9781770057968.
- Colvin, C., Riemann, K., C., B. and Le Maitre, D. (2009b). Ecological and Environmental Impacts of Large-scale Groundwater Development in the Table Mountain Group (TMG) Aquifer System. Tech. Rep., Water Research Commission.
- Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, vol. 297, no. 20, pp. 273–297.
- Couplet, M. (2003). Intermodal energy transfers in a proper orthogonal decomposition-Galerkin representation of a turbulent separated flow. *Journal of Fluid Mechanics*, vol. 491, pp. 275–284.
- Darcy, H. (1856). Les fontaines publiques de la ville de Dijon: : exposition et application des principes À suivre et des formules À employer dans les questions de distribution d'eau. Tech. Rep., Paris.
- De Vuyst, F. (2009). PDE Metamodeling using Principal Component Analysis, Multidisciplinary Design Optimization in Computational Mechanics. *Wiley ISTE*.
- Diersch, H.G. (2014). *Finite Element Modeling of Flow, Mass and Heat Transport in Porous and Fractured Media*. Springer.
- Doherty, J. (2010). PEST: Model-Independent Parameter Estimation. Tech. Rep., Watermark Numerical Computing.
- Fahl, M. (2000). *Trust-region Methods for Flow Control based on Reduced Order Modelling*. Ph.D. thesis, Universitat Trier.
- Gunes, H. and Cadirci, S. (2009). Modeling and Inverse Design of Convective Heat Transfer in a Grooved Channel Using Proper Orthogonal Decomposition. In: *ASME International Mechanical Engineering Congress & Exposition*, pp. 519–526. Asme, Florida. ISBN 978-0-7918-4382-6.
- Gunn, S.R. (1998). Support vector machines for classification and regression. Tech. Rep., University of Southampton, Faculty of Engineering, Science and Mathematics.
- Haasdonk, B. (2011). Convergence Rates of the POD-Greedy Method. *Stuttgart Research Centre for Simulation Technology*.



- Haasdonk, B. and Ohlberger, M. (2006). Reduced Basis Method for Finite Volume Approximations of Parametrized Evolution Equations. vol. 2, pp. 1–22.
- Hecht, F. (2012). New development in FreeFem++. *J. Numer. Math.*, vol. 20, no. 3-4, pp. 251–265. ISSN 1570-2820.
- Hiscock, K. (2005). *Hydrogeology: Principles and practice*. Blackwell.
- Jia, H. (2007). *Groundwater resource evaluation in Table Mountain Group aquifer systems*. Ph.D. thesis, University of the Western Cape.
- Jirka Šimůnek and Miroslav Šejna (2011). *Hydrus Technical Manual: Version 2*. Tech. Rep. January, PC-Progress, Prague.
- Jones, E., Oliphant, T., Peterson, P. *et al.* (2001). SciPy: Open source scientific tools for Python. [Online; accessed 2014-08-20]. Available at: <http://www.scipy.org/>
- Jovanovic, N.Z. and Bugan, R.D.H. (2010a). Deliverable 53: Reducing uncertainties of evapotranspiration and preferential flow in the estimation of groundwater recharge. *WRC Project No.K5/1909*.
- Jovanovic, N.Z. and Bugan, R.D.H. (2010b). Deliverable 54: Reducing uncertainties of evapotranspiration and preferential flow in the estimation of groundwater recharge. Tech. Rep. June.
- Jovanovic, N.Z. and Bugan, R.D.H. (2011). Deliverable 57: Preferential flow in the estimation of groundwater recharge. Tech. Rep., CSIR Natural Resources and the Environment, Stellenbosch.
- Kragel, B. (2005). *Streamline Diffusion POD Models in Optimization*. Ph.D. thesis, Trier University.
- Kresic, N. (2007). *Hydrogeology and Groundwater Modeling*. CRC Press, 2nd Edition.
- Kwon, Y.W. and Bang, H. (1996). *The Finite Element Method using MATLAB*. CRC Press.
- Liang, Y.C., Lin, W.Z., Lee, H.P., Lim, S.P. and Lee, K.H. (2002). Proper Orthogonal Decomposition and its applications part II : Model reduction for MEMS dynamical analysis. *Journal of Sound and Vibration*, vol. 256, pp. 515–532.
- Liberge, E. and Hamdouni, a. (2010 February). Reduced order modelling method via proper orthogonal decomposition (POD) for flow around an oscillating cylinder. *Journal of Fluids and Structures*, vol. 26, no. 2, pp. 292–311. ISSN 08899746.
- Lophaven, S.N., Nielsen, H.B. and Søndergaard, J. (2002). *Kriging Toolbox*. Tech. Rep., Technical University of Denmark.
- McPhee, J. and Yeh, W.W. (2008). Groundwater Management Using Model Reduction via Empirical Orthogonal Functions. *Journal of Water Resources Planning and Management*, , no. April, pp. 161–170.

- Oberbeck, A. (1879). Über die Wärmeleitung der Flüssigkeiten bei Berücksichtigung der Strömung infolge von Temperaturdifferenzen (on the thermal conduction of liquids with regard to flows due to temperature differences). *Ann. Phys. Chem.*, vol. 7, pp. 271–292.
- Ostrowski, Z., Bialecki, R.A. and Kassab, A.J. (2005). Advances in application of Proper Orthogonal Decomposition in inverse problems. In: *5th International Conference on Inverse Problems in Engineering: Theory and Practice*, July. Cambridge.
- Pechstein, C. (2012). *Finite and Boundary Element Tearing and Interconnecting Solvers for Multiscale Problems*. Springer.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830.
- Pinnau, R. (2008). *Model Order Reduction: Theory, Research Aspects and Applications, Mathematics in industry*, chap. Model Reduction via Proper Orthogonal Decomposition. Springer.
- Pope, S. (1997 January). Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation. *Combustion Theory and Modelling*, vol. 1, no. 1, pp. 41–63. ISSN 1364-7830.
- QGIS Development Team (2009). *QGIS Geographic Information System*. Open Source Geospatial Foundation.
- Rowley, C.W., Colonius, T. and Murray, R.M. (2004 February). Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, vol. 189, no. 1-2, pp. 115–129. ISSN 01672789.
- Schölkopf, B. (1997). *Support Vector Learning*. Ph.D. thesis, Technischen Universität Berlin.
- Segal, I.A. (2012). Finite element methods for the incompressible Navier-Stokes equations. Tech. Rep., Delft Institute of Applied Mathematics.
- Siade, A.J., Putti, M. and Yeh, W.W.-G. (2010 August). Snapshot selection for groundwater model reduction using proper orthogonal decomposition. *Water Resources Research*, vol. 46, no. 8, pp. 1–13. ISSN 0043-1397.
- Sirovich, L. (1987). Turbulence and the dynamics of coherent structures part i: coherent structures. *Quarterly of applied mathematics*, vol. XLV, no. 3, pp. 561–571.
- Todd, D.K. and Mays, L.W. (2005). Groundwater Hydrology. *John Wiley and Sons, Inc, 3rd Edition*.
- van Genuchten, M.T. (1978). Mass transport in Saturated-Unsaturated Porous Media: One-Dimensional Solutions. Tech. Rep., Water Resources Program, Department of Civil engineering, Princeton University, Princeton, NJ.

- Vanderplaats, G.N. (2001). *DOT Design Optimization Tools, USERS MANUAL*.
- Vanderplaats, G.N. (2005). *Numerical Optimization Techniques for Engineering Design*. 4th edn. Vanderplaats Research and Development, Inc.
- Vermeulen, P.T.M., Heemink, A.W. and Te Stroet, C.B.M. (2004 jan). Reduced models for linear groundwater flow models using empirical orthogonal functions. *Advances in Water Resources*, vol. 27, no. 1, pp. 57–69. ISSN 03091708.
- Vrugt, J.a., Stauffer, P.H., Wöhling, T., Robinson, B.a. and Vesselinov, V.V. (2008). Inverse Modeling of Subsurface Flow and Transport Properties: A Review with New Developments. *Vadose Zone Journal*, vol. 7, no. 2, p. 843.
- Šimůnek, J., Šejna, M. and van Genuchten, M.T. (2007). User Manual: The HYDRUS Software Package for Simulating the Two- and Three-Dimensional Movement of Water, Heat, and Multiple Solutes in Variably-Saturated Media.
- Winton, C., Pettway, J., Kelley, C., Howington, S. and Eslinger, O.J. (2011 December). Application of Proper Orthogonal Decomposition (POD) to inverse problems in saturated groundwater flow. *Advances in Water Resources*, vol. 34, no. 12, pp. 1519–1526. ISSN 03091708.
- Wu, Y. (2005). *Groundwater recharge estimation in Table Mountain Group aquifer systems with a case study of Kammanassie area*. Ph.D. thesis, University of the Western Cape.
- Xiao, D., Fang, F., Buchan, A.G., Pain, C.C., Navon, I.M., Du, J. and Hu, G. (2013). Non-Linear model reduction for the Navier-Stokes Equations using the residual DEIM method.
- Xiao, M., Breitkopf, P., Filomeno Coelho, R., Knopf-Lenoir, C., Sidorkiewicz, M. and Villon, P. (2009 October). Model reduction by CPOD and Kriging. *Structural and Multidisciplinary Optimization*, vol. 41, no. 4, pp. 555–574. ISSN 1615-147X.
- Xu, Y., Lin, L. and Jia, H. (2009). Groundwater Flow Conceptualization and Storage Determination of the Table Mountain Group (TMG) Aquifers. Tech. Rep. 1419/1/09, Water Research Commission.