UNIVERSITY OF READING

DEPARTMENT OF MATHEMATICS

Non-symmetric Methods in the Modelling of Contaminant Transport in Porous Media

by

Kieran Joseph Neylon

Thesis submitted for the degree of

Doctor of Philosophy

DECEMBER 1994

Abstract

In this thesis, the numerical solution of the governing equations for contaminant transport in porous media is considered and some of the implications of using an implicit Galerkin discretisation approach are examined.

The discretisation of the contaminant mass balance equation generates a large sparse non-symmetric linear system which must be solved to produce the approximate solution. This discretisation permits unphysical extrema in the approximate solution. Techniques to control this, including mesh refinement and flux corrected transport, are given and demonstrated to be effective.

A problem with the robustness of the recently developed non-symmetric iterative solver, Bi-CGSTAB, is highlighted. This is shown to be caused by rounding errors corrupting sensitive values. The accumulation of these rounding errors during the iteration destroys the properties of the underlying recursion process. These sensitive values are generated because the Bi-CGSTAB method is not capable of representing the eigenvalues of the matrix when these have a significant imaginary part. Two methods for overcoming this problem are given and shown to be effective. The first restarts the iteration, which has the effect of discarding rounding errors in the process so that they cannot accumulate. The second avoids the generation of the sensitive values by modifying the underlying recursion process.

The governing equations for the fluid and the contaminant cannot be solved separately due to the dependency of the fluid density on the contaminant concentration. A coupling iteration is used to allow the governing equations to be solved individually. A partial coupling approach for saline intrusion, which relies on the relative weakness of the dependency of the fluid density on the contaminant concentration, is shown to give results which show no appreciable loss of accuracy but are obtained at a much lower computational expense than the fully coupled solution.

Acknowledgements

Firstly, I want to thank Dr. Andrew Priestley for enduring the long hours of discussion to which I subjected him during my time at Reading; although not always cheerful, he was always available and helpful.

Special thanks also to Prof. Mike Baines and Dr. Nancy Nichols for their patient supervision over the past three years, and for their suggestions and comments during the preparation of this thesis.

I would also like to thank Dr. Andrea Maffio (CISE, Milan) for suggesting the investigation of a nonsymmetric approach for saline intrusion as a project, and Dr. Steve Lee (Oak Ridge National Laboratory) for suggesting the work in Section 6.4.

I am grateful to Dr. Gerard Sleijpen (Mathematical Institute, Utrecht University) and Dr. Nick Birkett (Oxford University Computing Laboratory) for allowing the use of their codes for BiCGSTAB(ℓ) and GMRES(k) respectively.

This work was financially supported by a NERC CASE studentship with the collaborating body being the Institute of Hydrology, Wallingford. The contacts at the Institute were David M. Cooper and Bob Moore.

Contents

Chapter 1

Introduction

The main purpose of this thesis is to examine the use of non-symmetric iterative solution methods on linear systems which arise from the non-symmetric discretisation of the governing equations of contaminant transport in porous media.

In order that other issues related to the numerical modelling of the physical problem can be addressed, the general structure of the thesis is such that the overall approach for the solution of the governing equations is examined.

1.1 The Physical Problem and its Significance

The particular application area of interest here is groundwater flow. Due to phenomena such as rainfall and the melting of snow and ice, some of the space in the porous material beneath the surface of the ground contains water - this is groundwater. This water is an important part of the hydrological cycle; it is used naturally (e.g. transpiration - the uptake of water from soil by plants) and artificially (e.g. the extraction of water through pumping wells for such uses as irrigation).

There are many sources of pollution in groundwater, for example contamination by domestic and industrial wastes, oil spills, and agricultural activities such as the use of fertilisers and pesticides. Groundwater quality is an issue of some importance. Serious environmental problems arise when groundwater which is polluted above a safe level emerges at ground surface, or discharges into rivers and lakes; or when it is used for water supply for domestic, industrial or agricultural purposes.

In its most general context, this thesis is concerned with the modelling of non-passive, non-reactive, single-species contaminant transport in a porous medium. In this type of flow, a contaminant is advected through a porous medium by a fluid and the contaminant also undergoes diffusion; the contaminant does not undergo any chemical or biological reactions but its presence does affect the physical properties of the fluid, e.g. density, viscosity. A common example of this type of flow is saline intrusion, this is the process of coastal saltwater moving inland and mixing with less dense freshwater. The saline intrusion system is the main physical system examined in this thesis.

1.2 The Need for, and Requirements of, the Mathematical Model

In regions where coastal aquifers are utilised for water supply, saline intrusion leads to a degradation of groundwater quality. In order to plan a strategy for management of groundwater resources, an accurate method of forecasting the response of a groundwater system to changes in usage patterns is essential. If accurate and reliable local expert knowledge is not available, then the best alternative for producing quantitative predictions is a mathematical model.

A mathematical model is a set of equations. These are usually differential equations in space and/or time. They relate the behaviour of the important, or influential, variables in the system. The model gives quantitative results on the behaviour of the system. The accuracy of these results depends on the effectiveness with which the governing equations represent the physical system, and also on how accurately these governing equations are solved. The effectiveness of the representation is not considered in this thesis. Instead the accuracy, and the related issue of the computational expense, are considered.

The modelling of saline intrusion is an important problem - witness the large amount of research and the international conferences held on the subject e.g. [17]. The accurate, fast and reliable modelling of saltwater intrusion systems is the underlying goal of the work in this thesis.

1.3 Mathematical Models of Saline Intrusion

In saline intrusion, there is always a transition zone between the freshwater and the saltwater. This is caused by hydrodynamic dispersion. In some circumstances, the width of this zone is small relative to the thickness of the aquifer so that it can be approximated as a sharp interface [6]. For the type of flow in this thesis, the transition zone is relatively wide and the sharp interface approximation is not valid [64]. In this case, a variable density model must be used.

Variable density transport models are well documented in the literature. They essentially consist of four main components : a *fluid mass balance equation* which ensures that no fluid is gained or lost except by flow through boundaries or sources/sinks, a *contaminant mass balance equation* which performs the same function for the contaminant, *Darcy's law* which is a momentum balance equation made specific to flow in porous media, and a *constitutive equation* which relates the contaminant concentration to the fluid density. Each of the mass balance equations has boundary conditions associated with it - these also form part of the model.

The governing equations for saline intrusion, and their associated boundary conditions, are described in Section 3.1.

1.4 Numerical Solution of the Governing Equations

For most practical problems, the governing equations are too complex to be solved analytically. This is due to many factors such as the heterogeneity of the domain, the irregular shape of the boundaries, and the non-analytic form of various source functions. In mathematical models where the governing equations cannot be solved analytically, numerical methods can be used to generate approximations to the solution.

Numerical methods for differential equations usually replace the true equation by a set of equations based on approximations to the solution at discrete points. For this reason they are known as discretisation methods. Some discretisation methods are described in Section 3.3, and discrete forms of the governing equations for saline intrusion are given in Section 3.4.

The governing equations for the fluid and the contaminant are coupled together due to the dependence of the fluid density on the contaminant concentration; hence they cannot be solved independently of each other. However, by using a coupling iteration, it is possible to treat the equations individually. This aspect of the whole solution procedure is introduced in Section 3.2 and examined in more detail in Chapter 7.

The discretisation of the individual equations results in large sparse linear systems which must be solved to generate the approximate solution. Due to the coupling iteration these systems are linear even though the governing equations are non-linear. The discretisation used for the equations governs the properties of the resulting linear systems. The particular discretisation approach used for the contaminant mass balance equation (a Galerkin finite element spatial approximation with Crank-Nicolson time-stepping) leads to a linear system which has a large, sparse, non-symmetric matrix. This discretisation results in a solution that is unconditionally stable and possesses a good degree of accuracy. The non-symmetry of the matrix is an important feature of the linear system. It is caused by the implicit Galerkin discretisation of the advection term [40, 72, 88].

One of the main differences between the numerical solution approaches for saline intrusion systems in the literature is the method used to discretise the contaminant mass balance equation.

Due to the success of the preconditioned conjugate gradient method for the iterative solution of linear systems with large sparse *symmetric* positive definite matrices, and the lack of a similarly successful solver for linear systems with large sparse non-symmetric matrices, discretisation methods which give rise to systems with non-symmetric matrices tend to be avoided in the literature.

Symmetry of the matrix in the discretised contaminant mass balance equation can be achieved by the use of operator splitting methods which allow the advection component of the equation to be treated explicitly, while the other terms are treated implicitly. This leads to either a decrease in the order of accuracy of the solution [50], or a restriction on the size of the allowable discrete time-step to ensure stability [10] (which makes these methods unfeasible for long term transient calculations).

Symmetry can also be achieved by the use of Lagrangian methods which effectively follow particles along characteristics to solve the advection component (e.g. [30]). The Lagrangian approach, although effective, is not taken in this thesis so that non-symmetric methods can be focussed on.

Due to recent advances in applied linear algebra, there now exist powerful methods for the solution of large, sparse, non-symmetric linear systems, e.g. QMR [28], GMRES [70], Bi-CGSTAB [84]. These methods still do not possess all the advantages of the pre-conditioned conjugate gradient method for symmetric positive definite systems, but they have been shown to be successful in many application areas e.g. [44, 66], and the good properties of the non-symmetric discretisation approach (i.e. simplicity, accuracy and unconditional stability) may outweigh the shortcomings of the available non-symmetric linear solvers.

1.5 Purpose and Structure of the Thesis

A comparison of a symmetric and a non-symmetric approach for the numerical solution of the contaminant mass balance equation is made in [62] where the symmetric approach is found to be more effective. But the study in [62] only considers the relative computing times for the two methods on different computer architectures for a complicated case study, it does not consider the errors caused by the discretisation or the practical robustness of the overall solution procedure.

The original purpose of this work was to fully compare the symmetric and non-symmetric approaches to determine if the latter is as unfeasible as commonly accepted. However, due to a problem with the robustness of the non-symmetric linear solver Bi-CGSTAB which was uncovered and examined, this original purpose was discarded. Hence, the main purpose of this thesis is to re-examine the non-symmetric approach to the numerical solution of the contaminant mass balance equation in the light of the new non-symmetric solvers which are available.

Chapter 2 provides relevant background on iterative methods for large sparse linear systems. This background covers methods for symmetric positive definite and non-symmetric systems.

In Chapter 3, the governing equations for contaminant transport in a porous medium are given. Discretisation methods in both space and time are also introduced and described in this chapter. Finally, a particular discretisation approach is applied to the governing equations so that these can be solved approximately.

The performance of the discretisation methods used on the governing equations is examined by numerical experiments in Chapter 4, with particular attention being given to the control of unphysical oscillations which can arise in the approximate solution due to the nature of the discretisation process.

In Chapter 5, the performance of the conjugate gradient method on the symmetric positive definite systems which arise is examined. These matrices are generated during the numerical solution of the fluid mass balance equation and Darcy's law. Particular attention is given to the behaviour of the solver when an extremely low convergence tolerance is requested, and also to mesh dependence of convergence and preconditioning. The behaviour of the symmetric positive definite solver used - the preconditioned conjugate gradient method - is well understood and documented in the literature. The purpose of Chapter 5 is to introduce the type of numerical experiments that are conducted on the non-symmetric solvers.

In Chapter 6, the performance of two non-symmetric solvers (Bi-CGSTAB and GMRES) is compared. The behaviour of the Bi-CGSTAB solver with an extremely low convergence tolerance is then examined and it is shown to be unreliable in some cases. Techniques for improving the robustness of Bi-CGSTAB are examined and tested.

The coupling iteration, which allows the (inter-linked) governing equations to be solved independently of each other, is examined in Chapter 7. An approach which operates at between approximately two-thirds and one-quarter of the computational effort is tested to examine its effect on the accuracy of the overall solution.

In the final chapter, the main conclusions drawn from this work are collated, and some suggestions for further, or related, studies are made.

Throughout this thesis, the following notation convention is used for mathematical symbols. Scalars are denoted by lowercase italics and lowercase Greek letters, vectors by lowercase bold italics. Matrices are denoted by uppercase italics - an exception to this being rank 2 tensors which are represented by underlined bold italics. Finally, sets and vector spaces are denoted by uppercase calligraphic letters.

Chapter 2

Solution of Large Sparse Linear Systems

As stated in the introduction, the numerical solution of differential equations requires a discrete representation of both the unknown function and the differential equation. The discretisation of a differential equation generally gives rise to a system of algebraic equations. Although discretisation techniques are outlined and investigated in Chapters 3 and 4, it is the solution of linear systems that forms the focus of the main part of this thesis.

In the current chapter, relevant techniques for the solution of systems of linear equations are reviewed. The problem can be stated as follows. Given a matrix $A \in \mathbb{R}^{n\times n}$ (assumed to be invertible) and a vector $\boldsymbol{b} \in \mathbb{R}^n$, solve

$$A\boldsymbol{x} = \boldsymbol{b}. \tag{2.1}$$

for $\boldsymbol{x} \in \mathbb{R}^n$.

The matrices that arise in this thesis are real, large and sparse[1], so only matrices of this type are considered. Such matrices are generated by most standard discretisation methods for partial differential equations (an exception being boundary element methods [7] which produce matrices that are fully populated by non-zero coefficients). If the sparsity in a matrix is fully exploited, only the non-zero entries are stored and methods using only these non-zero entries are employed.

[1]A matrix is said to be *sparse* if only a relatively small number of its entries are non-zero.

A primary distinction made between methods for the solution of (2.1) is whether the approach is *direct* or *iterative*.

- Direct methods require a fixed number of operations. If successful, they return the exact solution ($\boldsymbol{x} = A^{-1}\boldsymbol{b}$) to within the limits allowed by machine accuracy. Classical direct methods do not generally exploit the sparsity in a matrix. (An implementation of a direct method which does exploit sparsity is the frontal method which is described in the following section.)

- Iterative methods generate a sequence of iterates ($\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$) which, if the method is successful, converge to the exact solution. Numerically the convergence is measured in a suitable norm.

  Iterative methods are usually based on matrix-vector multiplications, which allows them to exploit any sparsity which is present in the matrix, both in terms of storage and operations per iteration - the goal being to minimise the number of iterations needed to achieve a specified accuracy.

In the next section, a brief overview of direct methods is given. This is followed in the remainder of the chapter by an overview of some current iterative methods.

2.1 Direct Methods

Most direct methods for the solution of linear systems are based on Gaussian elimination. This method uses elementary row operations, for example the addition of a constant multiple of one row to another row, to change the original system (2.1) to the row-equivalent form

$$U\boldsymbol{x} = L^{-1}\boldsymbol{b}, \tag{2.2}$$

where $L \in \mathbb{R}^{n\times n}$ is a unit lower triangular matrix (i.e. a lower triangular matrix with "1"s on the diagonal) and $U \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. The solution of the upper triangular system (2.2) is relatively trivial by backward substitution [31].

In many applications, systems with the same matrix A but different vectors $\boldsymbol{b}$ need to be solved. An effective approach in this situation is to compute the LU

factorisation of A, that is generate triangular matrices L and U (as previously defined) such that $A = LU$. Then the solution of (2.1) is reduced to one of solving the system

$$L\boldsymbol{y} = \boldsymbol{b}$$

for $\boldsymbol{y}$ by forward substitution, and then solving the system

$$U\boldsymbol{x} = \boldsymbol{y}$$

for $\boldsymbol{x}$ by backward substitution. In the symmetric case, the computation of the L and U $(= L^T)$ factors, known as a Cholesky factorisation, requires square root calculations. This is an expensive floating point operation compared with additions and multiplications so, in practice, the LDL^T factorisation, which does not require any square root calculations, is used. Algorithm 2.1 is the non-symmetric version of this factorisation, it generates unit lower-triangular matrices, L and M, and a diagonal matrix D such that $A = LDM^T$.

Algorithm 2.1 LDM^T Factorisation

Given $A \in \mathbb{R}^{n \times n}$, the following algorithm computes the factorisation $A = LDM^T$. A is overwritten by $L' + M'^T$ where L' and M' are the strictly lower triangular parts of L and M, and the diagonal matrix D is stored in the n-vector $[d_1, d_2, \ldots, d_n]^T$.

```
for $k = 1, \ldots, n-1$
  for $p = 1, \ldots, k-1$
    $r_p := d_p a_{pk}$
    $w_p := a_{kp} d_p$
  end for
  $d_k := a_{kk} - \sum_{p=1}^{k-1} a_{kp} r_p$
  if $(d_k = 0)$ quit
  for $i = k+1, \ldots, n$
    $a_{ik} := \left(a_{ik} - \sum_{p=1}^{k-1} a_{ip} r_p\right) / d_k$
    $a_{ki} := \left(a_{ki} - \sum_{p=1}^{k-1} w_p a_{pi}\right) / d_k$
  end for
end for
```

Note that, since the L and M factors are unit lower-triangular matrices, the diagonal matrix D can be stored in the diagonal entries of one of these triangular factors.

If Algorithm 2.1 runs successfully to completion, $\frac{1}{3}n^3$ floating point operations (flops) are required. However, the algorithm breaks down if the matrix has a singular leading principal submatrix (which corresponds to $d_k = 0$).

Round-off error analysis of Gaussian elimination (see e.g. [31] Section 4.3) shows that the method is very sensitive to rounding errors that occur in finite precision arithmetic when the diagonal entries of A are small relative to the other entries in the same column of the lower triangular part.

The method works well for systems in which the matrix is well-conditioned and has no singular leading principle sub-matrices, but it is unstable for general matrices. The instability and possible breakdowns are overcome by *pivoting*, this is the interchange of rows during the elimination to avoid zero (or small) a_{kk} in the algorithm. From [31], Gaussian elimination with pivoting is stable if A is non-singular.

The most successful Gaussian elimination based method for large problems is the *frontal method* [22, 23] in which the matrix is assembled row-by-row and the elimination is carried out during the assembly. For a strategy of this form used in conjunction with a matrix of bandwidth d, in the absence of pivoting, a $(d+1) \times (d+1)$ submatrix is required at each stage of the elimination (this submatrix is larger if pivoting is required). Also, after the elimination phase is completed for a row, that row is not required in the algorithm. Hence, if the bandwidth of the matrix is relatively small, only a small part of the matrix needs to be stored in main memory and the rest can be held in (slower) backing store.

Direct methods suffer from the problem of *fill-in*, i.e. zero entries in the original system become non-zero during the elimination. Because of this fill-in, more storage is required to store the factorised system than the original system. If the bandwidth of the matrix is large, the level of fill-in is high, so the required storage is greatly increased. Hence, the equations must be ordered (in a pre-processing phase before the elimination) so that the level of fill-in is minimised. This pre-processing phase must take into account future pivoting.

In some matrices, particularly those arising from the discretisation of 3-D problems, the bandwidth is large and direct methods based on elimination are impractical because of excessive demands on storage and time. For these types of problems, iterative methods are the only feasible approach. A comparison of direct and iterative methods for the types of systems of interest in this thesis is given in [8]. That comparison confirms the general statements made in this section on the storage and computational effort for direct methods.

Direct methods are not used in this thesis and all linear systems which arise are solved by iterative methods.

2.2 Classical Iterative Methods

Classical iterative methods for solving (2.1) are based on a *splitting* of the matrix, A, i.e.

$$A = M - N \tag{2.3}$$

where M is invertible. Given such a splitting, a classical iterative method is constructed by setting,

$$M\boldsymbol{x}_{i+1} = N\boldsymbol{x}_i + \boldsymbol{b},$$

i.e.

$$\boldsymbol{x}_{i+1} = M^{-1}N\boldsymbol{x}_i + M^{-1}\boldsymbol{b}. \tag{2.4}$$

$M^{-1}N$ is known as the iteration matrix. In order for the iteration to be computationally viable, M should be relatively trivial to invert, e.g. a diagonal or triangular matrix.

Definition 2.1 *The* spectral radius *of* $A \in \mathbb{R}^{n\times n}$ *is*

$$\rho(A) = \max_{1\leq i\leq n} |\lambda_i|$$

where λ_i $(i = 1, \ldots, n)$ *are the eigenvalues of* A*.*

It can be shown (see e.g. Theorem 10.1-1 in [31]) that

$$\rho(M^{-1}N) < 1$$

is a necessary and sufficient condition for the iteration given by (2.4) to converge for any choice of initial iterate, $\boldsymbol{x}_0$.

Bearing in mind the previous comment on the ease of invertibility of M, the matrix A can be usefully decomposed as

$$A = D - L - U, \tag{2.5}$$

where D is a diagonal matrix consisting of the diagonal entries of A, $-L$ is a strictly lower triangular matrix consisting of the sub-diagonal entries of A, and $-U$ is a strictly upper triangular matrix consisting of the super-diagonal entries of A. The relationship between the splitting (2.3) and the decomposition (2.5) of A governs the method. Table 2.1 shows this relationship for some classical iterative methods.

Iterative Method	M	N	$M^{-1}N$
Jacobi	D	$L+U$	B
Gauss-Seidel	$D-L$	U	$\mathcal{L}_1$
Successive Over-relaxation	$\frac{1}{\omega}(D-\omega L)$	$\frac{1}{\omega}\{(1-\omega)D+\omega U\}$	$\mathcal{L}_\omega$

Table 2.1: Classical iteration methods

Definition 2.2 *The* asymptotic rate of convergence *associated with a classical iterative method with iteration matrix A is*

$$R_\infty(A) = -ln\{\rho(A)\}.$$

The asymptotic rate of convergence is a measure of the rapidity of convergence of the iteration. From [87], if A is a 2-cyclic consistently ordered matrix with non-zero diagonal entries (conditions which are generally satisfied by finite difference discretisations of partial differential equations), then, if $\rho(B) < 1$,

- $\lambda = \mu^2$ where λ is an eigenvalue of B and μ is an eigenvalue of $\mathcal{L}_1$, so

$$R_\infty(\mathcal{L}_1) = 2R_\infty(B).$$

- successive over-relaxation (SOR) converges only if $0 < \omega < 2$. If the value of the relaxation parameter ω which gives the fastest rate of convergence is ω_{opt} then, since Gauss-Seidel is a special case of SOR,

$$R_\infty(\mathcal{L}_{\omega_{opt}}) \geq R_\infty(\mathcal{L}_1).$$

That is, SOR (with optimum relaxation parameter) converges at least as fast as the Gauss-Seidel method, and both converge faster than the Jacobi method.

Due to the advent of Krylov subspace methods, classical iterative methods are becoming obsolete as linear solvers.

2.3 Krylov Subspace Methods

As their name suggests, these methods search for the solution to (2.1) in a *Krylov subspace.*

Definition 2.3 *The* i^{th} Krylov subspace *of* $\mathbb{R}^n$ *of the matrix* $A \in \mathbb{R}^{n\times n}$ *and the vector* $\boldsymbol{r} \in \mathbb{R}^n$ *is*

$$\mathcal{K}_i(A, \boldsymbol{r}) = \text{span}(\boldsymbol{r}, A\boldsymbol{r}, A^2\boldsymbol{r}, \ldots, A^{i-1}\boldsymbol{r}).$$

The following theorem indicates why it is desirable to search for the solution in a space of this type.

Theorem 2.1 Cayley-Hamilton theorem : *Every matrix satisfies its own characteristic equation, i.e. if det*$(xI - A) = \psi(x)$ *(a polynomial of degree n) then* $\psi(A) = 0$

Proof : See [15] p337. □

From Theorem 2.1,

$$A^n + c_1A^{n-1} + c_2A^{n-2} + \ldots + c_{n-2}A^2 + c_{n-1}A + c_nI = 0$$

where $c_j \in \mathbb{R}$ and $c_n = (-1)^n \det(A)$, which implies that, if A is nonsingular, then

$$\begin{aligned} A^{-1} &= \frac{-1}{c_n}(c_{n-1}I + c_{n-2}A + \ldots + c_2A^{n-3} + c_1A^{n-2} + A^{n-1}) \\ &= \tilde{c}_0 I + \tilde{c}_1 A + \ldots + \tilde{c}_{n-3}A^{n-3} + \tilde{c}_{n-2}A^{n-2} + \tilde{c}_{n-1}A^{n-1}. \end{aligned} \tag{2.6}$$

Hence

$$A^{-1} \in \operatorname{span}(I, A, A^2, \ldots, A^{n-1}). \tag{2.7}$$

Now, given an arbitrary vector $\boldsymbol{v} \in \mathbb{R}^n$,

$$\begin{aligned} \boldsymbol{x} - \boldsymbol{v} &= A^{-1}\boldsymbol{b} - \boldsymbol{v} \\ &= A^{-1}(\boldsymbol{b} - A\boldsymbol{v}) \\ &= A^{-1}\boldsymbol{r} \end{aligned}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ is the residual vector corresponding to $\boldsymbol{x} = \boldsymbol{v}$ in (2.1). Hence, from (2.7),

$$\boldsymbol{x} - \boldsymbol{v} \in \operatorname{span}(\boldsymbol{r}, A\boldsymbol{r}, A^2\boldsymbol{r}, \ldots, A^{n-1}\boldsymbol{r}) = \mathcal{K}_n(A, \boldsymbol{r})$$

so, given an arbitrary vector $\boldsymbol{v}$, the solution to (2.1) lies in the vector space spanned by $\boldsymbol{v}$ and the Krylov space associated with the matrix A and the residual vector, $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{v}$.

From Definition 2.3, it is clear that $\mathcal{K}_i(A, \boldsymbol{r}) \supset \mathcal{K}_{i-1}(A, \boldsymbol{r})$. If d is the smallest integer such that

$$A^d\boldsymbol{r} \in \mathcal{K}_d(A, \boldsymbol{r})$$

then the dimension of $\mathcal{K}_i(A, \boldsymbol{r})$ is

$$\dim\{\mathcal{K}_i(A, \boldsymbol{r})\} = \begin{cases} i & \text{for} \quad i \le d \\ d & \text{for} \quad i \ge d. \end{cases}$$

As noted in [67], in general $d = n$ but, if A has multiple eigenvalues or $\boldsymbol{r}$ happens to have a zero component of any eigenvector of A, then $d < n$. Thus the solution vector $\boldsymbol{x}$ lies in the space given by

$$\boldsymbol{x} - \boldsymbol{v} \in \mathcal{K}_d(A, \boldsymbol{r})$$

and, by constructing the correct vector in this space, the solution is obtained in d steps where each step adds a vector to the current Krylov subspace. Hence this approach would appear to be a direct method since the exact solution is found, in theory, in a finite number of operations.

However, in the presence of rounding errors, this finite termination property does not hold. Also, for very large matrices, the computation of $\mathcal{K}_d(A, \boldsymbol{r})$ is expensive. Fortunately, as demonstrated in [67], Krylov space methods are practical if considered as iterative methods, i.e. given an initial iterate, $\boldsymbol{x}_0$, construct iterates for the solution of (2.1) that satisfy,

$$\boldsymbol{x}_i - \boldsymbol{x}_0 \in \mathcal{K}_i(A, \boldsymbol{r}_0) \qquad , i = 1, 2, \ldots$$

where $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ is the initial residual vector. This is equivalent to

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + c_0\boldsymbol{r}_0 + c_1 A\boldsymbol{r}_0 + c_2 A^2\boldsymbol{r}_0 + \ldots + c_{i-1}A^{i-1}\boldsymbol{r}_0 \qquad , i = 1, 2, \ldots \tag{2.8}$$

where the $c_j \in \mathrm{I\!R}$ are coefficients to be determined (not to be confused with the coefficients in (2.6)).

Definition 2.4 *The i^{th}* residual polynomial*, ϕ_i, is the (real) polynomial of degree at most i such that*

$$\boldsymbol{r}_i = \phi_i(A)\boldsymbol{r}_0$$

with $\phi_i(0) = 1$.

The existence of this polynomial is seen by considering the definition of the residual and the polynomial in (2.8). This gives

$$\begin{aligned}
\boldsymbol{r}_i &= \boldsymbol{b} - A\boldsymbol{x}_i \\
&= \boldsymbol{b} - A\{\boldsymbol{x}_0 + c_0\boldsymbol{r}_0 + c_1 A\boldsymbol{r}_0 + c_2 A^2\boldsymbol{r}_0 + \ldots + c_{i-1}A^{i-1}\boldsymbol{r}_0\} \\
&= (1 - Ac_0)\boldsymbol{r}_0 - c_1 A^2\boldsymbol{r}_0 - c_2 A^3\boldsymbol{r}_0 - \ldots - c_{i-1}A^i\boldsymbol{r}_0 \\
&= \phi_i(A)\boldsymbol{r}_0 \qquad\qquad \text{(with } \phi_i \text{ as in Definition 2.4).}
\end{aligned}$$

In Krylov subspace methods, ϕ_i (and hence the unknown coefficients in (2.8)) is chosen at each iteration such that

$$\boldsymbol{r}_i \approx \boldsymbol{0}$$

in some sense. Two ways of achieving this are :

1. Minimize the residual in some norm over the appropriate space,

$$\|\boldsymbol{r}_i\| = \min_{\mathbf{z}-\mathbf{x}_0 \in \mathcal{K}_i(A,\mathbf{r}_0)} \|\boldsymbol{b} - A\boldsymbol{z}\| = \min_{\phi \in \mathcal{P}_i : \phi(0)=1} \|\phi(A)\boldsymbol{r}_0\|, \tag{2.9}$$

   where $\mathcal{P}_i$ denotes the set of all real polynomials of degree at most i. This approach is known as a *minimum residual* method. An iterate that satisfies (2.9) is always uniquely defined.

2. Satisfy a Petrov-Galerkin condition,

$$\boldsymbol{s}^T \boldsymbol{r}_i = \mathbf{0} \ \ \forall \ \ \boldsymbol{s} \in \mathcal{S}_i, \tag{2.10}$$

   where $\mathcal{S}_i \subset \mathrm{IR}^n$ is a subspace of dimension i. This approach is known as a *projection* method. An iterate satisfying (2.10) is not always defined. In the case $\boldsymbol{s} = \boldsymbol{r}_j \ \ (i \neq j)$ these are known as *orthogonal residual* methods.

Many minimum residual methods also satisfy a projection condition.

Storage demands for Krylov subspace methods are greater than for the classical splitting-based methods in Section 2.2, but this is repaid by greater speed of convergence and applicability to systems in which the spectral radius of the iteration matrix is greater than unity.

The generation of a Krylov subspace only involves matrix-vector products. In general, satisfying a minimum residual or Petrov-Galerkin condition involves vector-vector products, vector updates and scalar operations. The matrix-vector product is usually the most expensive part of the algorithm.

The matrix-vector product can be successfully implemented on vector and parallel machines: this adds to the popularity of Krylov subspace methods. The parallel implementation of Krylov subspace methods is only discussed intermittently during this thesis - see [69] and the references therein for more information on this aspect of this class of methods.

2.3.1 Symmetric Positive Definite Matrices

In this section, the case where A is symmetric and positive definite is considered.

Definition 2.5 *A matrix A is* symmetric *if, for all i and j, the $(i,j)^{\text{th}}$ entry in the matrix is equal to the $(j,i)^{\text{th}}$ entry.*

Definition 2.6 *A* positive definite *matrix, $A \in \mathbb{R}^{n\times n}$, satisfies*

$$\boldsymbol{w}^T A \boldsymbol{w} > 0 \quad \textit{for} \;\; \{\boldsymbol{w} \in \mathbb{R}^n | \boldsymbol{w} \neq \boldsymbol{0}\}.$$

Matrices of this type occur throughout this thesis, so the phrase "symmetric and positive definite" is abbreviated to SPD. For this type of matrix, if the Krylov subspace method is made to possess the minimum residual property,

$$\|\boldsymbol{r}_i\|_{A^{-1}} = \min_{\mathbf{z}-\mathbf{x}_0 \in \mathcal{K}_i(A,\mathbf{r}_0)} \|\boldsymbol{b} - A\boldsymbol{z}\|_{A^{-1}}, \tag{2.11}$$

(where $\|\boldsymbol{w}\|_M = \sqrt{\boldsymbol{w}^T M \boldsymbol{w}}$ is the norm on $\mathbb{R}^n$ associated with the SPD matrix, $M \in \mathbb{R}^{n\times n}$), then the result is the *conjugate gradient* method (abbreviated to CG hereafter). This name arises because the method also possesses the orthogonal residual property,

$$\boldsymbol{r}_i^T \boldsymbol{r}_j = 0 \quad (i \neq j).$$

There are many versions of CG that are equivalent in exact arithmetic, but show different behaviour in finite precision. The algorithm used in this work is that given in [31].

Algorithm 2.2 CG

Given an initial iterate $\boldsymbol{x}_0 \in \mathbb{R}^n$, this algorithm generates iterates $\boldsymbol{x}_i \in \mathbb{R}^n$, for the solution of (2.1) where A is SPD.

```
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$
for $i = 1, 2, \ldots$ until satisfied (see following section)
  $\beta_i := (\boldsymbol{r}_{i-1}^T \boldsymbol{r}_{i-1})/(\boldsymbol{r}_{i-2}^T \boldsymbol{r}_{i-2})$        $(\beta_1 := 0)$
  $\boldsymbol{p}_i := \boldsymbol{r}_{i-1} + \beta_i \boldsymbol{p}_{i-1}$                   $(\boldsymbol{p}_1 := \boldsymbol{r}_0)$
  $\alpha_i := (\boldsymbol{r}_{i-1}^T \boldsymbol{r}_{i-1})/(\boldsymbol{p}_i^T A \boldsymbol{p}_i)$
  $\boldsymbol{x}_i := \boldsymbol{x}_{i-1} + \alpha_i \boldsymbol{p}_i$
  $\boldsymbol{r}_i := \boldsymbol{r}_{i-1} - \alpha_i A \boldsymbol{p}_i$
end for
```

In practice, apart from A, $\boldsymbol{x}$ and $\boldsymbol{b}$, this algorithm requires 3 vectors in $\mathbb{R}^n$ for storage. The computational cost per iteration is 1 matrix-vector product, 2 vector-vector products and 3 SAXPYs[2].

Convergence Criteria

The term "until satisfied" appears in Algorithm 2.2 (and all subsequent iterative algorithms). This means that the current iterate is perceived to be sufficiently close to the solution of (2.1). The error in the current iterate, $\boldsymbol{e}_i = \boldsymbol{x} - \boldsymbol{x}_i$, is difficult to measure directly since the solution to (2.1) is not known. Instead, stopping (or convergence) criteria based on the residual are used (in conjunction with a user-supplied tolerance, $\tau \geq 0$).

The simplest stopping criterion is

$$\|\boldsymbol{r}_i\|_2 \leq \tau.$$

This is an absolute measure and hence does not reflect any properties of the problem. A relative criterion is usually more effective, e.g.

$$\|\boldsymbol{r}_i\|_2 \leq \tau\|\boldsymbol{b}\|_2. \tag{2.12}$$

Definition 2.7 *The* condition number *of $A \in \mathbb{R}^{n\times n}$ associated with the 2-norm of the matrix is*

$$1 \leq \kappa_2(A) = \|A\|_2\|A^{-1}\|_2 \leq \infty$$

with the convention that $\kappa_2(A) = \infty$ for singular A. If A is SPD,

$$\kappa_2(A) = \frac{\lambda_{max}}{\lambda_{min}},$$

where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of A respectively.

Note that (e.g. from [2]) the relative error is bounded according to,

$$\frac{\|\boldsymbol{e}_i\|_2}{\|\boldsymbol{x}\|_2} \leq \kappa_2(A)\frac{\|\boldsymbol{r}_i\|_2}{\|\boldsymbol{b}\|_2}.$$

So, for a system with a well-conditioned matrix (i.e. $\kappa_2(A) = \mathrm{O}(1)$), the tolerance, τ, in (2.12) gives a bound on the relative error in the current iterate. Indeed, if an

[2]A SAXPY operation is the summation of a vector and a constant multiple of another vector.

upper bound on the condition number of the matrix is available, $\tilde{\kappa}_2(A)$ say, then an upper bound (of τ) on the relative error is provided by the stopping criterion,

$$\|\boldsymbol{r}_i\|_2 \leq \frac{\tau}{\tilde{\kappa}_2(A)}\|\boldsymbol{b}\|_2.$$

The effect of using different stopping criteria is not investigated in this thesis - the stopping criterion (2.12) is used throughout. See e.g. [1, 2] for some other stopping criteria and investigations into their behaviour.

Connection between CG and Lanczos Algorithm

CG is based on three-term recurrence relations, e.g. from Algorithm 2.2,

$$\boldsymbol{r}_i = \left(1 - \alpha_i A + \frac{\alpha_i}{\alpha_{i-1}}\beta_i\right)\boldsymbol{r}_{i-1} - \frac{\alpha_i}{\alpha_{i-1}}\beta_i \boldsymbol{r}_{i-2}$$

and

$$\boldsymbol{p}_{i+1} = \left(1 - \alpha_i A + \beta_{i+1}\right)\boldsymbol{p}_i - \beta_i \boldsymbol{p}_{i-1}.$$

Because information is held implicitly in these recurrence relations, the vectors which span the Krylov subspace do not need to be stored. This property makes the algorithm economical in terms of both computing time and storage. The three-term recurrence relations arise from the connection between the CG algorithm and the Lanczos tridiagonalisation process for symmetric matrices. A brief description of this connection follows; for full details see [26] and [31] (Chapters 7, 9 and 10).

From [31] (Theorem 9.1-1), the Lanczos procedure is such that, if an initial vector,

$$\boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\tilde{\beta}_0} \qquad \left(\tilde{\beta}_0 = \|\boldsymbol{r}_0\|_2\right)$$

and symmetric matrix, A, are supplied, then after i steps of the process, an orthogonal basis for $\mathcal{K}_i(A, \boldsymbol{v}_1)$ is generated and the following relationships hold,

$$AV_i = V_i T_i + \tilde{\beta}_{i+1}\boldsymbol{v}_{i+1}(\boldsymbol{e}_i^i)^T$$

where $\boldsymbol{e}_j^i$ is the j^{th} column of the $(i \times i)$ identity matrix (not to be confused with the error associated with the i^{th} iterate, $\boldsymbol{e}_i$),

$$T_i = \begin{bmatrix} \tilde{\alpha}_1 & \tilde{\beta}_1 & & & & \\ \tilde{\beta}_1 & \tilde{\alpha}_2 & \tilde{\beta}_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \tilde{\beta}_{i-2} & \tilde{\alpha}_{i-1} & \tilde{\beta}_{i-1} \\ & & & & \tilde{\beta}_{i-1} & \tilde{\alpha}_i \end{bmatrix} \in \mathbb{R}^{i\times i}$$

$$V_i = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i] \in \mathbb{R}^{n\times i} \text{ has orthonormal columns,}$$

and

$$V_i^T \boldsymbol{v}_{i+1} = \boldsymbol{0}$$

$$\text{Range}(V_i) = \mathcal{K}_i(A, \boldsymbol{r}_0).$$

From [59], an approximation to the solution of (2.1) can be constructed from the Lanczos coefficients in T_i and the orthonormal basis which is spanned by the columns of V_i in the form

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + V_i \boldsymbol{y}_i,$$

where $\boldsymbol{x}_0$ is the vector associated with the residual vector $\boldsymbol{r}_0$ and $\boldsymbol{y}_i \in \mathbb{R}^i$ arises from solving

$$T_i \boldsymbol{y}_i = \tilde{\beta}_1 \boldsymbol{e}_1^i. \tag{2.13}$$

The matrix, $T_i = V_i^T A V_i$ is SPD if A is SPD. In this case, the Cholesky factorisation

$$T_i = L_i D_i L_i^T \tag{2.14}$$

exists ($L_i \in \mathbb{R}^{i\times i}$ is a unit lower triangular matrix and $D_i \in \mathbb{R}^{i\times i}$ is a diagonal matrix). The factored system (2.14) is used in the solution of (2.13).

If the Cholesky factorisation is performed in such a way that $V_i \boldsymbol{y}_i$ can be accumulated as i increases (see e.g. [59]), and the Lanczos process is performed with overwriting so that the Lanczos vectors, $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{i-1}$, do not need to be stored (e.g. as in [31] Algorithm 9.1-1), then this approach is equivalent to CG as given in Algorithm 2.2.

Theoretical Convergence Results for CG

By recasting the minimum residual property (2.11) as a minimisation of the residual polynomial (Definition 2.4) over the set of all possible polynomials, and approximating this minimum by a Chebychev polynomial of the first kind on the interval $[\lambda_{min}, \lambda_{max}]$, it can be shown (see [4, 31]) that, in exact arithmetic, a bound on the error in the i^{th} CG iterate is given by

$$\|\boldsymbol{e}_i\|_A \leq 2\|\boldsymbol{e}_0\|_A \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}\right)^i . \tag{2.15}$$

This bound is quite loose and faster convergence is obtained in practice. It is possible to make the bound tighter by including more information about the eigenspectrum of A in the approximation to the minimum polynomial (see e.g. [83]).

The finite precision behaviour of CG has recently been analysed (see e.g. [32, 34, 57, 77]) with the conclusion that the effect of finite precision arithmetic is to "smear" the eigenvalues around their true position so that finite precision CG solves exactly an equivalent larger system

$$\tilde{A}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}} \quad \tilde{A} \in \mathrm{I\!R}^{n' \times n'} \ , \ \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}} \in \mathrm{I\!R}^{n'} \quad (n' > n)$$

where the eigenvalues of the SPD matrix $\tilde{A}$ are clustered around those of A.

From (2.15), the convergence rate of CG is very fast for matrices with condition number near unity, but slow for matrices with large condition number. Matrices with large condition number as said to be *poorly conditioned*. When CG is applied to a system with a poorly conditioned matrix, many iterations are needed to reduce the error in the solution to an acceptable level - this makes the method expensive in terms of the required number of operations. This drawback nearly led to the demise of the method until, in [67], it was demonstrated that a technique called *preconditioning* can be used to transform the system into one which has a matrix with a smaller condition number. This technique is described in the following section.

Preconditioning

The system to be solved (2.1) can be transformed into another system (with the same solution) that has a matrix which is better conditioned than the original matrix by pre- or post-multiplying the system by a preconditioning matrix Z. Pre-multiplying the system in this way is termed "preconditioning from the left",

$$Z^{-1}A\boldsymbol{x} = Z^{-1}\boldsymbol{b},$$

while post-multiplying by the preconditioner is termed "preconditioning from the right",

$$AZ^{-1}\boldsymbol{y} = \boldsymbol{b} \quad , \quad \boldsymbol{x} = Z^{-1}\boldsymbol{y}.$$

Only preconditioning from the left is used in this thesis. The ideal preconditioning matrix gives $Z^{-1}A$ to be the identity matrix (since $\kappa_2(I) = 1$), in which case, $Z^{-1} = A^{-1}$. However, the computation of the action of A^{-1} is the same as a computing a solution of the original problem (2.1).

In practice, Z^{-1} is taken to be a matrix which is close, in some sense, to the inverse of A, and which is relatively trivial to compute. It is rarely computed explicitly. Instead preconditioner systems of the form

$$Z\boldsymbol{u} = \boldsymbol{v}$$

are solved at each iteration to produce $\boldsymbol{u} = Z^{-1}\boldsymbol{v}$. Algorithm 2.3 is the preconditioned form of Algorithm 2.2.

Algorithm 2.3 PCG

Given an initial iterate, $\boldsymbol{x}_0 \in \mathbb{R}^n$, and an SPD preconditioning matrix, $Z \in \mathbb{R}^{n\times n}$, this algorithm generates iterates, $\boldsymbol{x}_i \in \mathbb{R}^n$, for the solution of (2.1) where A is SPD by preconditioned CG.

```
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$
for $i = 1, 2, \ldots$ until satisfied
  $\boldsymbol{z}_{i-1} := Z^{-1}\boldsymbol{r}_{i-1}$
  $\beta_k := (\boldsymbol{z}_{i-1}^T\boldsymbol{r}_{i-1})/(\boldsymbol{z}_{i-2}^T\boldsymbol{r}_{i-2})$        $(\beta_1 := 0)$
  $\boldsymbol{p}_i := \boldsymbol{z}_{i-1} + \beta_i\boldsymbol{p}_{i-1}$        $(\boldsymbol{p}_1 := \boldsymbol{r}_0)$
```

```
    $\alpha := (\boldsymbol{z}_{i-1}^T \boldsymbol{r}_{i-1})/(\boldsymbol{p}_i^T A \boldsymbol{p}_i)$
    $\boldsymbol{x}_i := \boldsymbol{x}_{i-1} + \alpha \boldsymbol{p}_i$
    $\boldsymbol{r}_i := \boldsymbol{r}_{i-1} - \alpha A \boldsymbol{p}_i$
end for
```

If this algorithm is compared with Algorithm 2.2, it can be seen to require one extra n-vector of storage and the only extra work involved is the solution of the preconditioner system at each iteration.

There are many different forms of preconditioners in the literature. A recent overview of the state-of-the-art in preconditioning is given in [3]. It is possible to use information on the problem to build preconditioners dedicated to systems for that problem, e.g. the use of Green's functions in [14]. Most general preconditioners are based on incomplete forms of matrix factorisations (Section 2.1), matrix splittings (Section 2.2), or truncated forms of polynomial expansions for the matrix inverse [43].

In this thesis the general, rather than problem-dependent, preconditioning approach is taken. Only splittings and incomplete factorisations are used as preconditioners - these methods are described in the rest of this section. For generality, the matrices are not assumed to be SPD during this description of preconditioning.

Splitting-based Preconditioners

Z is the matrix from a splitting in Section 2.2. For example, from Table 2.1

$$Z = D$$

is the diagonal (or Jacobi) preconditioner, while the lower triangular matrix

$$Z = \frac{1}{\omega}(D - \omega L)$$

is the SOR preconditioner. The SOR preconditioner is not symmetric. If a symmetric preconditioner based on SOR is required, the matrix which is inverted in symmetric successive over-relaxation (SSOR) [31] can be used. Provided the original matrix has non-zero entries on the diagonal, the existence of the inverse of these preconditioning matrices is guaranteed. If A is SPD, then the Jacobi and SSOR preconditioning matrices are SPD.

Incomplete factorisation-based Preconditioners

Rather than computing the full LDM^T factorisation from Section 2.1 (which has the disadvantages given in that section), it is possible to compute unit lower triangular matrices $L, M \in \mathbb{R}^{n\times n}$, and a diagonal matrix $D \in \mathbb{R}^{n\times n}$ such that

$$LDM^T = A - E.$$

where $E \in \mathbb{R}^{n\times n}$ is an error matrix. The factors L, D and M^T give the *incomplete LDM^T factorisation* ($ILDM^T$). One of the simplest forms of the $ILDM^T$ factorisation is one where L and M^T retain the sparsity pattern of the lower- and upper-triangular parts of the original matrix, i.e. during the factorisation, an entry in the matrix is only modified if it is non-zero.

The existence of the incomplete factorisation is shown for the class of M-matrices in [53].

Definition 2.8 *A matrix $A = (a_{ij})$ is an M-*matrix *if $a_{ij} \leq 0$ for $i \neq j$, A is nonsingular, and $A^{-1} \geq 0$.*

Establishing this result involves showing that no breakdown in the algorithm occurs due to the generation of a zero diagonal entry. The essence of the existence proof is that the application of an incomplete factorisation step to an M-matrix results in another matrix in the same class. The class of M-matrices is not particularly useful in the field of finite elements. Fortunately, the existence of the incomplete LDM^T is shown for the wider class of H-matrices in [51].

Definition 2.9 *A matrix $A = (a_{ij})$ is an H-*matrix *if the comparison matrix $B = (b_{ij})$ with $b_{ii} = a_{ii}$, $b_{ij} = -|a_{ij}|$ $(\forall i \neq j)$, is an M-matrix.*

From [87] (Theorem 3.11 Corollary 2), all (irreducible) SPD matrices are H-matrices, so incomplete factorisations exist for this class of matrix.

However, in the SPD case, the preconditioner must be SPD also (to ensure that the preconditioned matrix, $Z^{-1}A$, is similar to an SPD matrix). This requires that the entries in the diagonal matrix D are positive. In [48, 54], this requirement is achieved by neglecting operations which cause the diagonal entry to become non-positive. This "fix" can be applied to the non-symmetric algorithm to ensure that no zero entries are generated on the diagonal (which would lead to a breakdown).

One possible incomplete form of the LDM^T factorisation is given by Algorithm 2.4.

Algorithm 2.4 *$ILDM^T$* FACTORISATION
Given $A \in \mathbb{R}^{n \times n}$, this algorithm computes the incomplete factorisation $LDM^T = A - E$ such that A and $L + M^T$ have the same sparsity pattern. A is overwritten by $L' + M'^T$ where L' and M' are the strictly lower triangular parts of L and M, and the diagonal matrix D is stored in the vector $[d_1, d_2, \ldots, d_n]^T$.

```
for $k = 1, \ldots, n-1$
  for $p = 1, \ldots, k-1$
    $r_p := d_p a_{pk}$
    $w_p := a_{kp} d_p$
  end for
  $d_k := a_{kk}$
  for $p = 1, \ldots, k-1$
    if $(d_k > a_{kp} r_p)$ $d_k := d_k - a_{kp} r_p$
  end for
  for $i = k+1, \ldots, n$
    if $(a_{ik} \neq 0)$ $a_{ik} := \left(a_{ik} - \sum_{p=1}^{k-1} a_{ip} r_p\right) / d_k$
    if $(a_{ki} \neq 0)$ $a_{ki} := \left(a_{ki} - \sum_{p=1}^{k-1} w_p a_{pi}\right) / d_k$
  end for
end for
```

In practice, A is held in sparse storage so the "for" and summation loops in Algorithm 2.4 take on a very specialised structure to exploit the sparsity.

In some cases, the $ILDM^T$ factorisation as described is not powerful enough to yield an adequate rate of convergence. It can be made more powerful by allowing fill-in of some non-zero entries (e.g. those in the matrix positions which neighbour the non-zero entries in the original matrix). This modification is not used in this thesis as it is not deemed necessary for the systems which arise.

2.3.2 Non-symmetric Matrices

The CG solver for systems with a SPD matrix has the desirable properties that

- it uses a low amount of storage (by virtue of the recurrence relations),
- it possesses a minimisation property (so convergence bounds exist),
- it does not require any external parameters (although the preconditioning matrix has to be selected), and
- the most computationally expensive operations required (i.e. matrix-vector multiplications and the solution of preconditioner systems) are relatively easy to implement on parallel architectures.

Ideally, a solver for systems with a non-symmetric matrix should possess all these properties. Unfortunately this is not possible. This is shown in [24] where the class of residual methods is considered. These methods are based on the iteration

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \sum_{j=0}^{i} \eta_{ij} \boldsymbol{r}_j \qquad (\boldsymbol{r}_j = \boldsymbol{b} - A\boldsymbol{x}_j).$$

By looking at the existence of methods of this form that use at most s-term recursion relations and possess either a minimum residual or orthogonal residual property, the following result is obtained.

Theorem 2.2 (Faber and Manteuffel [24]) : *Except for a few anomalies, ideal CG-like methods, defined as methods that*

1. *either possess a minimum residual or orthogonal residual property, and*
2. *can be implemented based on short vector recursions,*

exist only for matrices of the form

$$A = \mathrm{e}^{i\theta}(T + \sigma I), \qquad \text{where} \qquad T = T^T, \quad \theta \in \mathbb{R}, \quad \sigma \in \mathbb{C}.$$

Proof : See [24, 26]. $\square$

Property *1* of the Faber and Manteuffel theorem means that the algorithm is robust and convergence bounds can be obtained, while property *2* means that work and storage requirements per iteration are low and roughly constant.

General non-symmetric matrices do not fall into the class of matrices identified in Theorem 2.2, so CG-like methods for non-symmetric matrices possess either property *1* or property *2*, but not both. This provides the basic distinction between current popular non-symmetric iterative methods.

Methods Possessing a Minimal Residual Property

The most successful method of this type for non-symmetric matrices is Generalized Minimal Residual (GMRES) [70]. This method involves two stages, the first stage generates an l_2-orthonormal basis, $V_k = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k]$, of $\mathcal{K}_k(A, \boldsymbol{r}_0)$ by the Arnoldi process [70] with initial vector $\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$. The Arnoldi method generates the orthonormal basis using Gram-Schmidt orthogonalisation (see e.g. [31]).

If the matrices,

$$\begin{aligned} V_k &= [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k] \in \mathbb{R}^{n \times k} \\ H_k &= [h_{i,j}]_{i,j=1,\ldots,k} \in \mathbb{R}^{k \times k} \quad \text{(upper Hessenberg)}, \end{aligned}$$

are defined (where the $h_{i,j}$ are scalar values generated by the Arnoldi process) then, from [70], these matrices satisfy the relationship

$$H_k = V_k^T A V_k$$

$$A V_k = V_{k+1} \bar{H}_k$$

where

$$\bar{H}_k = \left[\begin{array}{c|c} \multicolumn{2}{c}{H_k} \\ \hline \mathbf{0}^T & h_{k+1,k} \end{array} \right] \in \mathbb{R}^{(k+1) \times k}.$$

In the second stage of GMRES, an approximation to the solution is generated from the orthonormal basis by imposing a minimisation condition. In the SPD case (CG), the minimisation is carried out in the A-norm,

$$\|\boldsymbol{w}\|_A = \sqrt{\boldsymbol{w}^T A \boldsymbol{w}}.$$

For general non-symmetric A, this does not define a norm so the minimum residual condition is imposed in the 2-norm, i.e.

$$\begin{aligned} \|\boldsymbol{r}_k\|_2 &= \min_{\mathbf{z}-\mathbf{x}_0 \in \mathcal{K}_k(A,\mathbf{r}_0)} \|\boldsymbol{b} - A\boldsymbol{z}\|_2 \\ &= \min_{\mathbf{z} \in \mathcal{K}_k(A,\mathbf{r}_0)} \|\boldsymbol{b} - A(\boldsymbol{x}_0 + \boldsymbol{z})\|_2 \\ &= \min_{\mathbf{z} \in \mathcal{K}_k(A,\mathbf{r}_0)} \|\boldsymbol{r}_0 - A\boldsymbol{z}\|_2. \end{aligned}$$

From [69], this is equivalent to

$$\boldsymbol{r}_k \perp \operatorname{span}(A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots, A^k\boldsymbol{r}_0) = A\mathcal{K}_k(A, \boldsymbol{r}_0).$$

Defining $\beta = \|\boldsymbol{r}_0\|_2$, then after k steps of the Arnoldi process,

$$\begin{aligned} \min_{\mathbf{z} \in \mathcal{K}_k(A,\mathbf{r}_0)} \|\boldsymbol{r}_0 - A\boldsymbol{z}\|_2 &= \min_{\mathbf{y} \in \mathbb{R}^k} \|\beta \boldsymbol{v}_1 - AV_k \boldsymbol{y}\|_2 \\ &= \min_{\mathbf{y} \in \mathbb{R}^k} \|V_{k+1}(\beta \boldsymbol{e}_1^{k+1} - \bar{H}_k \boldsymbol{y})\|_2 \\ &= \min_{\mathbf{y} \in \mathbb{R}^k} \|\beta \boldsymbol{e}_1^{k+1} - \bar{H}_k \boldsymbol{y}\|_2 \end{aligned}$$

since the columns of V_{k+1} are orthonormal in the 2-norm.

Algorithm 2.5 GMRES

Given an initial iterate, $\boldsymbol{x}_0 \in \mathbb{R}^n$, this algorithm generates iterates, $\boldsymbol{x}_k \in \mathbb{R}^n$, for the solution of (2.1) by the generalized minimal residual method.

```
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$
$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
*Generate orthonormal basis using Gram-Schmidt orthogonalisation:*
for $j = 1, 2, \ldots,$ until satisfied (at $j = k$)
    for $i = 1, \ldots, j$
        $h_{i,j} := \boldsymbol{v}_i^T A \boldsymbol{v}_j$
    end for
```

```
        $\hat{\boldsymbol{v}}_{j+1} := A\boldsymbol{v}_j - \sum_{i=1}^{j} h_{i,j}\boldsymbol{v}_i$
        $h_{j+1,j} := \|\hat{\boldsymbol{v}}_{j+1}\|_2$
        $\boldsymbol{v}_{j+1} := \hat{\boldsymbol{v}}_{j+1}/h_{j+1,j}$
    end for
    *Form the approximate solution:*
    $\boldsymbol{x}_k := \boldsymbol{x}_0 + V_k\boldsymbol{y}_k$ where $\boldsymbol{y}_k$ minimises $\|\beta\boldsymbol{e}_1^{k+1} - \bar{H}_k\boldsymbol{y}\|_2$ over $\boldsymbol{y} \in \mathbb{R}^k$
```

There are two questions that need to be addressed on the operation of Algorithm 2.5. These are

- how is the convergence monitored if the solution is only constructed after the process is stopped ?
- how is the minimisation problem solved in the formation of the approximate solution ?

The latter question is addressed first.

Solution of the Minimisation Problem

The method suggested in [70] involves the QR-factorisation of the upper Hessenberg matrix, $\bar{H}_k$. This can be computed efficiently by Givens plane rotations or fast Householder transformations (see [31] for more detail on these methods) - the latter method is used in the implementation in this work. These methods allow the factorisation of $\bar{H}_k$ to be updated progressively as each column appears (i.e. at every step of the Arnoldi process).

The progressive QR-factorisation generates,

$$\bar{H}_k = Q_k R_k$$

where $Q_k \in \mathbb{R}^{(k+1)\times(k+1)}$ has orthonormal columns (i.e. $Q_k^T Q_k = I$) and $R_k \in \mathbb{R}^{(k+1)\times k}$ is an upper triangular matrix. With this factorisation, the minimisation problem becomes

$$\begin{aligned} \min_{\mathbf{y}\in\mathbb{R}^k} \|\beta\boldsymbol{e}_1^{k+1} - \bar{H}_k\boldsymbol{y}\|_2 &= \min_{\mathbf{y}\in\mathbb{R}^k} \|Q_k^T(\beta\boldsymbol{e}_1^{k+1} - \bar{H}_k\boldsymbol{y})\|_2 \\ &= \min_{\mathbf{y}\in\mathbb{R}^k} \|\boldsymbol{g}_k - R_k\boldsymbol{y}\|_2 \end{aligned}$$

where $\boldsymbol{g}_k = Q_k^T \beta \boldsymbol{e}_1^{k+1} = [g_1, g_2, \ldots, g_{k+1}]^T$. Due to the structure of $\bar{H}_k$, the last row of R_k is zero so the solution of the minimisation problem is

$$\boldsymbol{y}_k = \bar{R}_k^{-1} \bar{\boldsymbol{g}}_k,$$

where $\bar{R}_k$ is the leading principal $(k \times k)$ submatrix of R_k and $\bar{\boldsymbol{g}}_k = [g_1, g_2, \ldots, g_k]^T$.

Monitoring of Convergence

As already stated, a stopping criterion such as (2.12) is used with iterative methods to determine whether the current iterate is satisfactory. GMRES appears to have the drawback that k must be selected, the orthonormal basis which is the set of columns of V_k must be generated, and the minimisation problem on this basis must be solved in order to generate the iterate so that the residual can be computed (for use in the convergence test). This suggests that unnecessary computations are performed if k is not chosen correctly. However, as a progressive QR-factorisation is used, it is possible to monitor the size of the residual (at no extra cost) during the Arnoldi process since the 2-norm of the residual is given by

$$\begin{aligned} \|\boldsymbol{r}_k\|_2 &= \min_{\mathbf{y} \in \mathbb{R}^k} \|\boldsymbol{g}_k - R_k \boldsymbol{y}\|_2 \\ &= \|\boldsymbol{g}_k - R_k \boldsymbol{y}_k\|_2 \\ &= \|g_{k+1} \boldsymbol{e}_{k+1}^{k+1}\|_2 \\ &= |g_{k+1}|. \end{aligned}$$

The *classical* Gram-Schmidt method is used to generate the orthonormal basis in Algorithm 2.5. In practice, the *modified* Gram-Schmidt method [31] is preferred due to better numerical stability (each old vector is subtracted from the new vector as soon as its component is computed, rather than being accumulated into a sum and then subtracted - see Algorithm 2.6).

With either implementation, a consequence of the use of Gram-Schmidt orthogonalisation to generate a basis of $\mathcal{K}_k(A, \boldsymbol{r}_0)$ is that all the vectors in the basis must be stored in order to generate the next one, and also the work requirement for generating the next vector in the basis grows as the size of the basis increases. The size of the least squares problem in the minimisation also grows with iteration count. Apart from a matrix-vector product and a preconditioner solve, in the i^{th} iteration, this algorithm requires approximately $(i+1)$ vector-vector products and $(i+1)$ SAXPYs. Apart from A, $\boldsymbol{x}$ and $\boldsymbol{b}$, the storage requirement is $(i+3)$ n-vectors.

Both the operations count and the required storage become prohibitive if a large number of iterations is required. As stated in [70], it is possible to avoid these problems by using a *restarted* version of the method (see Algorithm 2.6).

Given a fixed integer m and an initial iterate, the restarted GMRES method computes a solution with minimal 2-norm over $\mathcal{K}_m(A, \boldsymbol{r}_0)$. If this is not sufficiently accurate, then the process is restarted using the previous solution $\boldsymbol{x}_m$ as the initial iterate.

Algorithm 2.6 GMRES(m)

Given an initial iterate, $\boldsymbol{x}_0 \in \mathbb{R}^n$, this algorithm generates iterates, $\boldsymbol{x}_i \in \mathbb{R}^n$, for the solution of (2.1) by the restarted generalized minimal residual method with restart period m.

```
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$
$\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$
label RESTART
*Generate orthonormal basis using modified Gram-Schmidt:*
for $j = 1, 2, \ldots, m$
    $\hat{\boldsymbol{v}}_{j+1} := A\boldsymbol{v}_j$
    for $i = 1, \ldots, j-1$
        $h_{i,j} := \hat{\boldsymbol{v}}_{j+1}^T \boldsymbol{v}_i$
        $\hat{\boldsymbol{v}}_{j+1} := \hat{\boldsymbol{v}}_{j+1} - h_{i,j}\boldsymbol{v}_i$
    end for
    $h_{j+1,j} := \|\hat{\boldsymbol{v}}_{j+1}\|_2$
    $\boldsymbol{v}_{j+1} := \hat{\boldsymbol{v}}_{j+1}/h_{j+1,j}$
end for
*Form the approximate solution:*
```

```
$\boldsymbol{x}_m := \boldsymbol{x}_0 + V_k\boldsymbol{y}_m$ where $\boldsymbol{y}_m$ minimises $\|\beta\boldsymbol{e}_1^{k+1} - \bar{H}_m\boldsymbol{y}\|_2$ over $\boldsymbol{y} \in \mathbb{R}^m$
$\boldsymbol{r}_m := \boldsymbol{b} - A\boldsymbol{x}_m$
if not satisfied
    $\boldsymbol{x}_0 := \boldsymbol{x}_m$
    $\boldsymbol{v}_1 := \boldsymbol{r}_m/\|\boldsymbol{r}_m\|_2$
    goto RESTART
end if
```

Algorithm 2.6 has the same work and storage requirements per iteration as Algorithm 2.5 but there is a controllable upper limit on these quantities over the whole process.

In [70], a bound on the value of m which guarantees convergence of the restarted method is given. This bound is impractical to use because it involves spectral data of the coefficient matrix. Also, it is generally not sharp, so it is likely that convergence takes place for a much lower restart parameter. In practice, the choice of m is usually based on experience and on such factors as the size of available fast memory.

Methods Based on Short Term Recurrences

Since, from Theorem 2.2, methods based on short term recurrences cannot possess a minimum residual property or an orthogonal residual property, they are constructed so that the residuals in the Krylov subspace satisfy the Petrov-Galerkin condition,

$$\boldsymbol{r}_i \perp \{\hat{\boldsymbol{r}}_0, (A^T)\hat{\boldsymbol{r}}_0, (A^T)^2\hat{\boldsymbol{r}}_0, \ldots, (A^T)^{i-1}\hat{\boldsymbol{r}}_0\},$$

where $\hat{\boldsymbol{r}}_0$ is an initial pseudo-residual vector.

As shown in Section 2.3.1, the conjugate gradient method (for A SPD) is related closely to the symmetric Lanczos tridiagonalisation process. A possible approach, when A is non-symmetric, is to base the solver on the *non-symmetric* Lanczos tridiagonalisation process - this leads to the bi-conjugate gradient method (Bi-CG) [25, 49]. A brief description of the relevant properties of the non-symmetric Lanczos process is included for use later in the thesis.

Given initial vectors, $\boldsymbol{v}_1, \boldsymbol{w}_i \neq \boldsymbol{0}$, after the i^{th} step of the process, the vectors

$\{\boldsymbol{v}_i\}$ and $\{\boldsymbol{w}_j\}$ are generated and the following relationships hold,

$$\begin{aligned} AV_i &= V_iH_i + \boldsymbol{v}_{i+1}\boldsymbol{e}_i^{i,T} = V_{i+1}H_{i+1} \\ A^TW_i &= W_iH_i^T + \boldsymbol{w}_{i+1}\boldsymbol{e}_i^{i,T} = W_{i+1}H_{i+1}^T \end{aligned} \tag{2.16}$$

where

$$H_i = \begin{bmatrix} \tilde{\alpha}_1 & \tilde{\beta}_1 & & & & \\ \tilde{\gamma}_1 & \tilde{\alpha}_2 & \tilde{\beta}_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \tilde{\gamma}_{i-2} & \tilde{\alpha}_{i-1} & \tilde{\beta}_{i-1} \\ & & & & \tilde{\gamma}_{i-1} & \tilde{\alpha}_i \end{bmatrix} \in \mathbb{R}^{i\times i}$$

$$\begin{aligned} V_i &= [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i] \in \mathbb{R}^{n\times i} \\ W_i &= [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_i] \in \mathbb{R}^{n\times i} \end{aligned}$$

and

$$\begin{aligned} V_i^T\boldsymbol{w}_{i+1} &= \mathbf{0} \\ W_i^T\boldsymbol{v}_{i+1} &= \mathbf{0} \\ \text{Range}(V_i) &= \mathcal{K}_i(A, \boldsymbol{v}_1) \\ \text{Range}(W_i) &= \mathcal{K}_i(A^T, \boldsymbol{w}_1). \end{aligned}$$

As noted in [92] (p389), the non-symmetric Lanczos process encounters a *fatal breakdown* if $\boldsymbol{w}_i^T\boldsymbol{v}_i = 0$ with neither $\boldsymbol{v}_i = \mathbf{0}$ or $\boldsymbol{w}_i = \mathbf{0}$. Except in the very special case of an incurable breakdown [80], this problem can be overcome by using block pivots in the algorithm whenever the use of scalar pivots would be dangerous, this is the same as performing the bi-orthogonalisation on blocks of vectors. This approach is known as the look-ahead Lanczos method [60]. It can also be used to avoid near-fatal breakdown, i.e. $\boldsymbol{w}_i^T\boldsymbol{v}_i \approx 0$ without either $\boldsymbol{v}_i \approx \mathbf{0}$ or $\boldsymbol{w}_i \approx \mathbf{0}$ - see [27].

In Bi-CG, the non-symmetric Lanczos process is used to construct approximations so that the residual vector, $\boldsymbol{r}_i$, is orthogonal to a set of pseudo-residual vectors $\{\hat{\boldsymbol{r}}_j\}_{(j=0,\ldots,i-1)}$ and, vice versa, $\hat{\boldsymbol{r}}_i \perp \{\boldsymbol{r}_j\}_{(j=0,\ldots,i-1)}$. Since the underlying non-symmetric process is based on two three-term recurrence relations (2.16), then the Bi-CG iteration also uses two three-term recurrence relations for the rows $\boldsymbol{r}_i$ and $\hat{\boldsymbol{r}}_i$. As with CG, the Bi-CG residual and pseudo-residual vectors can be expressed in the residual polynomial form as,

$$\boldsymbol{r}_i = \varphi_i(A)\boldsymbol{r}_0 \quad , \quad \hat{\boldsymbol{r}}_i = \varphi_i(A^T)\hat{\boldsymbol{r}}_0,$$

where $\varphi_i(\cdot)$ is a polynomial of degree $\leq i$.

In the case of convergence, both these vectors tend towards zero but only the convergence of $\boldsymbol{r}_i$ is exploited, $\hat{\boldsymbol{r}}_j$ only being required for the calculation of iteration parameters. The iteration parameters in Bi-CG are generated by inner product relations such as the bi-orthogonality condition

$$(\boldsymbol{r}_i, \hat{\boldsymbol{r}}_j) = (\varphi_i(A)\boldsymbol{r}_0, \varphi_j(A^T)\hat{\boldsymbol{r}}_0) = 0 \quad (i \neq j), \tag{2.17}$$

where $(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b}$ is the standard inner product. This inner product relation can be written as

$$(\boldsymbol{r}_i, \hat{\boldsymbol{r}}_j) = (\varphi_j(A)\varphi_i(A)\boldsymbol{r}_0, \hat{\boldsymbol{r}}_0) = 0 \quad (i \neq j). \tag{2.18}$$

Bi-CG has been superseded by a faster converging variant - the conjugate gradient-squared method (CG-S) of Sonneveld [76]. In CG-S, all the convergence effort is directed to $\boldsymbol{r}_i$ by constructing the iteration parameters using the bi-orthogonality condition in the form (2.18). In this way, the $\hat{\boldsymbol{r}}_j$ do not need to be formed and there is no requirement for A^T which would require lots of *memory jumping* in e.g. compressed row storage [5].

Algorithm 2.7 PRECONDITIONED CG-S

Given an initial iterate, $\boldsymbol{x}_0 \in \mathbb{R}^n$, and a preconditioning matrix, $Z \in \mathbb{R}^{n\times n}$, this algorithm generates iterates, $\boldsymbol{x}_i \in \mathbb{R}^n$, for the solution of (2.1) by the preconditioned conjugate gradient squared method.

```
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$
$\hat{\boldsymbol{r}}_0$ is an arbitrary vector such that $(\boldsymbol{r}_0^T\hat{\boldsymbol{r}}_0) \neq 0$
$\rho_0 := 1$; $\boldsymbol{p} := \boldsymbol{q} := \boldsymbol{0}$
for $i = 1, 2, \ldots$ until satisfied
    $\rho_i := (\hat{\boldsymbol{r}}_0^T\boldsymbol{r}_{i-1})$
```

```
        $\beta := \rho_i / \rho_{i-1}$
        $\boldsymbol{u} = \boldsymbol{r}_{i-1} + \beta \boldsymbol{q}$
        $\boldsymbol{p} := \boldsymbol{u} + \beta(\boldsymbol{q} + \beta \boldsymbol{p})$
        $\boldsymbol{y} := Z^{-1}\boldsymbol{p}$
        $\boldsymbol{v} := A\boldsymbol{y}$
        $\alpha := \rho_i / (\hat{\boldsymbol{r}}_0^T \boldsymbol{v})$
        $\boldsymbol{q} := \boldsymbol{u} - \alpha \boldsymbol{v}$
        $\boldsymbol{z} := Z^{-1}(\boldsymbol{u} + \boldsymbol{q})$
        $\boldsymbol{x}_i := \boldsymbol{x}_{i-1} + \alpha \boldsymbol{z}$
        $\boldsymbol{r}_i := \boldsymbol{r}_{i-1} - \alpha A \boldsymbol{z}$
    end for
```

In this thesis, $\hat{\boldsymbol{r}}_0$ is taken as $\boldsymbol{r}_0$. Apart from A, $\boldsymbol{x}$ and $\boldsymbol{b}$, the preconditioned CG-S algorithm requires 9 n-vectors storage. At each iteration, it requires 2 matrix-vector multiplications, 2 preconditioner solves, 2 vector-vector products and 7 SAXPYs.

The CG-S residual vectors are given by,

$$\boldsymbol{r}_i = \varphi_i^2(A)\boldsymbol{r}_0,$$

so, if $\varphi_i(A)$ is viewed as a Bi-CG contraction operator (in the case of convergence), then the CG-S contraction operator, $\varphi_i^2(A)$, is twice as effective.

However, situations arise in the iteration process where $\varphi_i(A)$ is not a contraction operator and spikes occur in the convergence history. In practice CG-S is found to have a rather erratic convergence behaviour, particularly when the starting iterate is close to the solution (as is the case in many transient and non-linear problems). Although this tends not to slow the overall convergence rate of the method, it can allow rounding errors to cause a breakdown in the method in finite precision.

Bi-CGSTAB [84] is a variant of CG-S which has a more smoothly varying convergence history. This method comes from a generalisation of the Bi-CG and CG-S methods; the orthogonality conditions (2.17) and (2.18) are imposed in those methods, but other iteration methods can be generated by choosing

$$\boldsymbol{r}_i \perp \left\{ \tilde{\varphi}_j(A^T)\hat{\boldsymbol{r}}_0 \right\}_{(j=0,\ldots,i-1)},$$

where $\tilde{\varphi}_j(\cdot)$ is another polynomial of degree $\leq j$, (in Bi-CG, $\tilde{\varphi}_j(\cdot) = \varphi_j(\cdot)$).

The choice of $\tilde{\varphi}_i(\cdot)$ means that there is a degree of freedom with which to obtain a more smoothly varying iteration history. The only constraint on the choice of this polynomial is that it should allow the use of short recursion relations in the resulting algorithm to keep the computational work and storage requirements small at each iteration.

In Bi-CGSTAB, $\tilde{\varphi}_i(\cdot)$ is of the form

$$\tilde{\varphi}_i(A) = (1 - \omega_1 A)(1 - \omega_2 A) \ldots (1 - \omega_i A), \tag{2.19}$$

where ω_j $(j = 1, \ldots, i)$ are constants determined by a local steepest descent step. As with CG-S, inner product conditions such as

$$(\tilde{\varphi}_j(A)\varphi_i(A)\boldsymbol{r}_0, \hat{\boldsymbol{r}}_0) = 0 \qquad (i \neq j)$$

are used to avoid the need to store the pseudo-residual vectors and compute with A^T.

Algorithm 2.8 Preconditioned Bi-CGSTAB-P

Given an initial iterate, $\boldsymbol{x}_0 \in \mathbb{R}^n$, and a preconditioning matrix, $Z \in \mathbb{R}^{n \times n}$, this algorithm generates iterates, $\boldsymbol{x}_i \in \mathbb{R}^n$, for the solution of (2.1) by the preconditioned Bi-CGSTAB method.

```
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$;
$\hat{\boldsymbol{r}}_0$ is an arbitrary vector such that $\hat{\boldsymbol{r}}_0^T \boldsymbol{r}_0 \neq 0$
$\rho_0 := \alpha := \omega := 1$; $\rho_1 := (\hat{\boldsymbol{r}}_0^T \boldsymbol{r}_0)$; $\boldsymbol{v} := \boldsymbol{p} := \boldsymbol{0}$
for $i = 1, 2, \ldots$ until satisfied
  $\beta := (\rho_i / \rho_{i-1})(\alpha / \omega)$
  $\boldsymbol{p} := \boldsymbol{r}_{i-1} + \beta(\boldsymbol{p} - \omega \boldsymbol{v})$
  $\boldsymbol{y} := Z^{-1}\boldsymbol{p}$
  $\boldsymbol{v} := A\boldsymbol{y}$
  $\alpha := \rho_i / (\boldsymbol{r}_0^T \boldsymbol{v})$
  $\boldsymbol{s} := \boldsymbol{r}_{i-1} - \alpha \boldsymbol{v}$
  $\boldsymbol{z} := Z^{-1}\boldsymbol{s}$
  $\boldsymbol{t} := A\boldsymbol{z}$
```

```
        $\omega := (\boldsymbol{t}^T\boldsymbol{s})/(\boldsymbol{t}^T\boldsymbol{t})$
        $\rho_{i+1} := -\omega(\hat{\boldsymbol{r}}_0^T\boldsymbol{t})$
        $\boldsymbol{x}_i := \boldsymbol{x}_{i-1} + \alpha\boldsymbol{y} + \omega\boldsymbol{z}$
        $\boldsymbol{r}_i := \boldsymbol{s} - \omega_i\boldsymbol{t}$
    end for
```

Algorithm 2.8 is similar to the one given by van der Vorst [84], and incorporates some of the modifications suggested in the original paper. In this thesis, $\hat{\boldsymbol{r}}_0$ is taken as $\boldsymbol{r}_0$.

Because of the form of the computation of ω_i used, i.e.

$$\omega_i = \frac{\boldsymbol{t}^T\boldsymbol{s}}{\boldsymbol{t}^T\boldsymbol{t}},$$

this is the Bi-CGSTAB-P variant. The full Bi-CGSTAB algorithm has

$$\omega_i = \frac{(Z^{-1}\boldsymbol{t})^T Z^{-1}\boldsymbol{s}}{(Z^{-1}\boldsymbol{t})^T Z^{-1}\boldsymbol{t}}.$$

Obviously the P-variant is computationally less expensive than the full algorithm, and practical experience has shown this variant gives a smoother convergence behaviour; for these reasons it is the preferred algorithm in this work.

Apart from A, $\boldsymbol{x}$ and $\boldsymbol{b}$, it requires 8 n-vectors of storage. Each iteration requires 2 matrix-vector multiplications, 2 preconditioner solves, 4 vector-vector products and 6 SAXPYs.

An apparent exception to Theorem 2.2 is the quasi-minimal residual method (QMR) [28] which is based on three-term recurrence relations and also performs a form of minimisation on the norm of the residual. Using the notation from the underlying non-symmetric Lanczos process previously described, an approximation to the solution of (2.1) can be obtained at the i^{th} iteration as

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + V_i\boldsymbol{y}_i$$

for some $\boldsymbol{y}_i \in \mathbb{R}^i$. Hence

$$\begin{aligned}
\boldsymbol{b} - A\boldsymbol{x}_i &= \boldsymbol{b} - A\boldsymbol{x}_0 - AV_i\boldsymbol{y}_i \\
\boldsymbol{r}_i &= \boldsymbol{r}_0 - AV_i\boldsymbol{y}_i \\
&= V_{i+1}\left(\tilde{\beta}_1\boldsymbol{e}_1^i - H_{i+1}\boldsymbol{y}_i\right)
\end{aligned}$$

When this stage is reached in the GMRES method, the 2-norm of the residual is minimised and V_{i+1}, being a unitary matrix generated by the Arnoldi process, drops out of the 2-norm. In the Lanczos procedure, V_{i+1}, is not a unitary matrix so it cannot be taken out of the 2-norm. Instead, to minimize in the 2-norm here would require V_{i+1} to be stored and used explicitly, which is costly both in terms of storage and number of floating point operations. To avoid this, the QMR approach is to minimise the 2-norm of the "quasi-residual" $V_{i+1}^{-1}\boldsymbol{r}_i$ at the i^{th} iteration i.e.,

$$\|V_{i+1}^{-1}\boldsymbol{r}_i\|_2 \left(= \|W_{i+1}^{T}\boldsymbol{r}_i\|_2\right) = \min_{\mathbf{y}\in\mathbb{R}^i} \|\tilde{\beta}_1\boldsymbol{e}_1^i - H_{i+1}\boldsymbol{y}\|_2$$

which is equivalent to minimising the residual in a norm that changes with iteration number. Hence this is not a true minimisation and, because of this, QMR does not contradict Theorem 2.2.

2.4 Closing comments

This chapter is by no means exhaustive. It only covers the methods used later in the thesis, and the relevant theory for these methods. Among the methods for the solution of linear systems that have not been described are those using Chebyshev acceleration [37] and the normal equations [39].

The preconditioned CG method is used to solve all the SPD systems in this thesis. The particular type of preconditioning used is determined by numerical experiment in Chapter 5.

The main non-symmetric solver used is Bi-CGSTAB. Chapter 6 is devoted to investigating its performance and selecting a preconditioner. Also in Chapter 6, some techniques for enhancing the performance of iterative solvers are described.

The next chapter describes the origin of the linear systems which arise in this thesis.

Chapter 3

Governing Equations and their Numerical Solution

The purpose of the first section of this chapter is to introduce the governing equations for non-passive transport of a single-species, non-reactive contaminant by a fluid in a saturated porous medium. The meaning of the hydrological terms in the preceding sentence is as follows:

- *non-passive transport* - the properties of the fluid (e.g. density, viscosity) are affected by the presence of any contaminants.
- *single-species* - there is only one contaminant in the system.
- *non-reactive* - the contaminant does not undergo any chemical reactions with the fluid or the solid "matrix" of the porous medium. This is also taken to mean that the contaminant does not undergo any radioactive changes.
- *saturated* - the pore space (i.e. the space not occupied by the solid "matrix" of the porous medium) is completely filled by the fluid. (In an unsaturated porous medium, the pore space is occupied by fluid and air.)

In order to model this system mathematically, a *continuum approach* (see e.g. [52]) is adopted. In the continuum approach, the fluid is treated as a continuous distribution of matter with no empty space. This is normally justifiable because the number of molecules involved is vast and the distance between them is very small. The continuum approach fails when either of these conditions is not satisfied, e.g. in a gas at extremely low pressure. Properties of the fluid such as pressure and viscosity, although molecular in origin, are ascribed to the continuum.

The governing equations which constitute part of the mathematical model are derived by applying basic physical laws such as conservation of mass, momentum and energy, as well as constitutive relations which define the behaviour of the particular fluid and contaminant involved.

Due to the continuous interaction of the entities in the system, the governing equations are coupled together, i.e. they cannot be solved independently of each other. A technique that allows the individual equations in the system to be solved separately is described in Section 3.2.

In general the governing equations are partial differential equations which cannot be solved analytically, so they must be solved approximately by numerical means. This involves replacing the equations by discrete analogues and solving these instead. In Section 3.3, methods for discretising the governing equations are described, together with some properties of these methods. Finally, in Section 3.4, the discretisation methods are applied to the governing equations.

3.1 The Governing Equations

Since the contaminant is a single-species and non-reactive then, assuming that the porous medium is fixed, there are only two entities in the system - fluid and contaminant.

The transport is non-passive. In this work it is assumed that only the density of the fluid is affected by the presence of the contaminant. An example of this kind of interaction is salt in water - pure water has a density of 1000kg/m^3, while fully saturated salt-water has a density of 1024.99kg/m^3. This is a useful example since it involves precisely the entities (water and salt) in the physical system for saline intrusion, and a saline intrusion problem is used as a major test case in this thesis. From this point on, the term "fluid" is used to denote water with dissolved contaminant.

An additional assumption in the model is that the effect of any temperature variations is negligible.

From [6], a mathematical model for the type of transport previously described can be formulated in terms of

- Darcy's law (a form of the momentum balance equation) for the fluid,
- a mass balance equation for the fluid,
- a mass balance equation for the contaminant,
- a constitutive equation relating the fluid density to the contaminant concentration,

together with a definition of the geometry of the domain (and its boundaries) and the relevant initial and boundary conditions.

It is convenient to write the governing equations in terms of either the pressure or the piezometric head defined as

$$h = \frac{p}{\rho g} + z,$$

where h is the piezometric head (L)
p is the pressure ($\mathrm{ML^{-1}T^{-2}}$)
$\rho = \rho(c)$ is fluid density ($\mathrm{ML^{-3}}$)
c is the dimensionless contaminant concentration[1]
g is acceleration due to gravity ($\mathrm{LT^{-2}}$)
z is the elevation above a datum (positive upwards) (L) .

The variables p and h are convenient since they are continuous across the interface between the contaminated and contaminant-free regions and so apply as single state variables over the whole domain.

3.1.1 Darcy's Law

If it is assumed that the porous medium is non-deformable, and that the internal friction inside the fluid and inertial effects are negligible, then the application of

[1]The dimensionless contaminant concentration is the actual contaminant concentration divided by the maximum contaminant concentration.

conservation of momentum to a representative volume of the fluid gives a general form of Darcy's law [6],

$$\boldsymbol{q} = -\underline{\boldsymbol{k}}\left(\boldsymbol{\nabla} p + \rho g \boldsymbol{\nabla} z\right)/\mu,$$

where $\boldsymbol{q}$ is the Darcy velocity vector (LT^{-1})
$\underline{\boldsymbol{k}}$ is the permeability tensor (L^2)
μ is the dynamic viscosity of the fluid ($\mathrm{ML}^{-1}\mathrm{T}^{-1}$) .

This equation indicates that the fluid moves under the influence of pressure differences (represented by the $\boldsymbol{\nabla} p$ term) and gravity (represented by the $g\boldsymbol{\nabla} z$ term).

In this work, it is assumed that variations in the viscosity of the fluid are negligible. Therefore μ is constant and equal to the viscosity of the non-contaminated fluid, μ_0. Hence, the form of Darcy's law used in this thesis is

$$\boldsymbol{q} = -\underline{\boldsymbol{K}}\left(\frac{\boldsymbol{\nabla} p}{\rho g} + \boldsymbol{\nabla} z\right) \tag{3.1}$$

where $\underline{\boldsymbol{K}} = \underline{\boldsymbol{k}}\rho g/\mu_0$ is the hydraulic conductivity tensor (LT^{-1}).

3.1.2 Fluid Mass Balance Equation

Mass balance equations are usually derived by conservation of mass considerations on a relative effective volume of the continuum. From [6], in the absence of source terms, the fluid mass balance equation for saturated flow is

$$\phi\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla}.\rho\boldsymbol{q} = 0, \tag{3.2}$$

where ϕ is the porosity of the porous medium.

3.1.3 Fluid Continuity Equation

In practice, the fluid continuity equation is solved for the pressure. This is obtained by eliminating the Darcy velocity vector from (3.2) using (3.1) to give

$$\phi\frac{\partial \rho}{\partial t} - \boldsymbol{\nabla}.\underline{\boldsymbol{K}}\left(\frac{\boldsymbol{\nabla} p}{g} + \rho\boldsymbol{\nabla} z\right) = 0. \tag{3.3}$$

3.1.4 Contaminant Mass Balance Equation

Assuming dispersion is adequately modelled by Fick's law then, from [6], the contaminant mass balance equation for saturated flow (in the absence of source terms) is

$$\phi\frac{\partial(\rho c)}{\partial t}+\boldsymbol{\nabla}.\left(\rho c\boldsymbol{q}-\phi\underline{\boldsymbol{D}}\boldsymbol{\nabla}(\rho c)\right)=0, \tag{3.4}$$

where $\underline{\boldsymbol{D}}=\underline{\boldsymbol{D}}(\boldsymbol{q})$ is the dispersion tensor ($\mathrm{L}^2\mathrm{T}^{-1}$). The (i,j) entry in the dispersion tensor is related to the i and j components of the fluid velocity vector, $\boldsymbol{v}$ $(=\boldsymbol{q}/\phi)$, by

$$D_{ij}=\left(\alpha_T|\boldsymbol{v}|+D_m\right)\delta_{ij}+\left(\alpha_L-\alpha_T\right)\frac{v_iv_j}{|\boldsymbol{v}|}, \tag{3.5}$$

where α_T is the transverse dispersivity (L)

α_L is the longitudinal dispersivity (L)

D_m is the coefficient of molecular diffusion ($\mathrm{L}^2\mathrm{T}^{-1}$)

$\delta_{ij}=1$ for $i=j$, zero otherwise.

α_L indicates the amount of dispersion which occurs in the direction of flow, α_L indicates the amount of dispersion which occurs transverse to the direction of flow, and D_m represents the amount of dispersion which occurs due purely to molecular diffusion.

Equation (3.4) indicates that the contaminant is transported by the processes of advection (represented by the $\boldsymbol{\nabla}.\rho c\boldsymbol{q}$ term) and diffusion (represented by the $\boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\boldsymbol{\nabla}(\rho c)$ term).

In [82] it is noted that the contaminant mass balance equation (3.4) is undefined when the Darcy velocity is zero due to singularities in the dispersion tensor (3.5). In practice, the dispersion tensor is unlikely to be evaluated at a stagnation point, but the possible existence of a problem is acknowledged (particularly at interfaces in a highly heterogeneous medium).

3.1.5 Constitutive Equation

The constitutive equation relates the fluid density to the contaminant concentration. The actual constitutive equation depends on the fluid, the contaminant and the conditions.

In this work, the fluid is water, the contaminant is salt and the conditions are typical coastal ones. Common forms of the constitutive equation in this case are, from [72], a linear relationship,

$$\rho = \rho_0 + \epsilon c, \tag{3.6}$$

where ρ_0 is the density of the non-contaminated fluid (ML^{-3})

ϵ is a constant (ML^{-3}) ,

or, from [82], a logarithmic relationship,

$$\rho = \rho_0 \exp(\gamma c),$$

where γ is a constant obtained from laboratory experiments.

In this work, the linear relationship (3.6) is used.

3.1.6 Initial and Boundary Conditions

For the fluid continuity equation (3.3), the initial condition is

$$h = h_0 \qquad \text{on } \Omega,$$

where Ω is the spatial domain and the boundary conditions are

$$\begin{aligned} h &= \tilde{h} \quad &\text{on} \quad &\Gamma_1 \quad \text{(prescribed head)} \\ -\boldsymbol{q}.\boldsymbol{n} &= q_n \quad &\text{on} \quad &\Gamma_2 \quad \text{(prescribed fluid flux)}, \end{aligned}$$

where $\boldsymbol{n}$ is the unit outward normal vector and $\Gamma = \Gamma_1 + \Gamma_2$ is the boundary of Ω. For the contaminant continuity equation (3.4), the initial condition is

$$c = c_0 \qquad \text{on } \Omega,$$

and the boundary conditions are

$$\begin{aligned} c &= \tilde{c} \quad &\text{on} \quad &\Gamma_1' \quad \text{(prescribed concentration)} \\ -\phi \underline{\boldsymbol{D}} \boldsymbol{\nabla}(\rho c).\boldsymbol{n} &= q_n^c \quad &\text{on} \quad &\Gamma_2' \quad \text{(prescribed dispersive solute mass flux)} \end{aligned}$$

(where $\Gamma_1' + \Gamma_2' = \Gamma$).

The convention used for the sign of both the prescribed fluid flux and the prescribed dispersive solute mass flux is that they are positive if the flow is directed into the region, Ω.

3.2 Coupling of the Governing Equations

Due to the nature of the interaction between the fluid density (ρ), the Darcy velocity ($\boldsymbol{q}$) and the contaminant concentration (c), the governing equations in Section 3.1 are coupled and cannot be solved independently of each other.

However, it is possible to treat the equations individually by the use of an iterative approach. In such an approach, an initial approximation is made for one of the state variables, then the governing equations are solved in such an order that the other state variables are generated and finally an approximation is made to the original state variable. Repeating this process restores the coupling and, assuming the iteration converges, a valid solution state (in which the fluid density and contaminant concentration match) is obtained.

The coupling iteration approach in this work is one which is commonly used in the literature e.g. [29, 40].

Algorithm 3.1 Coupling of Governing Equations

This algorithm is an outline of the iterative approach used to couple the governing equations so that they can be solved independently of each other.

1. Make an approximation for the fluid density.
2. Using the approximation for the fluid density, solve the fluid continuity equation (3.3) for an approximation to the pressure.
3. Calculate an approximation to the Darcy velocity vector using the approximations to the fluid density and pressure in (3.1).
4. Calculate an approximation to the entries in the dispersion tensor using the Darcy velocity vector.
5. Using the approximations to the fluid density, pressure, Darcy velocity and dispersion tensor, solve the contaminant mass balance equation (3.4) for an approximation to the contaminant concentration. Because approximations are available for ρ, $\boldsymbol{q}$ and $\underline{\boldsymbol{D}}$, this equation is now linear in c.

6. Calculate a new approximation to the fluid density from the approximation to the contaminant concentration using the constitutive equation (3.6).

7. Compare the new approximation to the fluid density with the previous one to see if a valid solution state has been achieved. If such a state has not been achieved (i.e. the approximation to the fluid density has changed during Stages 2 to 6), repeat from Stage 2.

Practical experience has shown that this coupling iteration approach converges, the maximum pointwise relative difference in the density over a coupling iteration decreasing monotonically. The coupling iteration is examined in more detail in Chapter 7.

3.3 Discretisation Techniques

In general, due to irregular geometry of the domain, spatial variability of physical properties such as porosity and conductivity, non-uniformity of initial conditions, and non-analytic forms of source and sink terms, analytic solutions of the governing equations are only possible for very simple problems. Also, due to lack of information, complete data for physical properties is not available so it must be approximated - hence an exact solution is a pointless luxury. Solutions of most problems can only be obtained in approximate form, and then only by numerical methods.

In general, these numerical methods consist of discretisation procedures which replace linear partial differential equations by linear systems of algebraic equations. The final solution is obtained by solving this system of equations.

In this section, discretisation methods are described. These are applied to the governing equations in the next section.

3.3.1 Spatial Discretisation

The first step in spatial discretisation is to construct a mesh on the region of interest. The finite number of nodes in this mesh are the positions at which discrete approximations to the solution variable are calculated.

The Finite Difference Method

The finite difference method [68, 75] is probably the oldest spatial discretisation technique. The meshes used with this method are usually regular (i.e. rows and columns of nodes aligned with the axis system).

In this method, the partial derivatives which appear in the differential equation are replaced by an algebraic approximation, with a quotient of two finite differences of the dependent and an independent variable replacing the differential quotient. The basic idea is that the derivative $\frac{du}{dx}$ of a function $u(x)$ evaluated at x_1 is defined as

$$\left.\frac{du}{dx}\right|_{x_1} = \lim_{\Delta x \to 0} \frac{u(x_1 + \Delta x) - u(x_1)}{\Delta x},$$

so an approximation to the derivative is obtained by omitting the limiting process, i.e.

$$\left.\frac{du}{dx}\right|_{x_1} \approx \frac{u(x_1 + \Delta x) - u(x_1)}{\Delta x} \tag{3.7}$$

In the finite difference method, approximations such as (3.7) are applied to the differential equation at each grid point, with Δx being the mesh size in the x-direction. This results in an equation for each node, involving the approximation to the solution variable at that node and the approximations to the solution variables at some (or all) of the neighbouring nodes. Hence all the equations are linked together (i.e. a matrix system is obtained for the unknowns at the nodes). The influence of one node is limited to those nodes connected to it directly via the mesh so the overall matrix system is sparse. The nodes at which values are used in the approximation of the derivatives at a point constitute the *local stencil* of the discretisation scheme.

The error in the approximation (3.7) is termed the *truncation error*, τ. An expression for this is calculated by performing a Taylor series expansion on the $u(x_1 + \Delta x)$ term about x_1. For example, assuming that the function u is sufficiently differentiable, the truncation error in approximation (3.7) is

$$\begin{aligned}
\tau &= \frac{u(x_1 + \Delta x) - u(x_1)}{\Delta x} - \left.\frac{du}{dx}\right|_{x_1} \\
&= \frac{1}{\Delta x}\left(u(x_1) + \Delta x \left.\frac{du}{dx}\right|_{x_1} + \frac{(\Delta x)^2}{2!}\left.\frac{d^2u}{d^2x}\right|_{(x_1+\theta\Delta x)} - u(x_1)\right) - \left.\frac{du}{dx}\right|_{x_1}
\end{aligned}$$

$$\begin{aligned} &= \frac{\Delta x}{2} \left.\frac{d^2u}{d^2x}\right|_{(x_1+\theta\Delta x)} \qquad (0<\theta<1) \\ &= \mathrm{O}(\Delta x). \end{aligned}$$

Hence, the truncation error tends towards zero as the mesh size is decreased - an approximation possessing this property is called *consistent*.

Approximation (3.7) is known as a *forward difference*. Other finite difference approximations are the *backward difference*,

$$\left.\frac{du}{dx}\right|_{x_1} \approx \frac{u(x_1) - u(x_1 - \Delta x)}{\Delta x},$$

which also has $\tau = \mathrm{O}(\Delta x)$; and the *central difference*,

$$\left.\frac{du}{dx}\right|_{x_1} \approx \frac{u(x_1 + \Delta x) - u(x_1 - \Delta x)}{2\Delta x},$$

which has $\tau = \mathrm{O}\left((\Delta x)^2\right)$. As can be seen from the respective truncation errors, as the mesh size is decreased the truncation error in the central difference approximation tends to zero faster than that in either the forward or backward differences.

Definition 3.1 *A discrete approximation to a derivative has an* order of accuracy *of p if the leading terms in the truncation error are of order $(\Delta x)^p$.*

Due to the $(\Delta x)^2$ leading term in the truncation error, the central discretisation is second order accurate, while the forward and backward discretisation are both only first order accurate.

Finite difference approximations also exist for higher derivatives. For example, a central difference approximation to the second derivative of the function $u(x)$ evaluated at x_1 is,

$$\left.\frac{d^2u}{d^2x}\right|_{x_1} \approx \frac{u(x_1 + \Delta x) - 2u(x_1) + u(x_1 - \Delta x)}{(\Delta x)^2},$$

which has $\tau = \mathrm{O}\left((\Delta x)^2\right)$ provided the function is sufficiently differentiable.

In order to apply the finite difference method to partial differential equations, the first stage is to define a grid on the domain of interest. Then a discrete approximation to the partial differential equation is generated by replacing each term in the equation by its discrete approximation on the (previously defined) grid.

Theoretically, the finite difference method only gives discrete solution values, i.e. the solution is not defined between nodes; a solution covering the whole domain may be recovered by interpolating between the nodal values.

The Finite Element Method

In the numerical solution of differential equations arising from the modelling of hydrological systems, the finite element method [19, 78] is a very popular spatial discretisation technique.

In the finite element method, the domain is divided into elements, typically triangles and/or quadrilaterals in 2-D, or tetrahedra and/or "bricks" in 3-D.

The solution variable is approximated by a finite dimensional expansion,

$$u(x) \approx \sum_{J}^{nodes} U_J N_J(x), \tag{3.8}$$

where U_J is the approximate solution at node J

$N_J(x)$ is the basis function associated with node J.

The basis function is defined so that

$$N_J(x) = \begin{cases} 1 & \text{at node } J \\ 0 & \text{outside the support of node } J, \end{cases}$$

where the support of a node is the part of the domain encompassed by the elements that possess that node. N_J is usually chosen to be a simple low order polynomial (e.g. linear, bi-linear, quadratic). An example basis function is shown in Figure 3.1.

If linear basis functions are used in (3.8), the approximate solution is piecewise linear. In general, the differential equation to be discretised has a solution which is more differentiable than the approximate solution. If piecewise linear basis functions are used, the approximate solution has an undefined second derivative and the first derivative is only defined uniquely inside each element.

To overcome this problem, the differential equation is replaced by a *weak form* which admits solutions that are less differentiable than the classical solution of

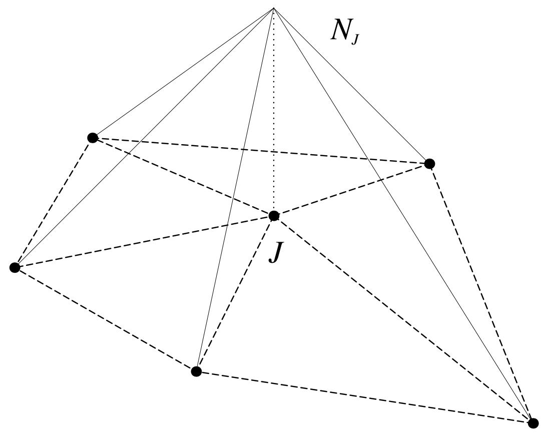


Figure 3.1: A linear basis function at node J in a 2-D mesh of triangles

the differential equation. A weak form is obtained by multiplying the differential equation by a test function, ω, and integrating over the whole spatial domain, i.e. if the differential equation is

$$\mathcal{L}u = f \qquad \text{on } \Omega$$

where $\mathcal{L}$ is a differential operator in space on u, then a weak form is

$$\int_\Omega \omega(\mathcal{L}u - f)d\Omega = 0. \tag{3.9}$$

If a function satisfies this weak form for *all* ω, then the function satisfies the differential equation at all points of the domain ([96] p210), i.e. it is the classical solution.

A solution satisfying (3.9) must be as differentiable as the classical solution. However, it is often possible to use Green's first identity,

$$\int_\Omega (v\boldsymbol{\nabla}.\boldsymbol{\nabla}w + \boldsymbol{\nabla}v.\boldsymbol{\nabla}w)d\Omega = \int_\Gamma v\boldsymbol{\nabla}w.\boldsymbol{n}d\Gamma \tag{3.10}$$

(where v and w are scalar functions and $\boldsymbol{n}$ is the unit outward normal vector at the surface Γ) to transfer part of the differential operator onto the test function - this generates another weak form, solutions of which can be less differential than the classical solution. For example, for the Poisson equation, (3.9) is

$$\int_\Omega \omega(\boldsymbol{\nabla}.\boldsymbol{\nabla}u - f)d\Omega = 0 \tag{3.11}$$

and Green's first identity applied to the differential operator gives

$$\int_\Gamma \omega\boldsymbol{\nabla}u.\boldsymbol{n}d\Gamma - \int_\Omega (\boldsymbol{\nabla}\omega.\boldsymbol{\nabla}u + \omega f)d\Omega = 0. \tag{3.12}$$

While (3.11) requires u to have an integrable second derivative, the equivalent equation (3.12) only requires u (and ω) to have an integrable first derivative.

The weak form is approximated discretely by requiring that it is only satisfied for a finite set of n test functions,

$$\omega = \omega_I \qquad I = 1, 2, \ldots, n$$

so that the approximation to the weak form (3.9) is

$$\int_\Omega \omega_I(\mathcal{L}u - f)d\Omega = 0 \qquad I = 1, 2, \ldots, n. \tag{3.13}$$

Since the approximation to u given by (3.8) involves a number of unknowns equal to the number of nodes in the mesh (these unknowns being the nodal values), then the same number of independent equations is required to define these unknowns uniquely.

Common choices for the test functions are:

- Dirac delta functions i.e.

  $$\omega_I = \delta_I \qquad I = 1, 2, \ldots, nodes$$

  where $\delta_I = 0$ everywhere apart from at node I and $\int_\Omega \delta_I d\Omega = 1$. This choice results in the *collocation* method.

- The basis functions already used to generate the finite dimensional approximation to the solution i.e.

  $$\omega_I = N_I(x) \qquad I = 1, 2, \ldots, nodes.$$

  This is known as the *Bubnov-Galerkin* method, commonly abbreviated just to the Galerkin method.

- A generalisation of the Galerkin method in which the test function is not necessarily from the same space as the basis function (e.g. the basis function could be linear while the test function is constant), i.e.

  $$\omega_I = M_I(x) \qquad I = 1, 2, \ldots, nodes$$

  where

  $$M_I(x) = \begin{cases} 1 & \text{at node } I \\ 0 & \text{outside the support of node } I \end{cases}$$

  This is known as the *Petrov-Galerkin* method.

In practice, the basic steps in a finite element discretisation are:

- Generation of weak form:
  - Multiply the differential equation by a test function and integrate over the whole spatial domain.
  - Bearing in mind that the solution will be replaced by an approximation which has limited continuity of derivatives, use Green's first identity to replace derivatives of the solution in the weak form by lower derivatives plus a boundary integral term.
- Discretisation steps:
  - Generate a mesh comprising elements covering the whole domain.
  - Approximate the solution variable by a finite dimensional expansion.
  - Approximate the general test function by a set of nodal test functions.
  - Split the integral in the weak form into the sum of integrals over elements which can be evaluated separately.
- Perform element integrals and assemble contributions to nodal equations from all elements in the support of each node. The influence of one node is limited to those elements in its support - hence, as in the finite difference method, the matrix systems which arise are sparse.

Compared with the finite difference method, the finite element method has the advantages that it allows flexible representation of features, it copes naturally with Neumann and Dirichlet boundary conditions, physical properties can be defined on each element and the solution exists everywhere. Hence it is the preferred spatial discretisation method for hydrological problems, and is the spatial discretisation method used in this thesis.

3.3.2 Temporal Discretisation

Since the finite element method is used to discretise in space, there are two obvious approaches for the time variable :

- Treat time as just another dimension and use test and basis functions in both space and time. The problem is usually an initial value problem in time (rather than a boundary value problem) for which test and basis functions are difficult to define. This approach leads to very large systems for 2-D (and more-so 3-D) time dependent problems.

- Treat the nodal variables as functions of time, and only use the space variables in the finite element analysis, i.e. (3.8) becomes

$$u(x,t) \approx \sum_{J}^{nodes} U_J(t) N_J(x). \tag{3.14}$$

  This leads to a system of ordinary differential equations in time which can be solved by a finite difference or finite element approach.

The latter approach is the one used here, with finite difference techniques used to discretise in time.

There are two main distinctions to be made between different finite difference techniques for discretisation in time. The first is whether the discretisation is *explicit* or *implicit*. This denotes the discrete time-level at which the spatial derivatives are approximated. The second distinction is whether the discretisation is based on the *Eulerian* or *Lagrangian* methodology. These distinctions are discussed separately in the following two sections.

Explicit versus Implicit - Convergence and Stability

In an explicit temporal discretisation, each nodal value at the new (i.e. unknown) time-level is given explicitly in terms of known nodal values (i.e. those at previous time-levels and boundary conditions); while in an implicit temporal discretisation, each nodal value at the new time-level depends implicitly on other (unknown) nodal values at the new time-level, as well as on known values from previous time-levels and boundary conditions (hence a matrix system must be solved to determine these implicit values). To illustrate this, consider the constant coefficient linear advection equation,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0. \tag{3.15}$$

A basic two-level temporal discretisation (with a three-point central finite difference spatial discretisation) for (3.15) on a regular (i.e. equally spaced) mesh, is

$$\frac{U_J^{n+1}-U_J^n}{\Delta t}+(1-\theta)a\left(\frac{U_{J+1}^n-U_{J-1}^n}{2\Delta x}\right)+\theta a\left(\frac{U_{J+1}^{n+1}-U_{J-1}^{n+1}}{2\Delta x}\right)=0 \qquad (0\leq\theta\leq 1) \tag{3.16}$$

where U_J^n is the approximation to the true solution u at time-level n and grid point J, Δt is the size of the time-step and Δx is the spatial mesh size. This approximation is fully explicit if $\theta = 0$ and fully implicit if $\theta = 1$. The degree of implicitness (i.e. the choice of θ) governs the properties of the discrete solution. A Taylor series expansion in both space and time shows that the discretisation (3.16) is second order accurate in time for $\theta = 1/2$ and first order for all other choices of θ.

Two major aspects of the quality of a numerical solution are *convergence* and *stability*. A numerical solution is said to be convergent if it tends to the true solution along a fixed time-level as Δx and Δt both tend to zero. A numerical solution is linearly stable if it is bounded, at a fixed time-level T, as $\Delta t \rightarrow 0$ (assuming that Δt and Δx are related so that $\Delta x \rightarrow 0$ as $\Delta t \rightarrow 0$).

Fully explicit methods usually require a time-step restriction to ensure a convergent approximation, while this is often achieved without such a restriction with implicit methods. In order to explain this, the origin of the numerical solution must be considered in conjunction with the idea of the *characteristics* of the true solution.

The *total derivative* of a variable $u(x(t),t)$ with respect to t is

$$\frac{Du}{Dt}=\frac{\partial u}{\partial t}+\frac{dx}{dt}.\frac{\partial u}{\partial x}. \tag{3.17}$$

Again using the scalar wave equation for illustration, comparing (3.15) and (3.17), it is easily seen that the solution is unchanged, i.e.

$$\frac{Du}{Dt}=0,$$

along the paths, $x(t)$ given by the ordinary differential equation,

$$\frac{dx}{dt}=a.$$

These paths are known as characteristics. They are the paths along which the true solution data is transferred. In the numerical scheme, if the characteristic through node J at time-level $(n+1)$ comes from outside the local stencil or support of node J at time-level n, then the transfer of information through time along the characteristics which drives the true solution cannot be modelled numerically by an explicit scheme. Hence an explicit scheme can only be convergent if the spatial and temporal mesh sizes are restricted so that the characteristic stays within the local stencil. Hence, for the approximate solution given by the fully explicit discretisation ((3.16) with $\theta = 0$) to be convergent to the solution of (3.15),

$$|a|\frac{\Delta t}{\Delta x} \leq 1.$$

The requirement that the characteristic stays within the local stencil of the scheme is known as the Courant-Friedrichs-Lewy (commonly abbreviated to CFL) condition.

Apart from the time step restriction required to ensure a convergent approximation, fully explicit methods usually also require a time-step restriction to ensure stability, while fully implicit methods often have unconditional stability (giving a stable solution with arbitrarily large time-steps). A rigorous approach to stability is provided by Fourier analysis, this involves putting a general Fourier mode into the scheme and looking for conditions such that this mode cannot grow over a time-step. For the discretisation scheme (3.16), Fourier analysis shows that the stability properties are

$$\theta \quad \begin{cases} < 1/2 & \text{stable if } |a|\dfrac{\Delta t}{\Delta x} \leq 1 \\ \geq 1/2 & \text{unconditionally stable} \end{cases}$$

Since it combines unconditional stability with second-order accuracy, the choice $\theta = 1/2$ is popular; the resulting temporal discretisation is known as the *Crank-Nicolson* method.

The quality of the numerical solution is not the only consideration in the temporal discretisation - the speed of the method is another important factor. Explicit methods tend to be much cheaper per time-step than implicit methods since no matrix inversion is required. In practice, the relative speed of the two approaches (i.e. explicit or implicit) also depends on the size of the largest acceptable time-step needed for other solution quality considerations such as accuracy and monotonicity.

Eulerian versus Lagrangian

Eulerian methods are based on the description of fluid flow which monitors the fluid behaviour at a fixed point by observing different fluid particles passing through that point. The focus of this approach is the fluid properties at the fixed point - not the fluid particles themselves. The stability discussion for explicit and implicit schemes earlier in this section is tacitly Eulerian (being focused on node J). In that description, the equation for a fluid property is discretised by approximating the equation at particular points (these points being the nodes).

In the Lagrangian approach, the focus is on "particles" of the fluid. The fluid is described by following these particles as they flow through the domain. A property of the fluid at a point is determined by the history of the particle currently at that point.

Mathematically, the Lagrangian description involves recasting the inertial and advective terms as a combined, or total, derivative which holds along characteristics, i.e.

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \boldsymbol{a}.\boldsymbol{\nabla} \qquad \text{on} \quad \frac{d\boldsymbol{x}}{dt} = \boldsymbol{a}.$$

Hence, in a Lagrangian framework, the advection-diffusion equation,

$$\frac{\partial u}{\partial t} + \boldsymbol{a}.\boldsymbol{\nabla} u = b\boldsymbol{\nabla}.\boldsymbol{\nabla} u \qquad (b \geq 0)$$

becomes

$$\frac{Du}{Dt} = b\boldsymbol{\nabla}.\boldsymbol{\nabla} u \qquad \text{on} \quad \frac{d\boldsymbol{x}}{dt} = \boldsymbol{a}.$$

So a temporally discretised equation (obtained by the backward Euler method) is

$$u(\boldsymbol{y}, t + \Delta t) = u(\boldsymbol{x}(t), t) + \Delta t \left\{ b\boldsymbol{\nabla}.\boldsymbol{\nabla} u(\boldsymbol{y}, t + \Delta t) \right\},$$

where the position vector $\boldsymbol{y}$ is the approximate solution of the Cauchy problem for the characteristic (obtained again by the backward Euler method)

$$\boldsymbol{y} = \boldsymbol{x}(t) + \boldsymbol{a}(t + \Delta t)\Delta t.$$

Even explicit Lagrangian methods can be unconditionally stable since the solution is followed along characteristics.

The simplest way to implement a Lagrangian method numerically is to specify initially a discrete set of "particles" in the fluid, then trace the positions of these particles in time by solving the equations for the characteristic, and modify the properties of the fluid "particles" by spatially approximating the total derivative equation. This approach is known as *particle tracking.* In its simplest form, it has some problems :

- The solution only exists at discrete points, i.e. the "particles"; however, as with the finite difference spatial discretisation, a solution over the whole domain can be recovered by interpolation.

- The chosen set of points may represent the fluid adequately throughout the region at the start of the process, but once they are allowed to move they may not necessarily do so. This can make the spatial derivatives (e.g. diffusion terms) difficult to approximate - particularly in regions where there are few particles.

In the worst scenario, all the "particles" leave the region and there are no discrete solution values - this can only be avoided if the set of "particles" is redefined after each time-step. One method for doing this is to have a prescribed grid, perform a Lagrangian time-step on the nodes, and interpolate the solution back onto the nodes in the prescribed grid.

This fixed grid method is called *semi-Lagrangian* if the underlying spatial discretisation is finite difference with an interpolation operator, and *Lagrange-Galerkin* if it is finite element with a projection operator.

3.3.3 Closing Comments

There are other discretisation approaches e.g. finite volume [41, 42], boundary element [7], spectral [79]; these are not described or used here. In general, the Galerkin finite element method is used to spatially discretise the governing equations in this thesis. Implicit Eulerian methods are used to perform the temporal discretisation where appropriate.

3.4 Discretisation of the Governing Equations

Because of the coupling iteration approach outlined on Section 3.2, each of the governing equations in the system can be considered in isolation for the purposes of the discretisation.

3.4.1 Discretisation of the Fluid Continuity Equation

Step 2 of Algorithm 3.1 requires the solution of the fluid continuity equation. For convenience, the first steps of the Galerkin finite element spatial discretisation of the fluid continuity equation (3.3) are performed on the fluid mass balance equation (3.2). A weak form is obtained by multiplying by a test function, ω, and integrating over the whole spatial domain, Ω, i.e.

$$\int_\Omega \left(\phi \frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} . \rho \boldsymbol{q} \right) \omega d\Omega = 0.$$

Applying Green's first identity (3.10) to the advective term gives,

$$\int_\Omega \left(\omega \phi \frac{\partial \rho}{\partial t} - \boldsymbol{\nabla} \omega . \rho \boldsymbol{q} \right) d\Omega + \int_\Gamma \omega \rho \boldsymbol{q} . \boldsymbol{n} d\Gamma = 0$$

and then imposing the prescribed fluid flux boundary condition gives,

$$\int_\Omega \left(\omega \phi \frac{\partial \rho}{\partial t} - \boldsymbol{\nabla} \omega . \rho \boldsymbol{q} \right) d\Omega = \int_\Gamma \omega \rho q_n d\Gamma. \tag{3.18}$$

In order to generate a weak form of the fluid continuity equation, substitute for $\boldsymbol{q}$ from (3.1),

$$\int_\Omega \left\{ \omega \phi \frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \omega . \underline{\boldsymbol{K}} \left(\frac{\boldsymbol{\nabla} p}{g} + \rho \boldsymbol{\nabla} z \right) \right\} d\Omega = \int_\Gamma \omega \rho q_n d\Gamma.$$

This weak form is spatially discretised by the Galerkin finite element method, i.e.

$$\omega = N_I \qquad I = 1, \ldots, nodes,$$

and

$$p \approx \sum_{J=1}^{nodes} p_J N_J$$

where the p_J are the nodal approximations to p, to give the system

$$\frac{1}{g} \sum_{J=1}^{nodes} p_J \sum_e \int_{\Omega^e} \boldsymbol{\nabla} N_I . \underline{\boldsymbol{K}} \boldsymbol{\nabla} N_J d\Omega^e$$
$$= \sum_e \int_{\Gamma^e} N_I \rho q_n d\Gamma^e - \sum_e \int_{\Omega^e} \left(N_I \phi \frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} N_I . \underline{\boldsymbol{K}} \rho \boldsymbol{\nabla} z \right) d\Omega^e$$

(for $I = 1, \ldots, nodes$) where $\sum_e$ denotes summation over all the elements in the support of node I and e superscripts denote element domains or boundaries.

The $\frac{\partial \rho}{\partial t}$ term is discretised temporally by the backward Euler method, i.e.

$$\left.\frac{\partial \rho}{\partial t}\right|_{n+1} = \frac{\rho^{n+1} - \rho^n}{\Delta t}$$

where the superscripts denote the discrete time-level. This discretisation is only first order accurate. It is deemed to be sufficient for this term since the density (i) has already been approximated as part of the coupling iteration and (ii) is assumed not to vary quickly. This gives the linear system

$$K^{n+1}\boldsymbol{p}^{n+1} = g\boldsymbol{F}^{n+1} - \frac{g}{\Delta t}\left(\boldsymbol{G}^{n+1} - \boldsymbol{G}^n\right) \tag{3.19}$$

where

$$K^i = \{K^i_{IJ}\}_{I,J=1,\ldots,nodes} \qquad \boldsymbol{F}^i = \{F^i_I\}_{I=1,\ldots,nodes}$$
$$\boldsymbol{p}^i = \{p^i_J\}_{J=1,\ldots,nodes} \qquad \boldsymbol{G}^i = \{G^i_I\}_{I=1,\ldots,nodes}$$

with the matrix and vector entries given by

$$\begin{aligned}
K^i_{IJ} &= \sum_e K^{e,i}_{IJ} = \sum_e \int_{\Omega^e} \boldsymbol{\nabla} N_I . \underline{\boldsymbol{K}}^i \boldsymbol{\nabla} N_J d\Omega^e \\
F^i_I &= \sum_e F^{e,i}_I = \sum_e \left(\int_{\Gamma^e} N_I \rho^i q^{h,i}_n d\Gamma^e - \int_{\Omega^e} \boldsymbol{\nabla} N_I . \underline{\boldsymbol{K}}^i \rho^i \boldsymbol{\nabla} z d\Omega^e \right) \\
G^i_I &= \sum_e G^{e,i}_I = \sum_e \int_{\Omega^e} N_I \phi \rho^i d\Omega^e .
\end{aligned}$$

In practice, the element integrals are simplified by approximating known material and fluid properties by their average values on elements (or element faces depending on the region of integration) denoted by $\langle \cdot \rangle$. Hence the matrix and vector entries in (3.19) are given by

$$\begin{aligned}
K^i_{IJ} &= \sum_e K^{e,i}_{IJ} = \sum_e \int_{\Omega^e} \boldsymbol{\nabla} N_I . \langle \underline{\boldsymbol{K}}^i \rangle \boldsymbol{\nabla} N_J d\Omega^e \\
F^i_I &= \sum_e F^{e,i}_I = \sum_e \left(\langle \rho^i \rangle \langle q^i_n \rangle \int_{\Gamma^e} N_I d\Gamma^e - \langle \underline{\boldsymbol{K}}^i \rangle \langle \rho^i \rangle \boldsymbol{\nabla} z . \int_{\Omega^e} \boldsymbol{\nabla} N_I d\Omega^e \right) \\
G^i_I &= \sum_e G^{e,i}_I = \sum_e \langle \rho^i \rangle \langle \phi \rangle \int_{\Omega^e} N_I d\Omega^e .
\end{aligned}$$

After the Dirichlet boundary conditions are applied, the matrix in this system is SPD if the tensor $\langle \underline{\boldsymbol{K}}^{n+1} \rangle$ is SPD ([19], p212). For a reasonably fine spatial discretisation, the system is also large and sparse.

3.4.2 Discrete Darcy Velocity Vector Calculation

Once the discrete fluid continuity equation from the previous section has been solved, it is possible to evaluate the Darcy velocity by taking derivatives of the calculated pressure field. However, as shown in [93], this approach leads to a discontinuity in the velocity at nodal points and a violation of the conservation of mass in a local sense. This can be avoided by evaluating the Darcy velocities using a Galerkin finite element spatial discretisation of (3.1),

$$\sum_{J=1}^{nodes} \boldsymbol{q}_J \sum_e \int_{\Omega^e} N_I N_J d\Omega^e$$
$$= -\frac{1}{g} \sum_{J=1}^{nodes} p_J \sum_e \int_{\Omega^e} N_I \frac{1}{\rho} \underline{\boldsymbol{K}} . \boldsymbol{\nabla} N_J d\Omega^e - \sum_e \int_{\Omega^e} N_I \underline{\boldsymbol{K}} . \boldsymbol{\nabla} z d\Omega^e$$

where

$$\boldsymbol{q} \approx \sum_{J=1}^{nodes} \boldsymbol{q}_J N_J \quad \text{and} \quad p \approx \sum_{J=1}^{nodes} p_J N_J .$$

Evaluating this at the $(n+1)^{\text{th}}$ discrete time-level (temporal discretisation is not required for this equation) gives the set of linear equations

$$M \boldsymbol{q}^{n+1} = -\frac{1}{g} \boldsymbol{V}^{n+1} \boldsymbol{p}^{n+1} - \boldsymbol{F}^{n+1} \tag{3.20}$$

where

$$M = \{M_{IJ}\}_{I,J=1,\ldots,nodes} \qquad \boldsymbol{q}^i = \{\boldsymbol{q}^i_J\}_{J=1,\ldots,nodes}$$
$$\boldsymbol{V}^i = \{\boldsymbol{V}^i_{IJ}\}_{I,J=1,\ldots,nodes} \qquad \boldsymbol{p}^i = \{p^i_J\}_{J=1,\ldots,nodes}$$
$$\boldsymbol{F}^i = \{\boldsymbol{F}^i_I\}_{I=1,\ldots,nodes}$$

with

$$\begin{aligned} M_{IJ} &= \sum_e M^e_{IJ} = \sum_e \int_{\Omega^e} N_I N_J d\Omega^e \\ \boldsymbol{V}_{IJ} &= \sum_e \boldsymbol{V}^{e,i}_{IJ} = \sum_e \frac{\langle \underline{\boldsymbol{K}}^i \rangle}{\langle \rho^i \rangle} \int_{\Omega^e} N_I \boldsymbol{\nabla} N_J d\Omega^e \\ \boldsymbol{F}^i_I &= \sum_e \boldsymbol{F}^{e,i}_I = \sum_e \langle \underline{\boldsymbol{K}}^i \rangle \boldsymbol{\nabla} z \int_{\Omega^e} N_I d\Omega^e \ . \end{aligned}$$

The angled brackets in have the same meaning as those in the discrete fluid continuity equation (3.19).

There is no need to impose the Dirichlet flow boundary conditions in this system as these are already incorporated in the pressure solution that is obtained from the fluid continuity equation (where they are imposed as Neumann conditions).

The matrix in this system is the finite element mass matrix. It is SPD.

3.4.3 Discretisation of the Contaminant Mass Balance Equation

Substituting (3.2) into (3.4) leads to the following form of the contaminant mass balance equation,

$$\rho\phi\frac{\partial c}{\partial t} + \rho\boldsymbol{q}.\boldsymbol{\nabla} c = \boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\boldsymbol{\nabla}(\rho c).$$

Due to the constitutive relation (3.6),

$$\begin{aligned}\boldsymbol{\nabla}(\rho c) &= \boldsymbol{\nabla}\left(\rho_0 c + \epsilon c^2\right)\\ &= \left(\rho_0 + 2\epsilon c\right)\boldsymbol{\nabla} c\\ &= (\rho + \epsilon c)\boldsymbol{\nabla} c.\end{aligned}$$

For the system in this thesis where the contaminant is salt and the fluid is water, ϵc is small compared to ρ, (in fact $\epsilon c \leq 0.02499\rho$) so the approximation

$$\boldsymbol{\nabla}(\rho c) \approx \rho\boldsymbol{\nabla} c$$

is reasonable. With this approximation, the contaminant mass balance equation becomes

$$\rho\phi\frac{\partial c}{\partial t} + \rho\boldsymbol{q}.\boldsymbol{\nabla} c = \boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\rho\boldsymbol{\nabla} c. \tag{3.21}$$

This is an advection-diffusion equation, the $\rho\boldsymbol{q}.\boldsymbol{\nabla} c$ term is the advection part and the $\boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\rho\boldsymbol{\nabla} c$ term is the diffusion part. In order to apply the Galerkin finite element method to (3.21), it is replaced by the weak form

$$\int_\Omega \left(\omega\rho\phi\frac{\partial c}{\partial t} + \omega\rho\boldsymbol{q}.\boldsymbol{\nabla} c + \boldsymbol{\nabla}\omega.\phi\underline{\boldsymbol{D}}\rho\boldsymbol{\nabla} c\right) d\Omega = -\int_\Gamma \omega q_n^c d\Gamma$$

where Green's first identity is used to transfer part of the second order spatial differential operator to the test function.

This weak form is discretised spatially by using a discrete set of test functions,

$$\omega = N_I \qquad I = 1, \ldots, nodes,$$

and a finite dimensional expansion for the dependent variable,

$$c \approx \sum_{J=1}^{nodes} c_J N_J$$

to give a system of ordinary differential equations in time,

$$M\frac{d\boldsymbol{c}}{dt} + (V + D)\boldsymbol{c} = -\boldsymbol{F} \tag{3.22}$$

where

$$M = \{M_{IJ}\}_{I,J=1,\ldots,nodes} \qquad \boldsymbol{c} = \{c_J\}_{J=1,\ldots,nodes}$$
$$V = \{V_{IJ}\}_{I,J=1,\ldots,nodes} \qquad \boldsymbol{F} = \{F_I\}_{I=1,\ldots,nodes}$$
$$D = \{D_{IJ}\}_{I,J=1,\ldots,nodes}$$

with

$$\begin{aligned}
M_{IJ} &= \sum_e M^e_{IJ} = \sum_e \langle\rho\rangle\langle\phi\rangle \int_{\Omega^e} N_I N_J d\Omega^e \\
V_{IJ} &= \sum_e V^e_{IJ} = \sum_e \langle\rho\rangle\langle\boldsymbol{q}\rangle . \int_{\Omega^e} N_I \boldsymbol{\nabla} N_J d\Omega^e \\
D_{IJ} &= \sum_e D^e_{IJ} = \sum_e \langle\rho\rangle\langle\phi\rangle \int_{\Omega^e} \boldsymbol{\nabla} N_I . \langle\underline{\boldsymbol{D}}\rangle \boldsymbol{\nabla} N_J d\Omega^e \\
F_I &= \sum_e F^e_I = \sum_e \langle q^c_n\rangle \int_{\Gamma^e} N_I d\Gamma^e .
\end{aligned}$$

Again, the angled brackets have the same meaning as in the discretisation of the fluid continuity equation. An average value of the Darcy velocity of the fluid on each element is needed in this part of the numerical solution method. This is obtained by averaging the nodal approximations to the Darcy velocity vector on each element.

A suitable method for the temporal discretisation of this equation in an implicit Eulerian manner is the Crank-Nicolson method from Section 3.3.2. Applying this to (3.22) gives the fully discretised equation,

$$\begin{aligned}
&\left(\frac{1}{\Delta t}M^{n+1} + \frac{1}{2}V^{n+1} + \frac{1}{2}D^{n+1}\right)\boldsymbol{c}^{n+1} \\
&\quad = \left(\frac{1}{\Delta t}M^{n+1} - \frac{1}{2}V^{n} - \frac{1}{2}D^{n}\right)\boldsymbol{c}^{n} - \frac{1}{2}\left(\boldsymbol{F}^{n+1} + \boldsymbol{F}^{n}\right).
\end{aligned} \tag{3.23}$$

Again, the superscripts denote the discrete time-level.

The relative proportion of the physical processes is characterised by two dimensionless parameters: the Courant number, Co, and the Peclet number, Pe. The Courant number measures the proportion of advection to inertia in the problem, e.g. for a 1-D form of (3.21),

$$Co = \frac{vT}{X},$$

where X is a representative length and T is a representative elapsed time. The Peclet number measures the proportion of advection to diffusion in the problem, e.g. for the 1-D form of (3.21),

$$Pe = \frac{vX}{D}.$$

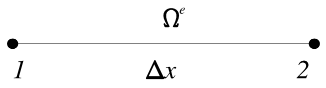
The physical Courant and Peclet numbers are not used in this work. Instead, Co and Pe are used to indicate the discrete versions of these numbers (e.g. Co indicates the proportion of the discrete advection term to the discrete inertia term). In these discrete versions, the representative length is the mesh size, Δx, and the representative elapsed time is the time-step, Δt.

From [56], the approach used to obtain (3.23), i.e. the Crank-Nicolson method in time combined with Galerkin finite element method in space, gives the same discrete equation as that obtained where the contaminant mass balance equation is discretised by an implicit Taylor-Galerkin method [20, 21]. The Taylor-Galerkin method uses (3.21) to replace the temporal derivatives in an approximate Taylor series expansion by spatial derivatives, and then performs a finite element discretisation in space.

The matrix in system (3.23) has three components, the mass matrix M^{n+1} which is SPD, the stiffness matrix D^{n+1} which is SPD (with the same assumptions on the structure of $\langle \underline{\boldsymbol{D}} \rangle^{n+1}$ as for $\langle \underline{\boldsymbol{K}} \rangle^{n+1}$ in Section 3.4.1) and the advection matrix V^{n+1} which is non-symmetric (and little can be said about its definiteness). Some examples of advection matrices for some simple elements are shown in Figure 3.2. The system (3.23) is therefore non-symmetric and, as with the discrete fluid continuity equation, it is large and sparse.

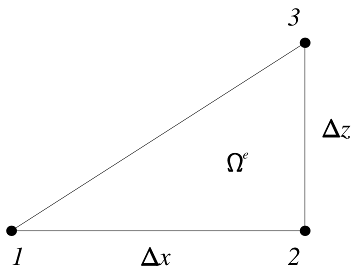
The origin of the non-symmetry is the advection term. This is the term which contains first spatial derivatives of the solution variable. A term of this form is not present in either the fluid continuity equation (when solved for p) or Darcy's law (when solved for $\boldsymbol{q}$).

Central discretisations - either finite difference or Galerkin finite element - of first derivatives lead to non-symmetry in the system matrix. The reason for this is most easily seen when a 1-D central finite difference framework is considered - relative to the centre of the local stencil, advection is an inherently non-symmetric process (as opposed to diffusion, for instance).



$$\int_{\Omega^e} N_I \frac{dN_J}{dx} d\Omega^e \Rightarrow \frac{1}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

(a) Linear 1-D



$$\int_{\Omega^e} N_I \frac{\partial N_J}{\partial x} d\Omega^e \Rightarrow \frac{\Delta z}{6} \begin{bmatrix} -1 & 1 & 0 \\ -1 & 1 & 0 \\ -1 & 1 & 0 \end{bmatrix}$$

$$\int_{\Omega^e} N_I \frac{\partial N_J}{\partial z} d\Omega^e \Rightarrow \frac{\Delta x}{6} \begin{bmatrix} 0 & -1 & 1 \\ 0 & -1 & 1 \\ 0 & -1 & 1 \end{bmatrix}$$

(b) Linear triangle

Figure 3.2: Advection Matrices for Some Simple Elements

The problems which arise in the solution of large, sparse non-symmetric linear systems are outlined in Section 2.3.2. Ideally, the matrix should be SPD rather than non-symmetric. Achieving this requires the elimination or reformulation of the V^{n+1} term in (3.23).

There are numerous techniques in the literature which are used to generate SPD matrices for advection-diffusion problems.

- Explicit treatment of advection - in this approach, the non-symmetric component appears in the RHS vector rather than the matrix. This approach leads to a restriction on the maximum allowable time-step for stability reasons.

- Lagrangian schemes - the advection part of the problem is dealt with in the solution of the trajectory, so there is no advection component in the matrix.

- Non-central schemes i.e. upwind or Petrov-Galerkin methods.

None of these approaches is used here - they are outside the scope of the discretisation approach chosen in Section 3.3.1 and 3.3.2.

Another approach for advection diffusion equations, which loosely fits the chosen discretisation approach is known as the *Leismann-Frind scheme* [50]. This is described in the following section.

Leismann-Frind Scheme

This scheme is based on an operator splitting in which the advection and the diffusion are treated differently. The method follows from work in [85, 86] where schemes for the discretisation of advection-diffusion equations with high orders of accuracy are generated by modifying the diffusion coefficient in order to eliminate terms in the truncation error. In the Leismann-Frind scheme, the terms in the dispersion tensor are modified to give *unconditional stability*.

The modification involves adding some *artificial diffusion*, $\underline{\boldsymbol{D}}^*$ so that the dispersion tensor (3.5) becomes

$$\hat{\underline{\boldsymbol{D}}} = \underline{\boldsymbol{D}} + \underline{\boldsymbol{D}}^* \quad \text{where} \quad \underline{\boldsymbol{D}}^* = \frac{\Delta t}{2}\boldsymbol{v}\boldsymbol{v}^T.$$

The advection term is treated fully explicitly ($\theta = 0$ - removing the source of the non-symmetry from the matrix) while the physical diffusion term ($\underline{\boldsymbol{D}}$) is treated fully implicitly ($\theta = 1$) and the artificial diffusion term ($\underline{\boldsymbol{D}}^*$) is treated in a Crank-Nicolson manner ($\theta = \frac{1}{2}$).

The resulting scheme possesses unconditional stability at the expense of accuracy - it is only first order in time. The amount of artificial diffusion depends on the size of the time-step so it doesn't affect the consistency of the approximation. In effect, this method converts the advection that needs to be treated implicitly (for stability) into an approximately equivalent amount of diffusion. This is possible due to the directionality of the dispersion tensor.

Applying the Leismann-Frind scheme to the contaminant mass balance equation gives the following fully discrete system,

$$\left(\frac{1}{\Delta t}M + D^{n+1} + \frac{1}{2}D^{*,n+1}\right)\boldsymbol{c}^{n+1} = \left(\frac{1}{\Delta t}M - V^n - \frac{1}{2}D^{*,n}\right)\boldsymbol{c}^n - \boldsymbol{F}$$

where

$$\begin{aligned} M &= \{M_{IJ}\}_{I,J=1,\ldots,nodes} & \boldsymbol{c}^i &= \{c^i_J\}_{J=1,\ldots,nodes} \\ V^i &= \{V^i_{IJ}\}_{I,J=1,\ldots,nodes} & \boldsymbol{F} &= \{F_I\}_{I=1,\ldots,nodes} \\ D^i &= \{D^i_{IJ}\}_{I,J=1,\ldots,nodes} & D^{*,i} &= \{D^{*,i}_{IJ}\}_{I,J=1,\ldots,nodes} \end{aligned}$$

with

$$\begin{aligned} M_{IJ} &= \sum_e M^e_{IJ} = \sum_e \int_{\Omega^e} N_I N_J d\Omega^e \\ V^i_{IJ} &= \sum_e V^{e,i}_{IJ} = \sum_e \frac{\langle \boldsymbol{q} \rangle^i}{\phi} . \int_{\Omega^e} N_I \boldsymbol{\nabla} N_J d\Omega^e \\ D^i_{IJ} &= \sum_e D^{e,i}_{IJ} = \sum_e \int_{\Omega^e} \boldsymbol{\nabla} N_I . \langle \underline{\boldsymbol{D}}^i \rangle \boldsymbol{\nabla} N_J d\Omega^e \\ D^{*,i}_{IJ} &= \sum_e D^{*,e,i}_{IJ} = \sum_e \int_{\Omega^e} \boldsymbol{\nabla} N_I . \langle \underline{\boldsymbol{D}}^{*,i} \rangle \boldsymbol{\nabla} N_J d\Omega^e \\ F_I &= \sum_e F^{e,i}_I = \sum_e \left(\frac{\langle q^{c,n+1}_n \rangle}{\langle \rho^{n+1} \rangle \langle \phi \rangle} + \frac{\langle q^{*,n+1}_n \rangle}{2\langle \rho^{n+1} \rangle \langle \phi \rangle} + \frac{\langle q^{*,n}_n \rangle}{2\langle \rho^{n} \rangle \langle \phi \rangle} \right) \int_{\Gamma^e} N_I d\Gamma^e \end{aligned}$$

Here, $q^*_n = -\phi \underline{\boldsymbol{D}}^* \boldsymbol{\nabla}(\rho c).\boldsymbol{n}$ on Γ'_2 in the style of the prescribed dispersive solute mass flux boundary condition described in Section 3.1.6.

Of the two components in the matrix of this system, the mass matrix (M) is SPD, while the positive definiteness of the stiffness matrix ($D^{n+1} + \frac{1}{2}D^{*,n+1}$) depends on whether the modified dispersion tensor is still positive definite. Now

$$\begin{aligned} \boldsymbol{x}^T \left(\underline{\boldsymbol{D}} + \frac{1}{2} \underline{\boldsymbol{D}}^* \right) \boldsymbol{x} &= \boldsymbol{x}^T (\underline{\boldsymbol{D}} + \frac{\Delta t}{4} \boldsymbol{v}\boldsymbol{v}^T) \boldsymbol{x} \\ &= \boldsymbol{x}^T \underline{\boldsymbol{D}} \boldsymbol{x} + \frac{\Delta t}{4} (\boldsymbol{v}^T \boldsymbol{x})(\boldsymbol{v}^T \boldsymbol{x}) \\ &> 0 \qquad 0 \neq \boldsymbol{x} \in \mathbb{R}^d \quad (\text{since } \underline{\boldsymbol{D}} \text{ is SPD and } \Delta t \geq 0) \end{aligned}$$

Hence the stiffness matrix is SPD (after boundary conditions have been imposed) so the matrix which results from the Leismann-Frind discretisation is SPD. This means it can be solved by the conjugate gradient method.

The Leismann-Frind scheme is a very elegant solution to the problem of combining symmetry and stable central discretisation of advection terms. However, the overall goal is to develop a solution approach which is capable of operating in a wide range of flow regimes. The Leismann-Frind scheme is first order in time with the coefficients of the leading error term dependent on the size of the physical diffusion - hence the scheme will not perform well in diffusion dominated problems. The Crank-Nicolson Galerkin finite element scheme is formally second order in time and, for this reason alone, it is the preferred discretisation approach for the contaminant mass balance equation.

The Crank-Nicolson Galerkin finite element scheme and the Leismann-Frind scheme are compared in [62]. In that work, the Leismann-Find scheme is found to give essentially the same results, but in less CPU time, for the particular case study used. However the flow regime (i.e. Courant and Peclet numbers) in the case study is not specified so it cannot be determined whether the results of the comparison are generally true.

3.5 Overview of Numerical Solution Approach

In summary, the governing equations in Section 3.1 must generally be solved approximately by numerical methods.

The overall solution approach used for the governing equations at each time-step in this work is given in Algorithm 3.2. This is an expanded version of Algorithm 3.1.

Algorithm 3.2 OVERALL APPROACH FOR THE GOVERNING EQUATIONS
This algorithm is an outline of the iterative approach used to couple and solve the governing equations.

1. Make an approximation for the fluid density at the nodes of the mesh by linear extrapolation from the nodal densities at the two previous time levels,

$$\rho_J^{n+1} = \rho_J^n + \frac{(\Delta t)^{n+1}}{(\Delta t)^n}(\rho_J^n - \rho_J^{n-1}) \qquad J = 1, \ldots, nodes.$$

2. Generate element densities from the nodal ones by simple averaging,

$$\langle \rho \rangle = \frac{1}{n_e} \sum_{I=1}^{n_e} \rho_I$$

   where n_e is the number of nodes in element e.

3. Using the approximation for the fluid density on elements, generate the discrete fluid continuity equation (3.19) and solve this for the nodal pressures.

The matrix in this system in SPD so the preconditioned conjugate gradient method is used to solve it.

4. Evaluate an approximation to the Darcy velocity vector at each node by generating the discrete Darcy law (3.20) using the approximations to the fluid density on elements and pressure at nodes. The matrix in this system is the SPD finite element mass matrix, hence this system is also solved by the preconditioned conjugate gradient method.

   Generate element Darcy velocity vectors by simple averaging in the same way as the element densities are evaluated.

5. Calculate an approximation to the entries in the dispersion tensor on elements using the Darcy velocity vector on elements.

6. Using the approximations to the fluid density, Darcy velocity and dispersion tensor on elements, generate the discrete contaminant mass balance equation (3.23) and solve for an approximation to the concentration by one of the non-symmetric iterative methods described in Section 2.3.2.

7. Calculate a new approximation to the fluid density on nodes from the approximation to the contaminant concentration using the constitutive equation (3.6).

8. Compare the new approximation to the fluid density with the previous one to see if the process has converged. If converged stop, else return to Stage 2.

This discretisation approach is used on two test problems in the next chapter with all the linear systems involved being solved exactly. The exact solution of the linear systems is impractically expensive and unnecessary since there are already errors in the solution process due to the approximations made in the discretisation. Exact solution is only carried out in the next chapter so that the performance of the discretisations can be examined in isolation (without any need to consider the effect of the approximate solution of the resultant linear systems). In practice, linear solvers are only used to generate an approximate solution of the system. The details of the linear solvers are investigated in Chapters 5 and 6.

Chapter 4

Discretisation Performance

The overall numerical solution approach is a combination of the discretisation applied to the governing equations and the solution of the resulting linear systems. In this chapter, the performance of the discretisation is examined in isolation; in order to facilitate this, all linear systems are solved exactly (to within the accuracy of the machine used). At this stage, no consideration is given to the expense this incurs.

Two tests cases are used to examine the performance of the discretisation - a 1-D tracer problem for which the analytic solution is known, and a 2-D problem which is a common test case in the literature. A tracer is a contaminant that does not affect the physical properties of the fluid so, in the 1-D tracer test case problem, only the contaminant mass balance equation needs to be solved so the discretisation of that equation can be examined in isolation.

A computer program was written in (double precision) FORTRAN 77 in order to carry out the numerical experiments. The two test cases serve partly to validate the program (e.g. solving a 1-D problem on 2-D meshes in order to confirm that the numerical results are 1-D) and partly to illustrate the behaviour of the discretisation and confirm that the overall approach solves the coupled system correctly.

4.1 1-D Passive Transport in a Column

This problem involves the 1-D transport of a tracer in a vertical column through which there is a constant fluid flow rate.

4.1.1 Specification of the 1-D Tracer Test Case

The transport is passive so ρ is constant and cancels from (3.1), (3.3) and (3.4); hence the density does not affect the problem so it can be given an arbitrary value.

This problem is 1-D but it is solved in a 2-D code. The physical data for this problem is :

$$K_{xx} = K_{xz} = K_{zx} = 0 \ , \ K_{zz} = 10^{-4}\text{m/s}$$

$$\phi = 0.2 \ , \ \alpha_L = 5\text{m} \ , \ \alpha_T = 0 \ , \ D_m = 0.$$

The physical region for the equivalent 2-D problem is shown in Figure 4.1. The notation is consistent with that in Section 3.1.6.

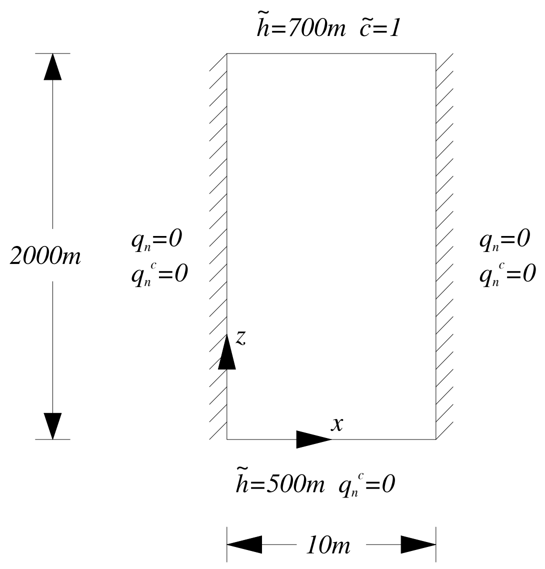


Figure 4.1: Physical domain and boundary conditions for 1-D problem

These conditions give rise to the constant flow field,

$$q_x = 0 \qquad , \qquad q_z = -10^{-5}\text{m/s}.$$

Only the contaminant mass balance equation (3.4) needs to be solved (for the dimensionless contaminant concentration c). In the tracer test case, (3.4) reduces to the 1-D equation,

$$\frac{\partial c}{\partial t} + v_z \frac{\partial c}{\partial z} - D_{zz} \frac{\partial^2 c}{\partial z^2} = 0 \qquad (D_{zz} > 0). \tag{4.1}$$

where v_z is the z-component of the fluid velocity vector This is discretised by the Crank-Nicolson Galerkin finite element method described in the previous chapter.

The Courant and Peclet numbers for this problem are

$$Co = \frac{|q_z|}{\phi} \frac{\Delta t}{\Delta z} = 5 \times 10^{-5} \frac{\Delta t}{\Delta z} \quad , \quad Pe = \frac{\Delta z}{\alpha_L} = \frac{\Delta z}{5}$$

where Δt is the time-step and Δz is the mesh size in the vertical direction.

Initially, the concentration of the tracer is zero everywhere in the region. The tracer front moves down the column under the action of gravity and disperses as it moves. Until the tracer reaches the bottom of the column, this problem can be treated as being on a semi-infinite domain for which the boundary conditions are

$$\begin{aligned} \tilde{c}(x, 2000, t) &= 1 \qquad & q_n^c(0, z, t) &= 0 \\ \tilde{c}(x, -\infty, t) &= 0 \qquad & q_n^c(10, z, t) &= 0 \ . \end{aligned}$$

The analytic solution to this semi-infinite domain problem is derived in [58] as

$$c(x, z, t) = \frac{1}{2} \left\{ \exp\left(\frac{v_z z'}{D_{zz}}\right) \operatorname{erfc}\left(\frac{z' - v_z t}{2\sqrt{D_{zz} t}}\right) + \operatorname{erfc}\left(\frac{z' + v_z t}{2\sqrt{D_{zz} t}}\right) \right\} \tag{4.2}$$

where $z' = 2000 - z$ and erfc is the complementary error function, given by

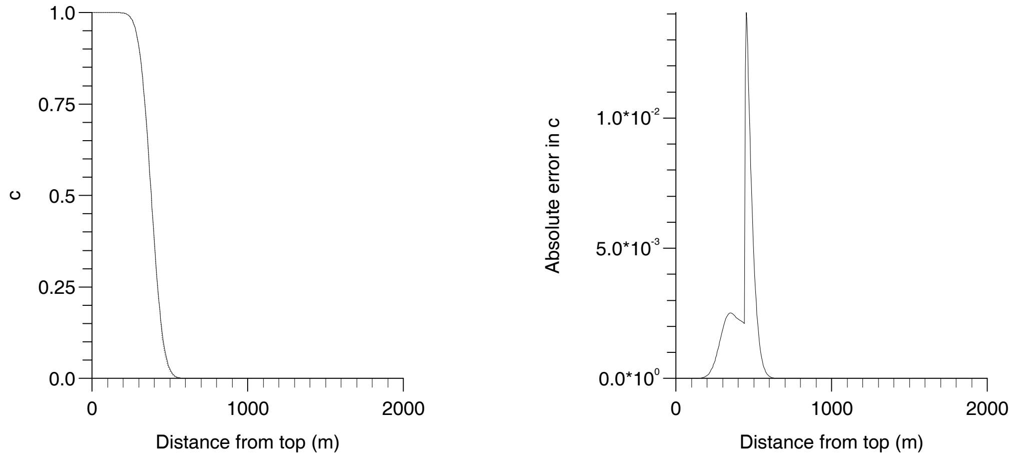
$$\operatorname{erfc}(s) = \frac{2}{\sqrt{\pi}} \int_s^\infty e^{-u^2} du.$$

In the results given in this work, the complementary error function is computed using the SUNOS C library intrinsic function `erfc` on a SPARCstation 1+.

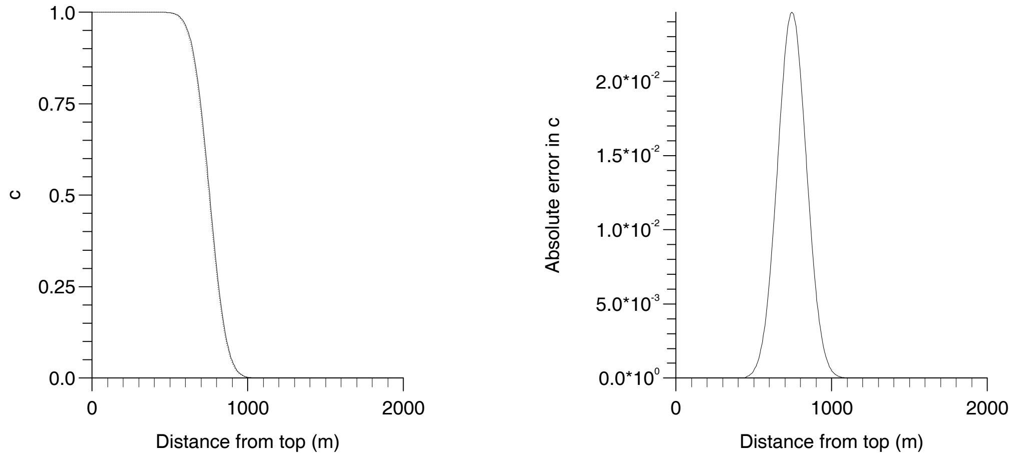
4.1.2 Results for 1-D Tracer Test Case

A uniform mesh of bilinear rectangular elements with 2 nodes in the x-direction (and a varying number of nodes in the z-direction) is used for this problem.

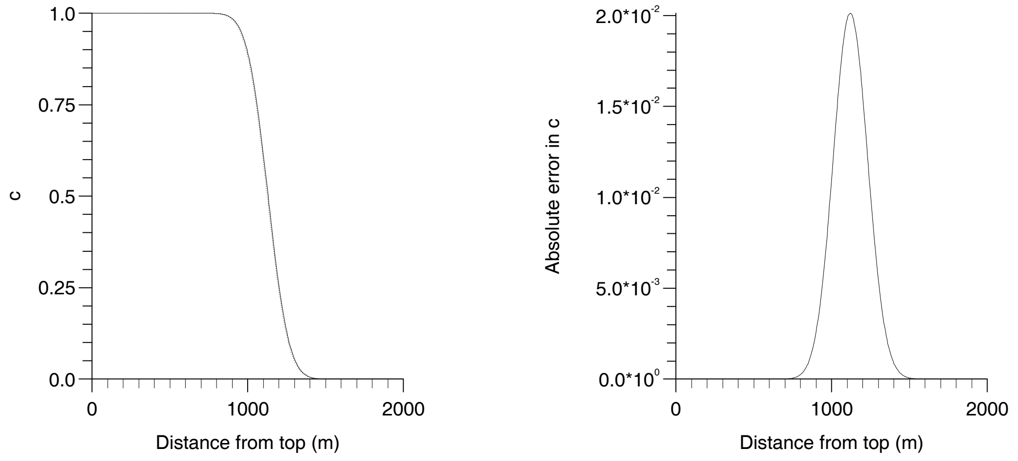
Although the experiments were carried out on a 2-D grid, the results are one-dimensional through symmetry. This is a good test case for the program - the



(a) $t = 7.5 \times 10^6$ s



(b) $t = 15 \times 10^6$ s



(c) $t = 22.5 \times 10^6$ s

Figure 4.2: Solution and error at $Co = 1$, $Pe = 1$

required symmetry is achieved in practice. For clarity, the results are presented in 1-D form.

Figure 4.2 shows both the approximate (dotted line) and analytic (solid line) solutions, and the error in the solution (defined as the approximate solution minus the analytic solution) for $\Delta z = 5$m and $\Delta t = 5 \times 10^4$ s after 150, 300 and 450 time-steps respectively. This corresponds to a Courant number of 0.5 and a Peclet number of 1.0. Due to the accuracy of the approximate solution, it is hard to distinguish it from the analytic one at each of the times shown. The graphs of the error show the difference more clearly.

Table 4.1 shows the behaviour of the approximate solution as the Courant number is increased with the same spatial mesh. The minimum and maximum values in the numerical solution during the time-stepping, c_{min} and c_{max} respectively, are shown in this table, as is the relative error in the approximate solution defined as,

$$\text{Relative Error} = \sqrt{\frac{\sum_J \{c_J - c(z_J)\}^2}{\sum_J \{c(z_J)\}^2}},$$

where the subscripts denote nodal values and c_J is a discrete approximation to $c(x_J)$.

Co	t ($\times 10^6$s)	c_{min}	$c_{max} - 1$	Relative Error
0.5	7.5	0	0	4.61×10^{-3}
	15	0	0	1.92×10^{-2}
	22.5	0	0	8.74×10^{-3}
1.0	7.5	0	0	4.59×10^{-3}
	15	0	0	1.18×10^{-2}
	22.5	0	0	8.62×10^{-3}
5.0	7.5	0	3.12×10^{-2}	1.90×10^{-2}
	15	0	8.86×10^{-3}	1.07×10^{-2}
	22.5	0	2.44×10^{-3}	7.67×10^{-3}

Table 4.1: Error and Extrema at $Pe = 1$

The relative error is approximately the same at the different Courant numbers (the spatial mesh is fixed while the time-step is coarsened to increase the Courant number).

The minimum and maximum of the analytic solution for the dimensionless contaminant concentration are 0 and 1. Hence the maxima that occurs in Table 4.1 at $Co = 5$ are unphysical - they are not features of the true solution and are caused by the numerical solution technique. The Crank-Nicolson Galerkin finite element scheme is not a monotonicity preserving scheme, i.e. it admits unphysical oscillations in the numerical solution. This is typical of a higher than first order scheme. The unphysical maximum appears to decay with time, suggesting that it is caused by conditions at the beginning of the simulation.

Co	t ($\times 10^6$s)	c_{min}	$c_{max} - 1$	Relative Error
0.5	7.5	0	0	2.22×10^{-2}
	15	0	0	2.23×10^{-2}
	22.5	0	0	1.64×10^{-2}
1.0	7.5	0	3.05×10^{-4}	2.07×10^{-2}
	15	0	1.43×10^{-6}	1.98×10^{-2}
	22.5	0	0	1.45×10^{-2}
5.0	7.5	0	2.85×10^{-1}	2.14×10^{-1}
	15	0	2.21×10^{-1}	1.41×10^{-1}
	22.5	0	1.93×10^{-1}	1.11×10^{-1}

Table 4.2: Error and Extrema at $Pe = 5$

Table 4.2 shows the same solution properties as Table 4.1 but at a higher Peclet number ($Pe = 5$), i.e. on a mesh which is five times coarser. Comparing these two tables, the relative errors are larger and the unphysical maxima are more pronounced at the higher Peclet number. Again, the unphysical maxima are decaying with time.

The unphysical extrema that occur in the numerical solution are an undesirable feature as they can interfere with any subsequent chemical or physical processes which are applied to the transported quantity. In the next section, the flow regimes where unphysical oscillations can occur are discussed and ways of removing or controlling them are described.

4.1.3 Unphysical Oscillations and their Control

The steady-state form of the 1-D advection-diffusion equation (4.1) is

$$v_z \frac{\partial c}{\partial z} - D_{zz} \frac{\partial^2 c}{\partial z^2} = 0 \qquad (D_{zz} > 0).$$

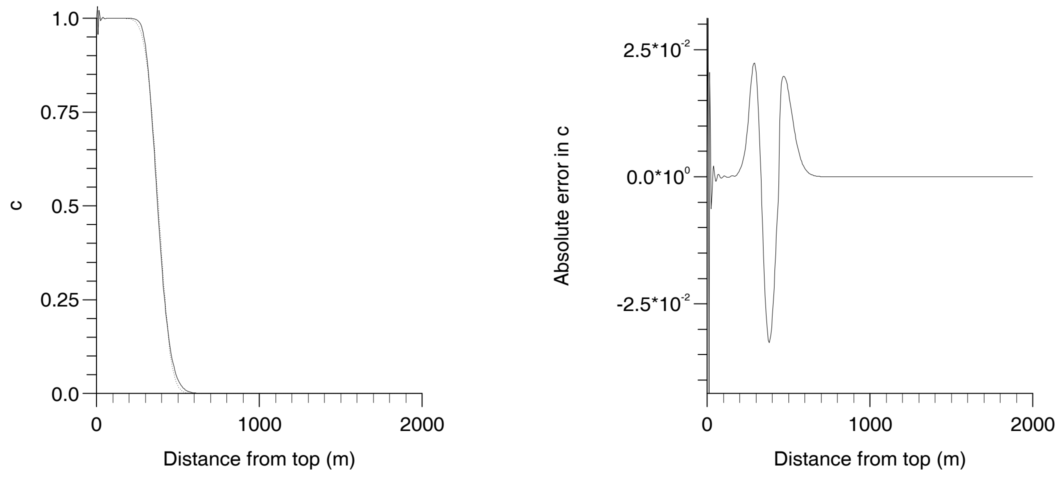
When this is discretised by the Galerkin finite element method, it is known (see for example [18, 91]) that unphysical oscillations occur in the discrete solution unless $Pe(= v_z \Delta z / D_{zz}) \leq 2$. Time dependent problems are the subject of this work so the steady state constraint only applies as the time-step tends to infinity.

For the time dependent case, it is noted in [50] that the discretisations based on the Crank-Nicolson method are prone to oscillations behind steep fronts when the Courant number exceeds unity. This is supported by analysis in [18] (with a lumped finite element mass matrix[1]) which leads to the additional constraint that $Co \leq 1$ for there to be no unphysical oscillations. The analysis is not valid for the distributed mass matrix used in this thesis; schemes with the distributed matrix being less diffusive and hence more prone to unphysical oscillations.

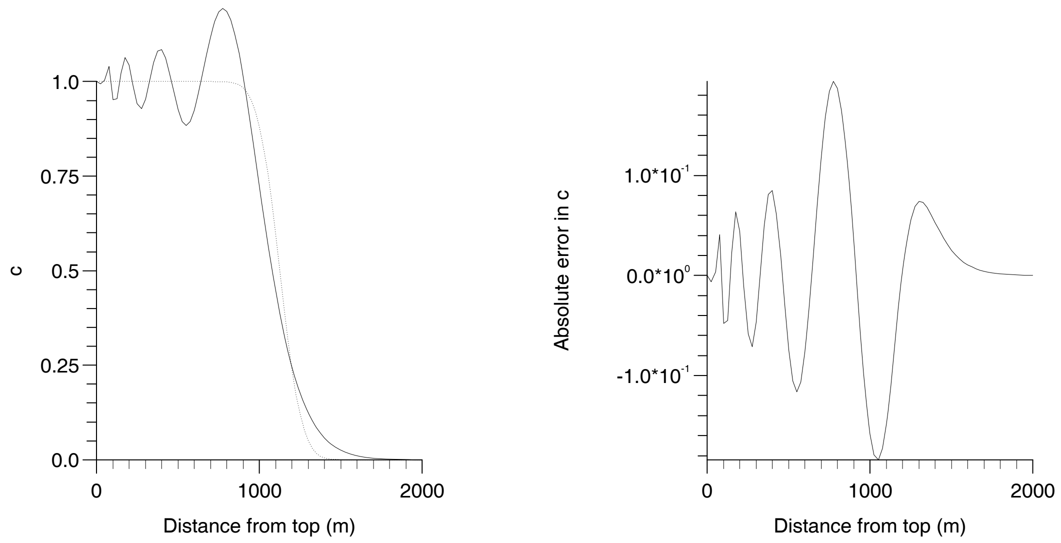
Even though the analysis doesn't apply to the method here, these constraints ($Co \leq 1$, $Pe \leq 2$) are generally accepted as reasonable guidelines. The results in Tables 4.1 and 4.2 support this.

Figure 4.3 shows unphysical oscillations in the numerical solution. In case (a), the oscillations are caused by the extremely steep initial gradient as the tracer enters the region - these oscillations remain close to the inflow boundary. As the front diffuses, it "appears" less steep (to the mesh) and the extrema near the front are hardly visible. In case (b), the mesh is five times coarser so the front appears to be five times steeper (to the mesh) - hence the unphysical oscillations are more pronounced.

[1]The element mass matrix given in Chapter 3 is the distributed one - the lumped element mass matrix is the diagonal matrix obtained by summing entries in the rows of the distributed element mass matrix and putting these sums on the diagonal. Lumping the mass matrix gives a formally less accurate scheme.



(a) $Co = 5,\ Pe = 1,\ t = 7.5 \times 10^6$ s



(b) $Co = 5,\ Pe = 5,\ t = 22.5 \times 10^6$ s

Figure 4.3: Sample solutions to illustrate unphysical oscillations

There are numerous ways to control these oscillations. For example

- **mesh refinement** - the size of the space and time-steps can be varied either (*i*) to bring the Courant and Peclet number within the bounds for which no unphysical oscillations occur or (*ii*) to reduce the size of the oscillations.
- **artificial diffusion** - extra diffusion is added to the numerical scheme to damp out the unphysical oscillations.
- **flux-corrected transport** - local averaging of a monotonicity preserving and a non-monotonicity preserving scheme to generate a new monotone

solution.

The remainder of this section is devoted to descriptions of these techniques.

Mesh refinement

This method for the control of unphysical oscillations involves varying the size of the space- and time-steps. Decreasing the size of the mesh decreases the truncation error in the discretisation, but it also increases the computational expense required to generate the solution. Hence global refinement of the discretisation is avoided and local refinement is used; that is, a small time or space step is used only when or where it is deemed necessary.

Local refinement in time - the unphysical maxima in Tables 4.1 and 4.2 decay with time so they must be caused by some feature of the initial conditions. As appears to be the case here, many unphysical oscillations are caused by steep profiles being introduced into a region through a boundary.

In this problem, the steep front which exists early in the simulation is diffused as time progresses. Hence a time-stepping strategy of the form

$$\Delta t_{i+1} = \min(\alpha \Delta t_i, \Delta t_{max}), \tag{4.3}$$

where the subscripts denote time levels, allows control of oscillations during the initial phase when the front is very steep. This time-step control relaxes with time, i.e. as the front spreads. Here, the parameter $\alpha \geq 1$, Δt_0 and $\Delta t_{\max}$ are user specified.

Table 4.3 shows solution features obtained when the time-stepping strategy,

$$\Delta t_{i+1} = \min(1.2\Delta t_i, 2.5 \times 10^6 \text{s}) \qquad \Delta t_0 = 0.25 \times 10^6 \text{s},$$

is used. As expected (since the Courant number is smaller during the critical initial stages of the simulation), the unphysical oscillations are smaller than in the corresponding cases with a uniform time-stepping (c.f. Tables 4.1 and 4.2). The errors are also smaller due to the (on average) finer temporal discretisation.

Local spatial grid refinement - oscillations occur when the mesh Peclet number is too large i.e. when the grid is not sufficiently fine to resolve local features (e.g. shocks and steep fronts) in the solution. In practice, as it is wasteful to refine

Pe	t ($\times 10^6$s)	c_{min}	$c_{max} - 1$	Relative Error
1.0	7.5	0	7.49×10^{-4}	1.38×10^{-2}
	15	0	2.34×10^{-4}	1.03×10^{-2}
	22.5	0	7.02×10^{-5}	7.52×10^{-3}
5.0	7.5	0	6.29×10^{-2}	4.77×10^{-2}
	15	0	1.32×10^{-1}	6.99×10^{-2}
	22.5	0	1.48×10^{-1}	7.43×10^{-2}

Table 4.3: Error and Extrema at $Co_{max} = 5$ with progressive time-stepping

globally, an adaptive local refinement strategy based on the solution is used. Nodes are moved or added in order to evenly distribute a strictly positive weight function such as the arc length. In this way, there are few computational points in regions where the solution is "uninteresting". The use of local spatial grid refinement for the control of unphysical oscillations is not investigated in this thesis.

Artificial Diffusion

In this method, extra diffusion is added to the numerical approximation to the problem to damp out the unphysical oscillations. The amount of this artificial diffusion decreases as the spatial and/or temporal mesh is refined so that the discretisation remains consistent with the original governing equation.

In [91], this is implemented as an addition to the coefficient of longitudinal dispersion in (3.5); and in [47], it is introduced as "anisotropic balancing dissipation" in the form of an extension of a Petrov-Galerkin method to 2-D. As with local spatial grid refinement, this method is not investigated in this thesis.

This approach is not to be confused with the Leismann-Frind scheme in which artificial diffusion is added to give stability with an explicit central discretisation of the advection term, not to control unphysical oscillations.

Flux-corrected Transport

Flux-corrected transport [9, 94] (FCT) uses local averaging of a monotonicity preserving and a non-monotonicity preserving scheme to generate a new monotone solution. In this section, the phrase "monotonicity preserving" is taken to mean that the scheme does not introduce unphysical new extrema to the numerical solution.

As the monotonicity preserving scheme is usually low order and the non-monotonicity preserving scheme is usually high order, these solutions are denoted by c^L and c^H respectively. The FCT method finds a weighted average of c^L and c^H which uses c^H almost everywhere and uses c^L only in places where the high order approximation struggles for monotonicity (e.g. due to steep fronts).

The (monotone) weighted approximation to the solution at the J^{th} node, c_J^M, is written as

$$c_J^M = \alpha_J c_J^H + (1-\alpha_J)c_J^L \qquad\qquad (0 \leq \alpha_J \leq 1), \tag{4.4}$$

where a suitable averaging is achieved by choosing each of the α_J such that c_J^M is monotone and the proportion of the high order approximation is maximised. (Note that it is always possible to ensure that c_J^M is monotone by taking $\alpha_J = 0 \;\forall\; J$.)

Local bounds on the solution values at node J, c_J^{max} and c_J^{min}, are required in order to determine α_J. Here these values are taken as the maximum and minimum solution values at the previous time level on the element which contains the trajectory from node J. Since the fluid velocity is constant for this test case, this element is the one which contains the point, z, given by

$$z = z_J - v_z \Delta t.$$

Since it follows the trajectory, this process for determining the local bounds is a Lagrangian technique. FCT is most easily used with Lagrangian schemes because the local bounds are available as a natural part of the scheme. Here, the scheme is Eulerian and the Lagrangian process is used purely to determine bounds for use in the application of FCT.

For monotonicity, i.e. in order to ensure that no new unphysical extrema are created,

$$\min(c_J^L, c_J^{min}) \leq c_J^M \leq \max(c_J^L, c_J^{max}),$$

i.e.

$$\min(c_J^L, c_J^{min}) \leq \alpha_J c_J^H + (1-\alpha_J)c_J^L \leq \max(c_J^L, c_J^{max}).$$

Subtracting c_J^L (where $c_J^L \geq 0$) gives,

$$\min(c_J^L, c_J^{min}) - c_J^L \leq \alpha_J(c_J^H - c_J^L) \leq \max(c_J^L, c_J^{max}) - c_J^L.$$

Useful bounds can be generated from this two-sided inequality by considering the three possible scenarios which can arise at each point.

<u>Case I</u>: If $c_J^H > c_J^L$, the left inequality is redundant since $\min(c_J^L, c_J^{min}) \leq c_J^L$, hence,

$$\alpha_J \leq \frac{\max(c_J^L, c_J^{max}) - c_J^L}{c_J^H - c_J^L}.$$

<u>Case II</u>: If $c_J^H < c_J^L$, the right inequality is redundant since $\max(c_J^L, c_J^{max}) \geq c_J^L$, hence,

$$\alpha_J \leq \frac{\min(c_J^L, c_J^{min}) - c_J^L}{c_J^H - c_J^L}.$$

<u>Case III</u>: If $c_J^H = c_J^L$, then c_J^M takes the same value regardless of the choice of α_J.

To maximise the amount of the high order approximation in the weighted average, the largest possible values of α_J must be taken. Hence, the value of α_J (enforcing the positive weighting $0 \leq \alpha_J \leq 1$) is,

$$\alpha_J = \begin{cases} \min\left\{1, \dfrac{\max(c_J^L, c_J^{max}) - c_J^L}{c_J^H - c_J^L}\right\} & (c_J^H > c_J^L) \\ \min\left\{1, \dfrac{\min(c_J^L, c_J^{min}) - c_J^L}{c_J^H - c_J^L}\right\} & (c_J^H < c_J^L) \\ 1 & (c_J^H = c_J^L) \end{cases} \tag{4.5}$$

Hence the FCT method is defined by the weighted average (4.4) with the weights given by (4.5). In order to demonstrate this method, FCT is applied to numerical solutions of (3.4). The non-monotonicity preserving scheme used to generate c^H is the Crank-Nicolson Galerkin finite element method (3.23). The only requirements of the scheme for generating c_L are that it should produce a monotone solution and not add significantly to the overall cost of the solution of the advection-diffusion. Next, a suitable scheme is presented and analysed.

The low order monotone solution is generated by a simple 1-D implicit upwind finite difference discretisation of (4.1), i.e.

$$\left(\frac{c_J^{n+1}-c_J^n}{\Delta t}\right)+v_z\left(\frac{c_J^{n+1}-c_{J-1}^{n+1}}{\Delta z}\right)-D_{zz}\left(\frac{c_{J+1}^{n+1}-2c_J^{n+1}+c_{J-1}^{n+1}}{(\Delta z)^2}\right)=0 \qquad (4.6)$$

where c_J^n is the approximation to the solution at (interior) node J and time level n and the node index increases in the direction of v_z. Fourier analysis shows that this scheme is unconditionally stable, and a Taylor series expansion shows it to be first order accurate in both space and time. The implicitness of the method is important as it ensures stability at Courant numbers exceeding unity i.e. in the regime where the high order scheme (3.23) is not monotonicity preserving.

This implicit upwind scheme is monotonicity preserving. In order to demonstrate this, first consider the difference between the discretisation at node J and node $(J+1)$, (i.e. the difference between (4.6) and the corresponding equation centred at node $(J+1)$), this is

$$\left(\frac{d_{J+1}^{n+1}-d_J^n}{\Delta t}\right)+v_z\left(\frac{d_J^{n+1}-d_{J-1}^{n+1}}{\Delta z}\right)-D_{zz}\left(\frac{d_{J+1}^{n+1}-2d_J^{n+1}+d_{J-1}^{n+1}}{(\Delta z)^2}\right)=0.$$

where $d_J^n = c_{J+1}^n - c_J^n$. The matrix system for the differences is

$$M\boldsymbol{d}^{n+1} = \boldsymbol{d}^n$$

where typically, the J^{th} equation, with $Co = v_z\frac{\Delta t}{\Delta z}$ and $\frac{Co}{Pe} = D_{zz}\frac{\Delta t}{(\Delta z)^2}$, is

$$\begin{pmatrix} \cdots & (-Co-\frac{Co}{Pe}) & (1+Co+2\frac{Co}{Pe}) & (-\frac{Co}{Pe}) & \cdots \end{pmatrix}\begin{pmatrix} \vdots \\ d_{J-1}^{n+1} \\ d_J^{n+1} \\ d_{J+1}^{n+1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \vdots \\ d_J^n \\ \vdots \\ \vdots \end{pmatrix}.$$

The matrix M is real and square. Since $Co \geq 0$ (due to the assumption on the ordering of the nodes) and $Pe \geq 0$, then M has non-positive entries on the off-diagonals, and positive entries on the diagonal. Hence, from [87] (Theorem 3.11 Corollary 1), M^{-1} is positive (i.e. contains only positive elements). Now,

$$\boldsymbol{d}^{n+1} = M^{-1}\boldsymbol{d}^{n},$$

so if the solution is monotone at time level n (i.e. all the entries in $\boldsymbol{d}^n$ have the same sign), then since $M^{-1} > 0$, the solution at time level $(n+1)$ is also monotone. Hence this scheme is monotonicity preserving.

(Note: For a multidimensional problem, a suitable low-order positive[2] discretisation is an operator splitting method in which the advection is approximated by a Lagrangian method (i.e. tracing particle trajectories) and the diffusion is approximated using the Galerkin finite element method with a lumped mass matrix. However, in the simple 1-D test case here, the particle trajectories are known straight lines so the approximation to the advection is exact. Hence, this simple Lagrangian method is not used here to illustrate FCT because it is *too good* for this particular problem, i.e. it does not provide a suitable low order solution.)

In order to avoid the effects of the initial stiffness on the convergence rates, the numerical experiments on the FCT method are run with initial data obtained from (4.2) at some time, $t_0 > 0$, after the contaminant has entered the region. This ensures that the initial profile is smooth and monotone. All the tests are run to the same end time, t_f, to allow comparisons of errors.

Table 4.4 shows some solution features at Courant number, $Co = 5$ (above unity so oscillations are expected near steep fronts) for the low order scheme (4.6), the high order scheme (3.23) and the FCT scheme (4.4). The refinement factor is defined as

$$\text{Refinement Factor} = \frac{\text{Relative error at time } t_f \text{ with discretisation } (\Delta z,\, \Delta t)}{\text{Relative error at time } t_f \text{ with discretisation } (2\Delta z,\, 2\Delta t)}.$$

The refinement factor can be used to determine the practical order of accuracy of the scheme.

The high order scheme is more accurate (i.e. has lower relative error) than the low order scheme for each of the discretisations used. The low order scheme has a

[2] *Positivity* is a multi-dimensional form of monotonicity preservation.

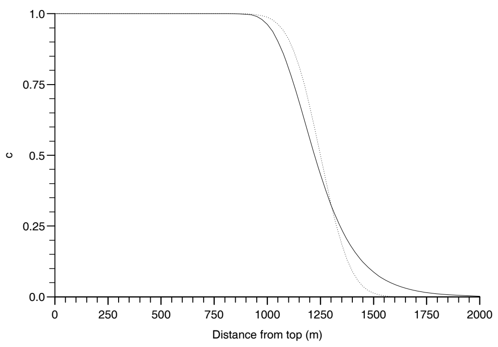
Scheme	Δz (m)	Δt ($\times 10^5$s)	c_{min}	$c_{max} - 1$	Relative error at t_f	Refinement Factor
Low order	50	50	2.69×10^{-2}	0	1.29×10^{-1}	–
	25	25	6.87×10^{-3}	0	8.90×10^{-2}	0.69
	12.5	12.5	7.04×10^{-4}	0	5.70×10^{-2}	0.68
	6.25	6.25	2.19×10^{-5}	0	3.40×10^{-2}	0.60
High order	50	50	1.22×10^{-3}	7.92×10^{-2}	6.17×10^{-2}	–
	25	25	4.91×10^{-5}	2.15×10^{-2}	2.49×10^{-2}	0.40
	12.5	12.5	4.78×10^{-7}	2.59×10^{-4}	7.63×10^{-3}	0.31
	6.25	6.25	0	2.48×10^{-8}	2.03×10^{-3}	0.27
FCT	50	50	1.23×10^{-3}	0	5.08×10^{-2}	–
	25	25	4.92×10^{-5}	0	2.10×10^{-2}	0.41
	12.5	12.5	4.77×10^{-7}	0	7.29×10^{-3}	0.35
	6.25	6.25	0	0	2.03×10^{-3}	0.28

Table 4.4: Performance of FCT at $Co = 5$ with $t_0 = 2 \times 10^7$s, $t_f = 2.5 \times 10^7$s

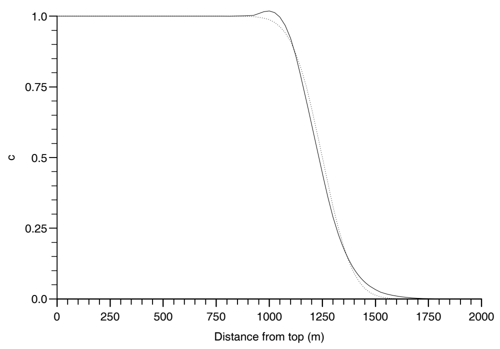
refinement factor that tends towards 0.5 ($= (\frac{1}{2})^1$) as the space and time-steps are refined - typical of a scheme that is first order in space and time. The high order scheme has a refinement factor that tends towards 0.25 ($= (\frac{1}{2})^2$), as expected from a scheme that is second order in both space and time. Overall, the FCT scheme has slightly lower relative errors than the high order scheme, and it also appears to have a refinement factor that tends towards 0.25.

The high order scheme has unphysical maxima while these are not present in either the low order or the FCT scheme. Figure 4.4 shows the solution for the low order, high order and FCT schemes and the variation of the weighting factor across the region at time t_f for the discretisation with $\Delta z = 25$m and $\Delta t = 25 \times 10^5$s . FCT removes the unphysical maximum which is clearly visible in Figure 4.4(b).

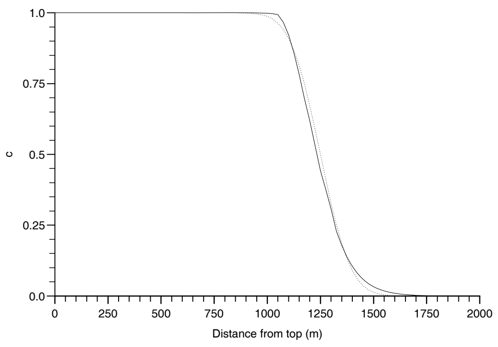
The FCT scheme is obviously more computationally expensive to implement than the high order scheme. However, in the numerical experiments conducted here, the FCT scheme requires less than 10% extra CPU time than the high



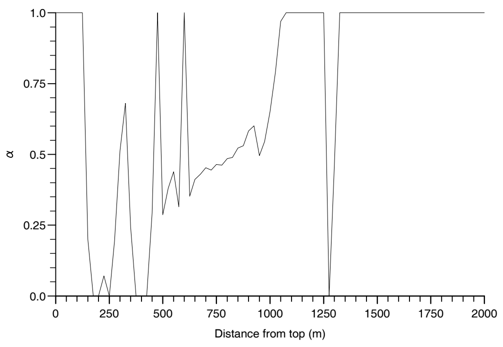
(a) Low order scheme



(b) High order scheme



(c) FCT scheme



(d) Variation of α

Figure 4.4: Illustration of FCT with $\Delta z = 25\text{m}$, $\Delta t = 25 \times 10^5\text{s}$ at $t = t_f$

order scheme alone (with exact solves on all the linear systems involved) so the overhead is not great.

The weights used to make the FCT weighted average are optimal in the sense that they maximise the amount of high order scheme in the solution while enforcing monotonicity. It is possible to recover other desirable properties in the solution by selecting sub-optimal weights; this technique is used to recover conservation in [65].

4.1.4 Concluding Remarks

The Crank-Nicolson Galerkin finite element method gives good results (i.e. second order in space and time and unconditionally stable) for the numerical solution of the contaminant mass balance equation. Putting aside accuracy considerations, the unconditional stability allows arbitrarily large time-steps to be used. However, the method is prone to unphysical oscillations when the Courant number exceeds unity. The techniques described in the Section 4.1.3 can be used to control these unphysical oscillations.

4.2 The Henry Problem

The Henry problem is a 2-D saturated groundwater flow problem which involves fresh-water in a confined aquifer[3] discharging to a vertical open sea boundary over a diffuse wedge of sea-water that has intruded into the aquifer. An approximate analytic solution to the Henry problem was given in the original paper [38], but no known numerical model matches this solution.

In [88], it is suggested that there is an inaccuracy in the approximate analytic solution caused by missing higher-order terms which were originally discarded to reduce computation time. Since a large number of numerical models give nearly identical results, this problem has been widely adopted for validation of variable density transport models by comparison with accepted results from the literature.

The first numerical solution of this problem [63] used a finite difference method to solve the fluid continuity equation and the "method of characteristics" (a Lagrangian approach) to solve the contaminant mass balance equation. Subsequent solution approaches have employed other Lagrangian methods (e.g. [30]) and the Galerkin finite element method (e.g. [40, 72]).

This test case is non-passive so, apart from the discretisation of the contaminant mass balance equation already examined in the tracer test case in the previous section, it allows the discretisation techniques for both the fluid continuity equation and components of the Darcy velocity vector to be examined. The overall solution technique is that given in Algorithm 3.2.

[3]An *aquifer* is a geological formation that contains water and permits significant amounts of this water to move through it under ordinary field conditions. A *confined* aquifer is one bounded above and below by impervious formations.

4.2.1 Specification of the Henry Problem

Figure 4.5 shows the physical domain and boundary conditions for the Henry problem (where ρ_s is the density of pure sea-water). The notation is consistent with that in Section 3.1.6. The confined aquifer is a 2m$\times$1m rectangular region, fresh-water enters on the right side and sea-water enters from the sea boundary on the left side. There can be no flow of water or salt through the horizontal faces.

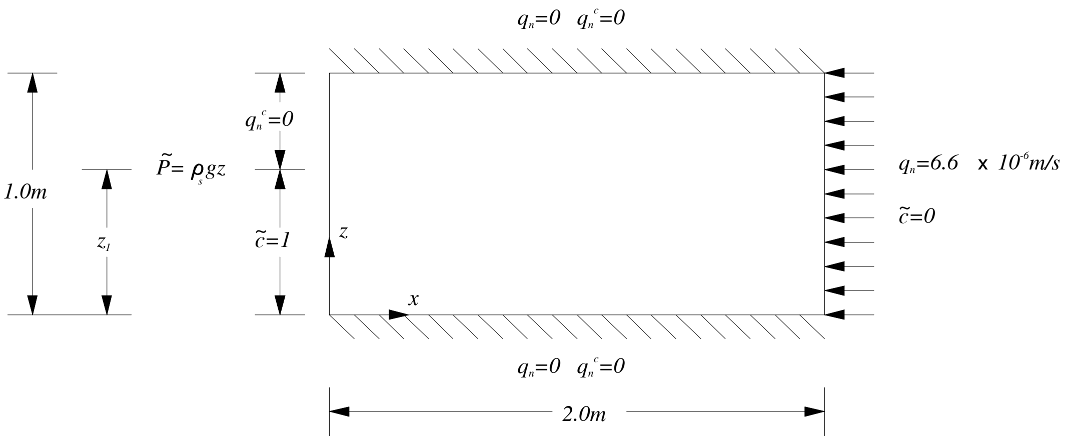


Figure 4.5: Physical Domain and Boundary Conditions for the Henry Problem

In the original problem, the dimensionless salt concentration is set to unity (i.e. pure sea-water) on the whole of the coastal boundary (the left face). However, this causes a conflict when the freshwater flowing into the region tries to exit at the Dirichlet sea-water face. In accordance with other authors [10, 30, 40, 72], this problem is avoided by changing the coastal boundary condition so that it consists of two components as shown in Figure 4.5. On the inflow part of the coastal boundary ($0 \leq z \leq z_1$) the dimensionless salt concentration is set to unity, while on the remainder of the coastal boundary ($z_1 < z \leq 1$m) the prescribed dispersive solute mass flux, q_n^c, is set to zero.

Now z_1 is not known *a priori*. In the literature, there are two methods used to determine this value. In the first (used in e.g. [40]), an initial iterate for z_1 is made and the problem is solved to the desired time with the position of z_1 fixed at the initial iterate. This solution is analysed and the problem is reformulated, but this time with z_1 at the position where the flow on the sea face boundary changes direction from inflow to outflow, and the problem is solved again to the desired time. By repeating this procedure iteratively, the correct position of z_1 can be obtained. In [40], convergence of this process is assumed, not proved.

In the second method, the position of z_1 is determined dynamically (i.e. while the time-stepping is being performed) according to

$$\begin{aligned} q_x \geq 0 & \quad \text{for} \quad 0 \leq z \leq z_1 \\ q_x < 0 & \quad \text{for} \quad z_1 < z \leq 1\text{m} \end{aligned}$$

where q_x is the horizontal component of the Darcy velocity vector, $\boldsymbol{q}$. Hence, when the flow is directed into the region, there is a Dirichlet boundary condition on the dimensionless salt concentration, but when the flow is directed out of the region, a Neumann boundary condition applies. Forms of this dynamic boundary condition can be found in [30, 72].

The second method for determining z_1 is used in this thesis because its flexibility allows transient features which depend on the position of z_1 to be modelled and only requires that the problem be solved once. In all tests, the value of z_1 used at the start of the simulation is 0.5m.

The values of the fluid, material and physical properties for the Henry problem test case are :

Physical	:	$g = 9.81 \text{ m/s}^2$
Fluid	:	$\mu_0 = 0.001$ Pa s , $\rho_0 = 1000 \text{ kg/m}^3$, $\epsilon = 24.99 \text{ kg/m}^3$
Material	:	$\phi = 0.35$, $\underline{\boldsymbol{k}} = 1.019368\underline{\boldsymbol{I}} \times 10^{-9} \text{ m}^2$
		(where $\underline{\boldsymbol{I}}$ is the 2×2 identity matrix).

There are two standard Henry problem test cases in the literature, the first (and original) is the constant dispersion coefficient case. In [29], it is noted that constant dispersion coefficients are not sufficient to represent the dynamics that occur in saline intrusion, so the second (more physical) case has velocity dependent dispersion coefficients. The dispersion parameters for these cases are given in Table 4.5.

In this test case, a uniform triangular grid is used with 21 nodes in the x-direction and 11 nodes in the z-direction (giving a total of 400 elements). The grid is shown in Figure 4.6.

Case	α_L(m)	α_T(m)	D_m(m^2/s)
Constant dispersion coefficient	0	0	6.6×10^{-6}
Velocity dependent dispersion coefficient	0.035	0.035	0

Table 4.5: Values of parameters in dispersion tensor for Henry problem test cases

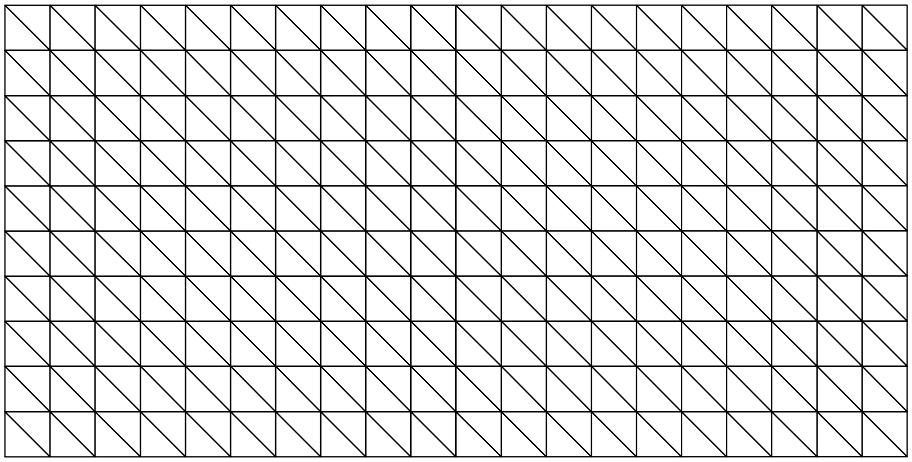


Figure 4.6: Grid for Henry problem

For this problem, the Courant number is

$$Co = \frac{|\boldsymbol{q}|}{\phi}\frac{\Delta t}{\Delta},$$

and Peclet number is

$$Pe = \frac{|\boldsymbol{q}|}{\phi}\frac{\Delta}{\|\underline{\boldsymbol{D}}\|_2},$$

assuming $\boldsymbol{\nabla}\rho$ and $\boldsymbol{\nabla}\underline{\boldsymbol{D}}$ are negligible. Here Δ is a representative spatial mesh size. There are many ways of defining a representative mesh size which give different Courant and Peclet numbers for the same problem. The actual forms used in this thesis lead to

$$Co = \frac{\Delta t}{\phi}\sqrt{\left(\frac{q_x}{\Delta x}\right)^2 + \left(\frac{q_z}{\Delta z}\right)^2}$$

and

$$Pe = \frac{\sqrt{(q_x\Delta x)^2 + (q_z\Delta z)^2}}{\phi\|\underline{\boldsymbol{D}}\|_2}.$$

Initially, the salt concentration everywhere inside the region is taken to be zero. As with the 1-D tracer test case, when the salt initially enters the region, the problem is quite stiff so unphysical extrema are expected. In order to control these unphysical extrema, the progressive time-stepping strategy,

$$\Delta t_{i+1} = \min(1.2\Delta t_i, 600\text{s}) \qquad \Delta t_0 = 15\text{s}$$

is used (where the subscripts denote successive time levels). This is similar to the time-stepping approach used in [30]. The other oscillation control techniques described in Section 4.1.3 were not found to be necessary for this problem.

As it is the properties of the discretisations that are being examined in this chapter, the tolerance used in the coupling convergence criterion (Step 8 of the coupling iteration given in Algorithm 3.2) is set impractically low. The purpose of this is to remove any errors the iteration introduces. The actual convergence criterion used is that the pointwise relative difference in the fluid density between two successive coupling iterations changes by no more than 10^{-15}. The effect of the coupling on the overall process is considered in Chapter 7.

4.2.2 Results for Constant Dispersion Case

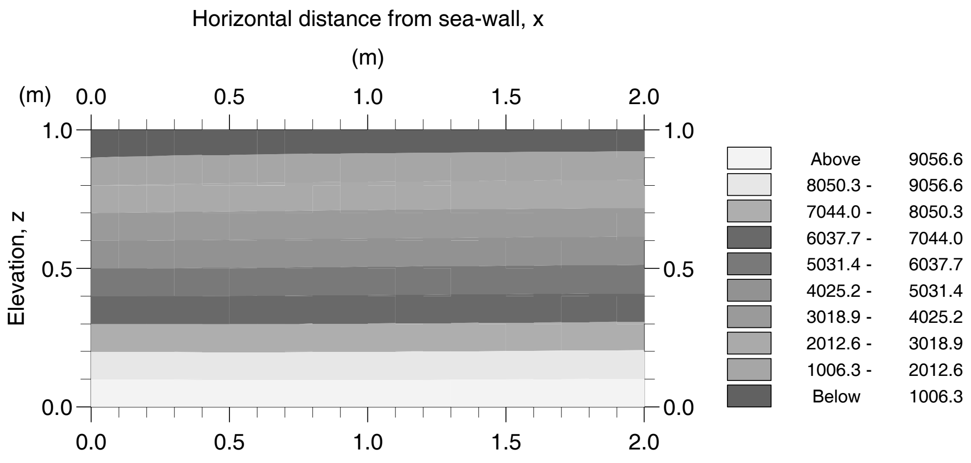
In this section, the performance of the overall numerical solution approach on the constant dispersion coefficient case from Table 4.5 is examined.

Using the mesh and time-stepping strategy already described, the initial Courant number and Peclet numbers are 0.18 and 13.01 respectively. At time t=100 min., the maximum Courant number which has occurred during the time history is 6.65 and the maximum Peclet number is the initial value.

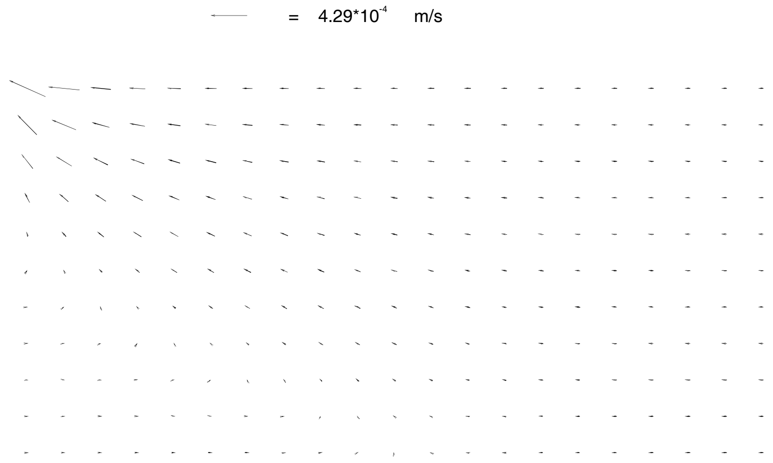
The pressure at time t=100 min. (which results from Stage 2 of Algorithm 3.2) is shown in Figure 4.7(a).

The horizontal isobars indicate that gravity, rather than the pressure gradient arising from the boundary conditions, is the predominant influence in this system.

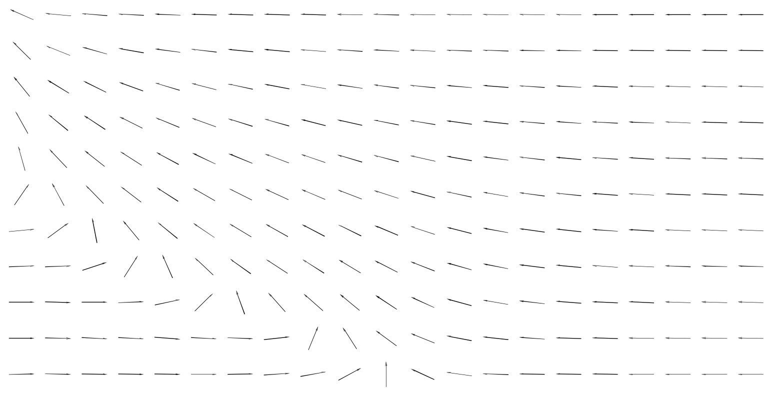
The resulting Darcy velocity field (from Stage 3 of Algorithm 3.2) is shown in Figure 4.7(b). The size of the arrows in this figure indicate the magnitude of the velocity. Due to the wide range of velocities which exist in the solution, the direction of these velocities is hard to see and the unscaled Darcy velocity field is



(a) Pressure (in Pa)



(b) Darcy velocity field



(c) Unscaled Darcy velocity field

Figure 4.7: Pressure and Darcy velocity at t=100 min.

included in Figure 4.7 in order to show the flow direction more clearly.

From the plots of the velocity field, the intrusion of the salt-water into the region at the lower part of the left boundary can be seen, as can the entry of freshwater at the right boundary. Most of the freshwater leaves at the upper part of the left boundary but some mixing of the two fluids occurs in the region of varying salt concentration. The position of z_1 on the dynamic sea boundary (see Figure 4.5) at time t=100 min. is $z_1 = 0.5$m.

Figure 4.8 shows the positions of the salt isochlors (lines of constant concentration) at time t=100 min (resulting from Stage 5 of Algorithm 3.2). The set of isochlor contours that are plotted is $\{-0.1, 0.1, 0.3, 0.5, 0.7, 0.9, 1.1\}$ - this is the standard set of contours used in the literature for the Henry problem. The salt forms the expected wedge shape.

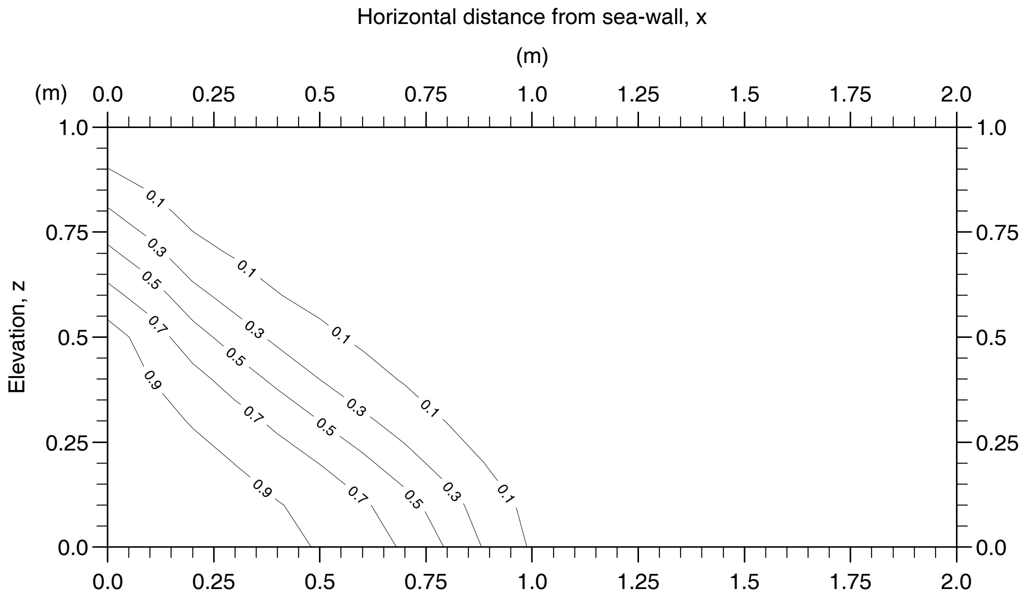
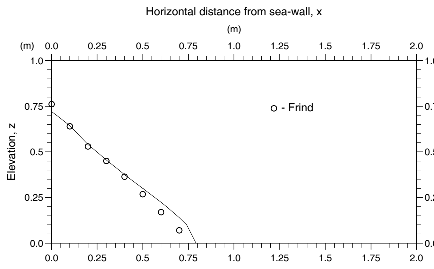


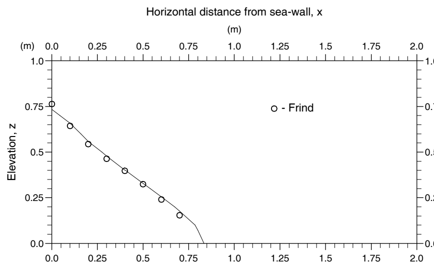
Figure 4.8: Salt isochlors at t=100 min.

In the literature, Henry problem results are given for the positions of the salt isochlors only. The isochlors produced by the methods used in this thesis are compared to some of those from the literature for verification of the overall model.

Comparisons of the position of the 0.5 isochlor at various times with results from [29, 72] are shown in Figures 4.9 and 4.10 - there is good agreement, showing the transient accuracy of the whole discretisation approach for this problem.

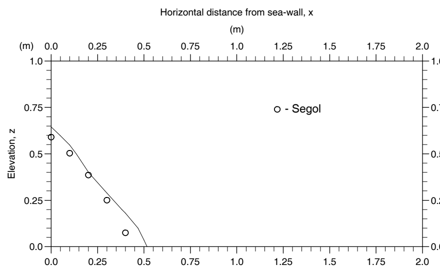


(i) t=100 min.

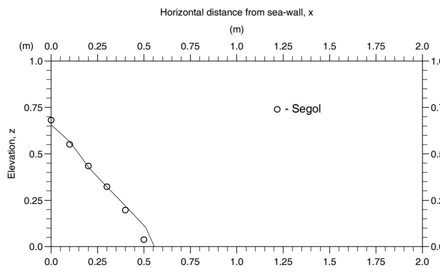


(ii) At equilibrium (t = 300 min.)

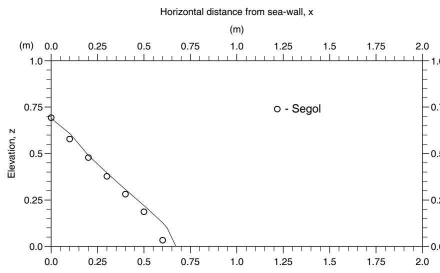
Figure 4.9: Comparison of position of 0.5 isochlor with [29]



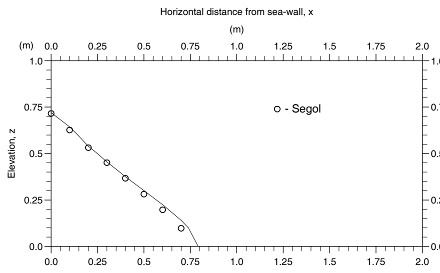
(i) t=25 min.



(ii) t=30 min.



(iii) t=50 min.



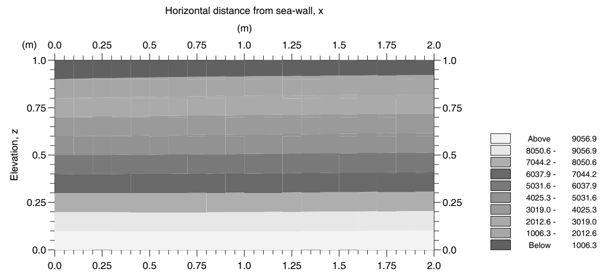
(iv) t=100 min.

Figure 4.10: Comparison of position of 0.5 isochlor with [72]

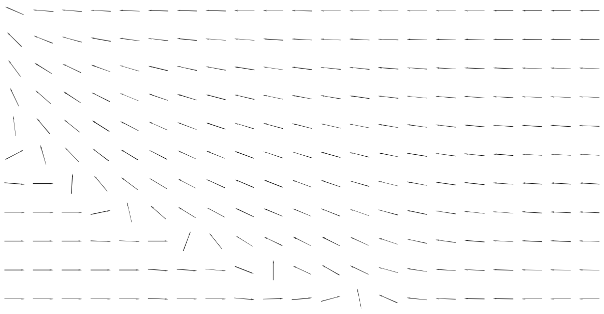
4.2.3 Results for Velocity Dependent Dispersion Case

In this section, the performance of the numerical discretisation approach on the velocity dependent dispersion coefficient case from Table 4.5 is examined.

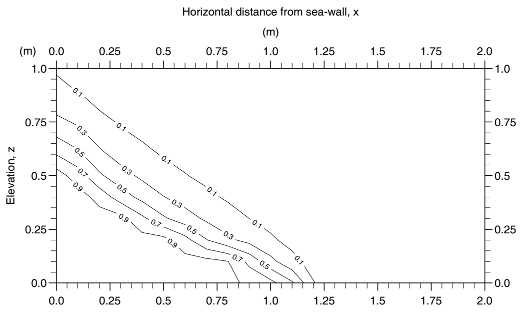
The initial Courant number and Peclet numbers are 0.18 and 6.36 respectively. At t=360 min., the maximum Courant number which has occurred during the time history is 6.59 and the maximum Peclet number is the initial value. Figure 4.11 shows the pressure, direction of Darcy velocity vector and position of the salt isochlors at time t=360 min.



(a) Pressure (in Pa)



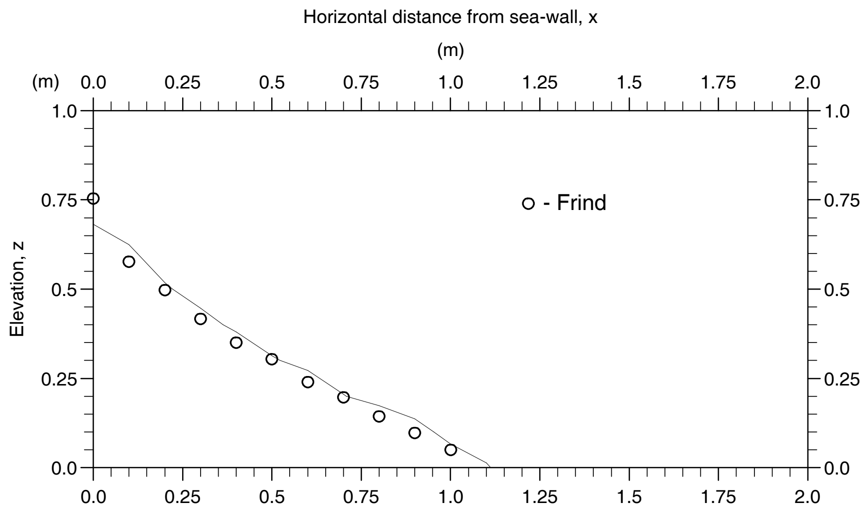
(b) Darcy velocity (direction only)



(c) Salt isochlors

Figure 4.11: Solution at time t=360 min. in Velocity Dependent Dispersion Case

Compared with the constant dispersion coefficient test case in the previous section, there are fewer results for the velocity dependent case in the literature. Figure 4.12 shows a comparison of the position of the 0.5 isochlor at equilibrium (taken to be when $t = 720$ min.) with results from [29].



At equilibrium ($t = 720$ min.)

Figure 4.12: Comparison of position of 0.5 isochlor with [29]

As noted in [29], in the physical problem, there is a stagnation point at the bottom of the aquifer at the dynamic equilibrium. This phenomenon cannot be modelled in the constant dispersion coefficient case as there will always be diffusion even when the flow is zero. However, the velocity dependent dispersion coefficient is a mechanism which allows this feature to be modelled. In the numerical solution of the velocity dependent dispersion coefficient case, the minimum nodal velocity that occurs at equilibrium is 10^{-6}m/s and occurs at (0.9m,0.1m). This can be compared with the constant dispersion coefficient case where the minimum nodal velocity is 1.077×10^{-5}m/s at (0.8m,0.1m) - i.e. the minimum velocity in the velocity dependent case is an order of magnitude smaller.

4.3 Concluding Remarks

The tracer test case gives quantitative results for the discretisation of the contaminant mass balance equation. The discretisation method is demonstrated to give a solution that is second order accurate in both space and time and unconditionally stable. Its only disadvantage is that it is prone to generating unphysical oscillations in the solution. Techniques for the control of unphysical oscillations have also been given (with particular attention being paid to the flux-corrected transport method).

The results from the Henry test case are more qualitative, and show that the numerical solution approach used gives results that are in good agreement with those from the literature for a more realistic saline intrusion problem.

In the following two chapters, the performance of the iterative methods that are used to solve the discretised governing equations is examined.

Chapter 5

Performance of the Symmetric Positive Definite Solver

In the previous chapter, all the linear systems which occurred were solved "exactly" (that is, to the limits allowed by finite precision arithmetic) since it was the properties of the discretisations that were being examined. No consideration was given to the cost this incurs. As stated in Chapter 2, this approach is not feasible for very large sparse systems and, in general, iterative methods must be used to solve these systems approximately.

The focus of this chapter (and the next one) concerns the performance of the iterative solvers. This part of the work is divided into two chapters in order to allow the symmetric and non-symmetric solvers to be examined separately. In the current chapter, the performance of the preconditioned CG iterative solver (Algorithm 2.3) on the linear systems with SPD matrices is examined. These matrices arise during the computation of the fluid continuity and Darcy velocity vector. The theoretical and practical properties of the solver are well understood and documented in the literature (see Section 2.3.1) - the main purpose of this chapter are to illustrate these properties, and to introduce the types of tests that are carried out in the investigation of the performance of the non-symmetric solvers in Chapter 6.

The matrices used in the tests in the following two chapters are taken from the same problems as the discretisation test cases in Chapter 4. However, as the 1-D tracer test case is passive, it does not require the discrete fluid continuity equation

(3.19) nor the discrete Darcy velocity vector equation (3.20) to be solved. Hence it does not require the solution of any systems with symmetric matrices, so the systems used in the tests on the symmetric solver are representative ones taken from the Henry problem.

Since the matrices in the discrete fluid continuity equation and the discrete Darcy velocity vector equation have different properties, the tests are carried out separately on the matrices from these equations.

To generate "representative" matrices, the constant dispersion Henry problem test case (Section 4.2) is run under the following conditions :

- The variable time-stepping (4.3) is used with $t_0 = 0$, $\Delta t_0 = 15$s, $\alpha = 1.0$.
- The convergence criterion on the coupling iterations is the same as that used in the previous chapter, i.e. the pointwise relative difference in the density from one coupling iteration to the next is less than 10^{-15}.
- The mesh is the one shown in Figure 4.6 i.e. 21×11 (resulting in a matrix of size $n = 231$).
- All linear systems (apart from the one being examined) are solved "exactly", i.e. $\tau = 10^{-15}$. Again, no consideration is given to the cost this incurs.

The representative matrix system is taken as the one generated in the first coupling iteration of the 10th time-step. Variations around this "representative" matrix system are taken to investigate behaviour further.

In order to exploit the sparsity in the matrix, the system used to hold and access the sparse matrix is compressed row storage [5].

5.1 Matrix from Fluid Continuity Equation

The matrix in the discrete fluid continuity equation (3.19) is the finite element stiffness matrix. This is symmetric and semi-definite but, with the imposition of the physical boundary conditions associated with the problem, is SPD. It is therefore a candidate for solution by CG.

5.1.1 Low Tolerance Test

The convergence criterion is (2.12) applied to the preconditioned system, that is convergence is taken to have occurred when

$$\|Z^{-1}\boldsymbol{r}_i\|_2 \leq \tau\|Z^{-1}\boldsymbol{b}\|_2.$$

As a first test, a tolerance, τ of 10^{-9} is requested and the diagonal preconditioner is used. A typical convergence history for CG on the representative matrix system is shown in Figure 5.1. This figure shows the preconditioned recurrence residual

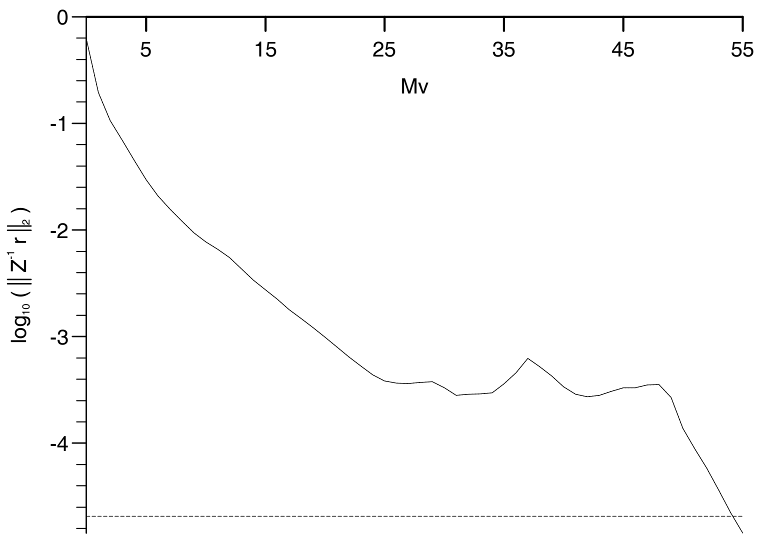


Figure 5.1: Iteration history for CG solver with low tolerance

which is obtained automatically during the iteration, i.e. from Algorithm 2.3,

$$Z^{-1}\boldsymbol{r}_i = Z^{-1}\boldsymbol{r}_{i-1} - \alpha_i Z^{-1}A\boldsymbol{p}_i$$

as the solid line, and the preconditioned true residual (which must be generated at the extra expense of another matrix vector multiplication) i.e.

$$Z^{-1}\boldsymbol{r}_i = Z^{-1}(\boldsymbol{b} - A\boldsymbol{x}_i)$$

as the dotted line - this second line is not visible because the two residuals are in good agreement. The target residual for convergence ($= \tau\|Z^{-1}\boldsymbol{b}\|_2$) is also shown in the iteration history in Figure 5.1 - this is the horizontal dashed line at approximately -4.7.

Matrix-vector multiplications are generally the most expensive part of the algorithm. Hence these residuals are plotted against the number of matrix-vector multiplications performed (Mv) (not including extra ones used to calculate true residuals). For CG, since there is one matrix-vector multiplication per iteration, Mv is equivalent to the number of iterations. However, some of the iterative solvers used on the non-symmetric matrices in the next chapter use two (or more) matrix-vector multiplications per iteration. In that case Mv is a better measure than the number of iterations for the relative performance of the iterative methods, hence Mv is used as the abscissa on iteration histories in this thesis.

Convergence is quite slow considering the size of the system ($n = 231$) and it does not appear to be monotone (a local maximum at $Mv = 37$ is clearly visible). This lack of monotonicity is not surprising since (from Section 2.3.1) it is the A^{-1}-norm of the residual, $\|\boldsymbol{r}_i\|_{A^{-1}} (= \|\boldsymbol{e}_i\|_A)$ that is minimised in CG, not the 2-norm $\|\boldsymbol{r}\|_2$. In order to illustrate this, Figure 5.2 shows the iteration history of $\|\boldsymbol{e}_i\|_A$ on the same case as Figure 5.1. The error is calculated by comparing each iterate

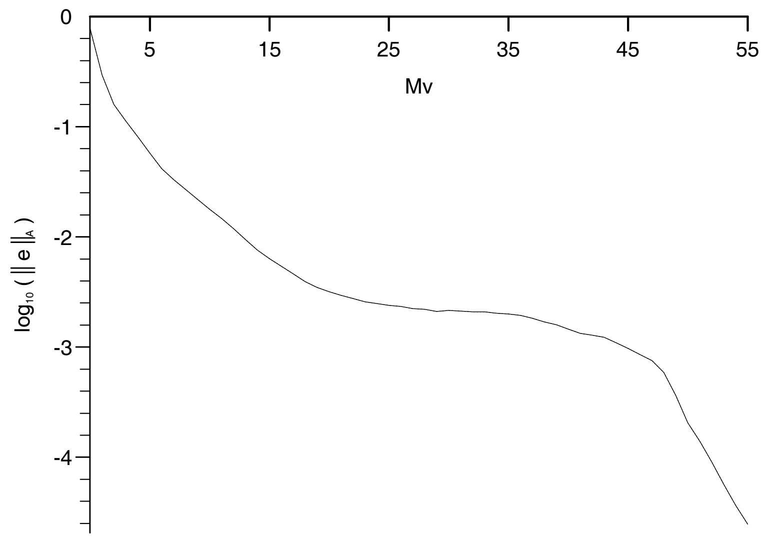


Figure 5.2: Iteration history (showing A norm of error) for CG solver with low tolerance

with an "exact" solution (e.g. the one used was the result of a successful iteration with $\tau = 10^{-15}$). In agreement with theory, monotone convergence is achieved.

If the representative matrix is taken at a point later in the time history, convergence is faster, e.g. only 34 iterations are required at the 100th time-step. This is due to the smaller distance from steady state; that is, at later points in the time history, the initial iterate (taken as the solution at the previous time-step) is "closer" to the required solution at this time-step.

5.1.2 High Tolerance Test

To fully test the solver, the convergence tolerance is tightened to $\tau = 10^{-15}$. This tolerance is much harsher than would usually be requested in practice. It is used to test the solver near the limits of the finite precision arithmetic used.

The resulting iteration history is shown in Figure 5.3.

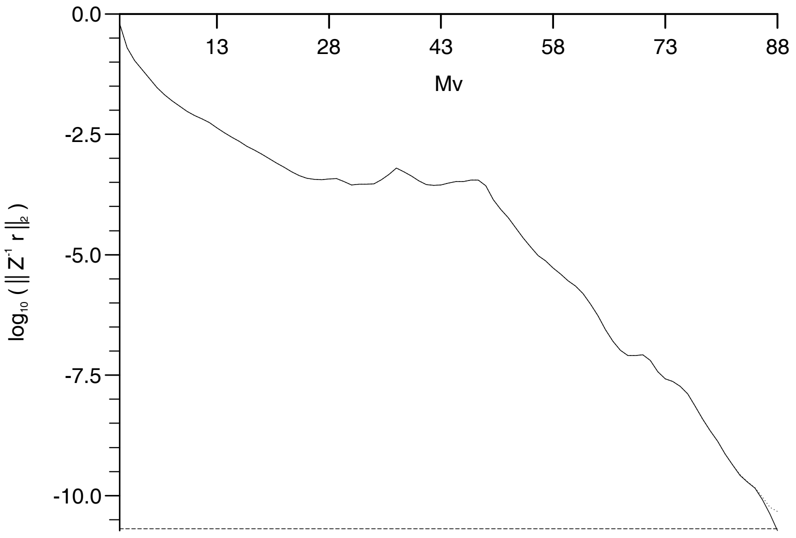


Figure 5.3: Iteration history for CG solver with high tolerance

The recurrence residual deviates slightly from the true residual near the convergence tolerance (due to rounding error in the recursion process becoming more important in that regime). Apart from the lack of monotonicity in the 2-norm, convergence is still relatively direct, if somewhat slow. A more effective preconditioning matrix can be used to accelerate the convergence - this aspect of the solver will be returned to later in this chapter.

5.1.3 Mesh Dependence of Convergence

If a finer mesh is used in the high tolerance test in the previous section, more iterations are needed. This appears to scale proportionately to $\sqrt{n}$ (where n is the number of nodes in the mesh) for this problem, e.g. the problem on an 11×6 mesh needs 42 iterations for convergence, on a 21×11 mesh it needs 88, while on a 41×21 mesh it needs 177.

The dependence of the convergence rate on the size of the elements (i.e. the number of nodes in the discretisation) has already been noted. Practical problems usually involve non-uniform meshes and spatially variable physical properties. Both of these effects can lead to difficulties for conjugate gradient-type solvers. To illustrate this, some high tolerance tests are run on distorted meshes. The standard mesh for the Henry problem (Figure 4.6) with $n_x \times n_z$ nodes has its (i,j)th node at

$$\left(x_{min} + \frac{(i-1)}{(n_x-1)}(x_{max} - x_{min}) \ , \ z_{min} + \frac{(j-1)}{(n_z-1)}(z_{max} - z_{min}) \right) .$$

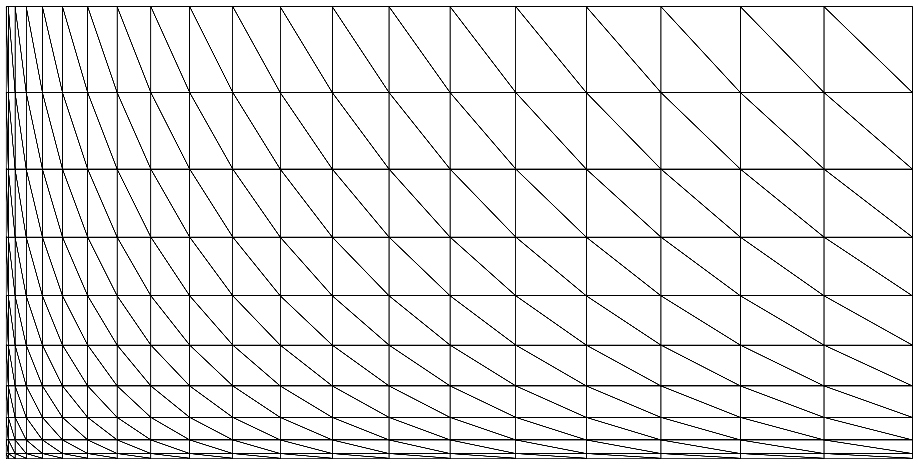
This mesh is distorted by keeping the same connectivity between nodes but taking the (i,j)th node to be at

$$\left(x_{min} + \frac{(i-1)^p}{(n_x-1)^p}(x_{max} - x_{min}) \ , \ z_{min} + \frac{(j-1)^p}{(n_z-1)^p}(z_{max} - z_{min}) \right)$$

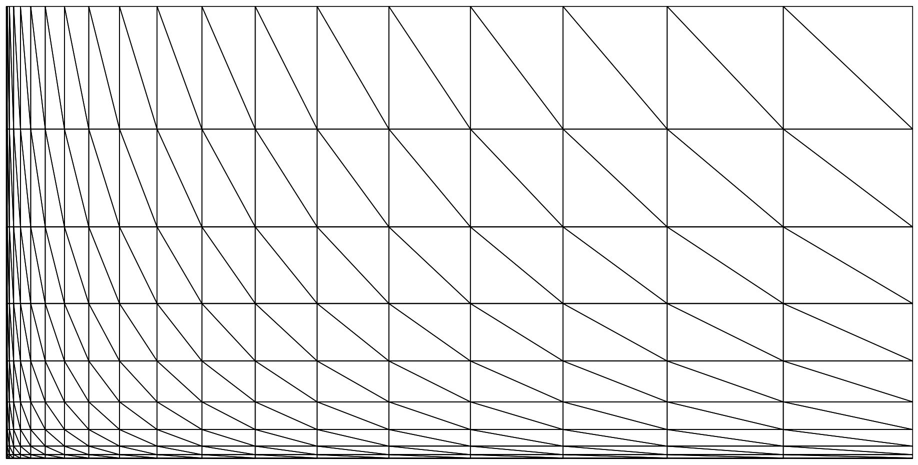
where $p(\in \mathbb{N}) = 1$ gives the standard (linear) mesh, $p = 2$ gives a quadratic mesh, $p = 3$ gives a cubic mesh, etc. The quadratic and cubic meshes are shown in Figure 5.4.

Most of the important activity in the Henry test case occurs in the bottom left corner of the domain (in the orientation shown in Figure 4.5). Hence, the distorted meshes used here are similar to those that would be used if spatial mesh refinement was deemed necessary to increase accuracy or control oscillations in the regions of steep solution gradient (as described in Section 4.1.3).

Figure 5.5 shows the iteration histories for the representative problem on the distorted meshes. The harsh convergence criterion is used for these tests (i.e. $\tau = 10^{-15}$ in (2.12)). The rate of convergence decreases as the amount of distortion increases, and the lack of monotonicity in the iteration history becomes more apparent.

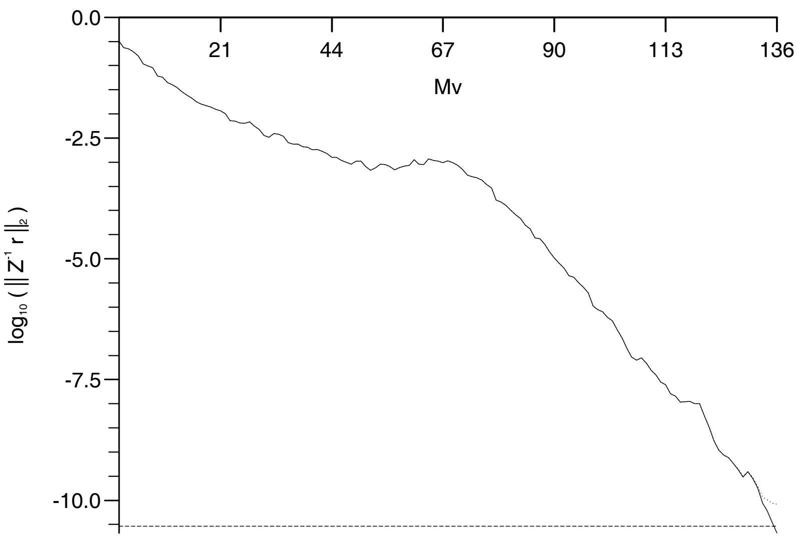


(a) Quadratic mesh

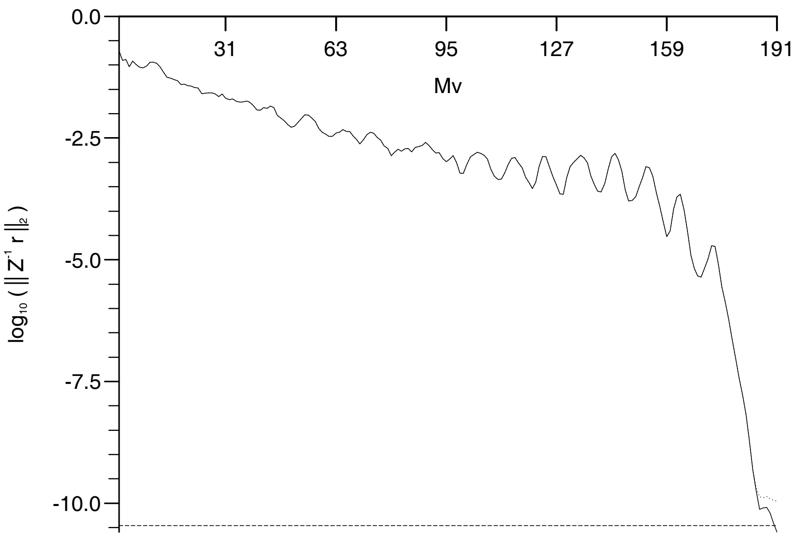


(b) Cubic mesh

Figure 5.4: Distorted meshes



(a) Quadratic



(b) Cubic

Figure 5.5: Iteration histories on distorted meshes

5.1.4 Preconditioning

Mesh dependent convergence rates are undesirable as they mean that any mesh refinement used to improve the quality of the numerical solution has a detrimental effect on the performance of the linear solver.

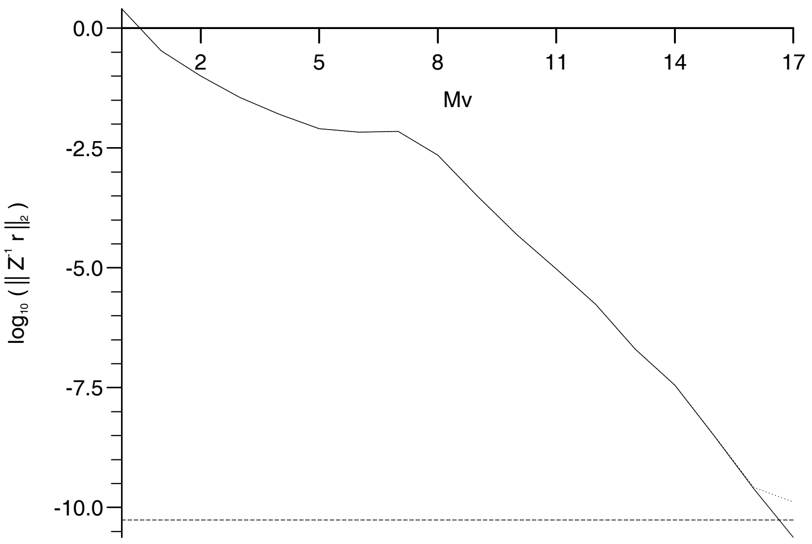
The usual way of overcoming mesh dependent convergence is to incorporate information on the irregularity of the mesh (or the physical properties) into the formulation of the preconditioning matrix. At the very least, this involves the use of off-diagonal terms in the matrix. A simple example of a suitable preconditioning matrix is the Incomplete LDL^T (or $ILDL^T$) factorisation which is the symmetric version of the factorisation given by Algorithm 2.4. As already stated, preconditioning is also used to accelerate the convergence.

The properties of the $ILDL^T$ preconditioner depend on the ordering of the equations in the system. Natural ordering is used to label the nodes in the meshes in this thesis with the nodes being numbered fastest in the vertical direction. Figure 5.6 shows the iteration histories for the representative problem on the distorted meshes with an $ILDL^T$ preconditioner. As in the previous section, the harsh convergence criterion is used (i.e. $\tau = 10^{-15}$ in (2.12)).

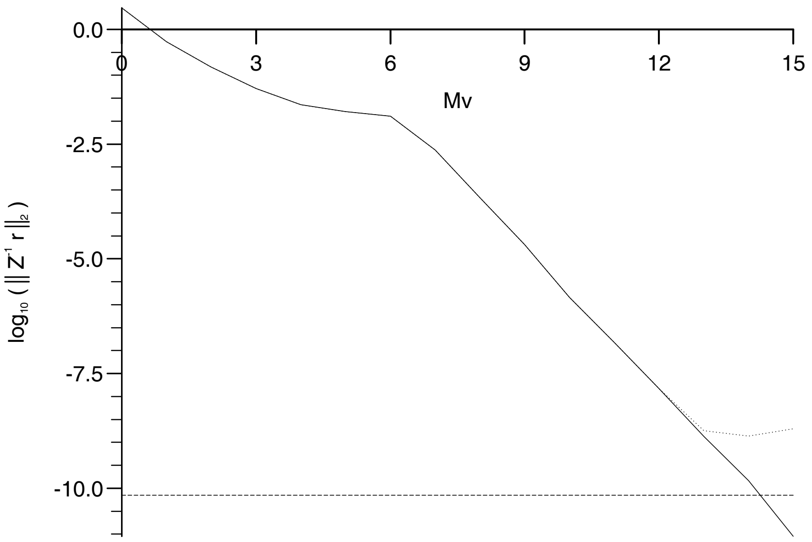
Comparing Figures 5.5 and 5.6, convergence is much smoother and acceptably faster (in terms of the number of matrix-vector multiplications) with an $ILDL^T$ preconditioner. In both cases, convergence is achieved in fewer iterations than with the diagonal preconditioner.

$ILDL^T$ preconditioned CG achieves a convergence rate for this problem which is reasonably mesh-independent; that is, the number of matrix-vector multiplications required to achieve convergence stays approximately constant as the mesh changes, only varying from 15 to 20 iterations as the mesh is distorted. In fact, the number of iterations required by the $ILDM^T$ preconditioned CG decreases as the mesh is distorted, this may be because the distorted meshes are more able to represent the "interesting" part of the region so that the initial iteration at a given time step is more accurate.

This discussion of mesh independent convergence only considers the distortion of the mesh. The rate of convergence still varies quite markedly with the number of points in the mesh, even with the $ILDL^T$ preconditioner (e.g. the problem on



(a) Quadratic



(b) Cubic

Figure 5.6: Iteration histories with $ILDL^T$ preconditioner

an 11×6 mesh needs 12 iterations for convergence, on a 21×11 it mesh needs 20, while on a 41×21 it needs 35). If truly mesh independent convergence is required, a more powerful preconditioner must be used; for example, an incomplete factorisation which allows some degree of fill-in. The use of such a preconditioner is not examined here.

Given that the $ILDL^T$ preconditioner is more expensive to compute (and "invert") at each iteration than a diagonal preconditioner, it is an issue whether its use leads to a faster approach. Table 5.1 shows the CPU time spent in the solver routines for both the diagonal and $ILDL^T$ preconditioners. These solver timings indicate that $ILDL^T$ is indeed faster for this particular problem.

Mesh	Number of Mvs		Time in solver[1]/ seconds	
type	Diagonal	$ILDL^T$	Diagonal	$ILDL^T$
Linear	88	20	0.93	0.56
Quadratic	136	17	1.42	0.48
Cubic	191	15	1.99	0.43

Table 5.1: Performance of diagonal and $ILDL^T$ preconditioners

5.1.5 Comparison of CG with SOR

In order to compare the performance of CG with the classical splitting iterative methods (Section 2.2), successive over-relaxation (SOR) is used to solve an "easy" test case ($\tau = 10^{-9}$ on a 21×11 linear mesh) and a "hard" test case ($\tau = 10^{-15}$ on a 21×11 cubic mesh). The results are shown in Tables 5.2 and 5.3 respectively for different values of the SOR relaxation parameter, ω.

Note that preconditioning is not used in SOR but the (diagonally) preconditioned residual is monitored to allow comparison with diagonally preconditioned CG. In order to facilitate this comparison, one SOR iteration is taken to require the same amount of computation as one matrix-vector multiplication.

[1]These and, unless otherwise stated, all subsequent solver timings were carried out using the SUN OS FORTRAN library (4.1) routine `dtime` on a SPARCstation 1+.

ω	Iterations required to achieve convergence
1.00	453
1.10	359
1.20	282
1.30	216
1.40	158
1.50	105
1.60	48
1.65	32
1.70	27
1.75	44
1.80	71
1.90	92

Table 5.2: Performance of SOR on “easy” test case

ω	$\min \|Z^{-1}\boldsymbol{r}\|_2$ after 462 $(= 2n)$ iterations
1.00	7.42×10^{-4}
1.10	5.14×10^{-4}
1.20	3.54×10^{-4}
1.30	2.42×10^{-4}
1.40	1.63×10^{-4}
1.50	1.08×10^{-4}
1.60	7.51×10^{-5}
1.70	6.00×10^{-5}
1.80	4.73×10^{-5}
1.85	3.33×10^{-5}
1.90	1.28×10^{-5}
1.95	1.24×10^{-7}
1.99	2.66×10^{-3}

Table 5.3: Performance of SOR on “hard” test case

From Table 5.2, the optimum SOR relaxation parameter for the matrix from the easy test case is $\omega \approx 1.7$. When $1.6 \leq \omega \leq 1.8$, convergence is faster than diagonally preconditioned CG on the same problem (that taking 55 matrix-vector multiplications). Techniques exist for generating approximations to the optimal SOR parameter which suggests that SOR(ω_{opt}) is a strong rival to CG for this problem. However, from Table 5.3, SOR does not achieve the required convergence tolerance for the hard test case with any value of the relaxation parameter so SOR is not as effective on problems with distorted meshes. Also, due to the asymptotic nature of its convergence, SOR struggles on problems where a tight tolerance is requested. Diagonally preconditioned CG converges in 191 iterations in this case and the use of an $ILDL^T$ preconditioner dramatically reduces this (to 15 iterations).

5.2 Matrix from Darcy Equation

The matrix in the discrete Darcy velocity vector equation (3.20) is the finite element mass matrix which, even before the imposition of physical boundary conditions, is SPD. As with the matrix from the discrete fluid continuity equation in the previous section, it is a candidate for solution by preconditioned CG.

There are two velocity systems to solve, one for each component of the 2-D velocity. Arbitrarily, the representative system is taken to be the one for the z-component. As before, this representative system is taken as the one generated in the first coupling iteration of the 10th time-step.

5.2.1 Low Tolerance Test

Figure 5.7 shows the same information as Figure 5.1 but for the representative matrix system of the z-component of the Darcy velocity vector equation. As in Section 5.1.1, a tolerance of 10^{-9} is requested in the convergence criterion (2.12) and a diagonal preconditioner is used.

Again, there is no visible difference between the preconditioned recurrence residual and the true preconditioned residual. Convergence is obtained in a reasonable number of iterations and is quite direct.

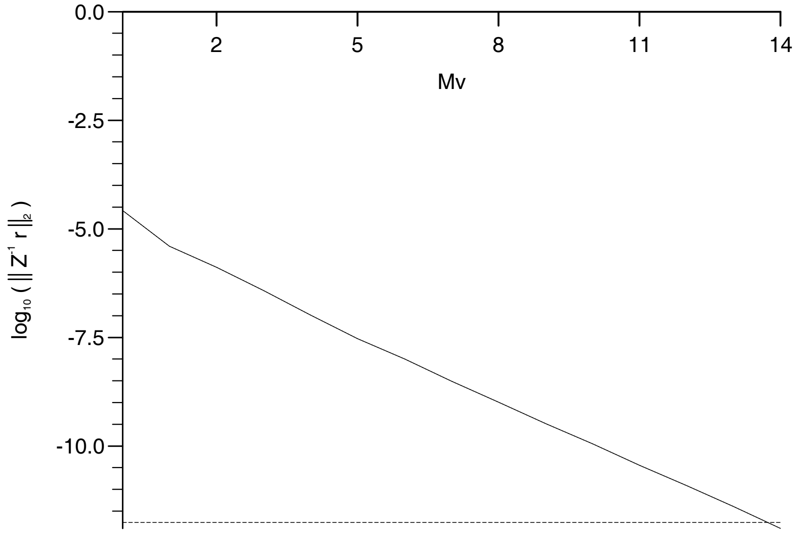


Figure 5.7: Iteration history with CG solver - low tolerance

As with the system from the fluid continuity equation, if the representative matrix is taken at a point later in the time history, convergence is faster (e.g. at the 100th time-step, 12 iterations are needed for convergence) due to the smaller distance from steady state.

If different mesh sizes are used, approximately the same number of iterations are required by the solver (e.g. 14 iterations are required for convergence for the problem on each of the 11×5 , 21×11 and 41×21 meshes).

In [89], it is shown that

- for any symmetric fully assembled finite element matrix, the eigenvalues of the diagonally preconditioned matrix are bounded by the eigenvalues of the diagonally preconditioned element matrices, and

- it is possible to bound the upper and lower eigenvalues of certain diagonally preconditioned element mass matrices independently of the size of the elements or the mesh irregularity. This result applies to a wide range of commonly used elements, including the linear triangles and bi-linear rectangles used in this thesis.

In [89], these results are used to derive mesh independent bounds on the convergence rate of the diagonally preconditioned conjugate method (via (2.15)) applied to the global finite element mass matrix. This explains the behaviour of the solver described in the previous paragraph.

Due to the way physical properties such as porosity, conductivity and dispersivity are treated as constants on elements, their presence does not affect the eigenvalues of the diagonally preconditioned element matrix. Hence mesh independent convergence for the diagonally preconditioned mass matrix is also expected for systems with variable coefficients.

These element bounds are not as useful for the global finite element stiffness matrix which occurred in the previous section because the element stiffness matrix is always singular. Hence the lower eigenvalue is always zero and there is no bound on the condition number.

5.2.2 High Tolerance Test and Preconditioning

Figure 5.8 shows the iteration history for the same representative Darcy velocity vector equation with the harsher convergence criterion of $\tau = 10^{-15}$. Convergence

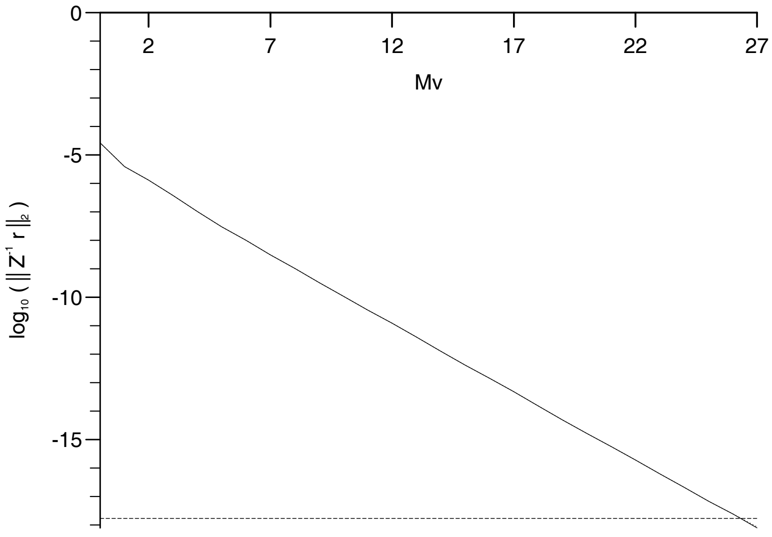


Figure 5.8: Iteration history with CG solver - high tolerance

is still acceptably fast and direct.

As already stated, the diagonal preconditioner gives mesh independent convergence for a system in which the matrix is the finite element mass matrix. Hence a more sophisticated preconditioner is not necessary for this matrix system. However it is possible that a more powerful preconditioner leads to a faster solver.

In order to investigate this possibility, an $ILDL^T$ preconditioner is also used to solve the representative matrix (with the three different meshes). Results which allow a comparison of the speed of the two different preconditioned methods are shown in Table 5.4. The $ILDL^T$ preconditioned CG consistently requires fewer

Mesh	Number of Mvs		Time in solver / seconds	
type	Diagonal	$ILDL^T$	Diagonal	$ILDL^T$
Linear	27	10	0.80	1.12
Quadratic	27	10	0.80	1.14
Cubic	29	10	0.85	1.12

Table 5.4: Performance of diagonal and $ILDL^T$ preconditioners

matrix-vector multiplications to reach convergence than the diagonally preconditioned version. However, due to the extra expense of computing the $ILDL^T$ factorisation and then solving the upper and lower triangular systems at each iteration, the diagonally preconditioned CG method spends less time in the solver. So the diagonal preconditioner is more effective for this system.

5.3 Concluding Remarks

The preconditioned conjugate gradient method is a good solver for the SPD linear systems that arise in the discrete solution of the fluid continuity equation and the Darcy velocity vector equation. It is robust and efficient and requires a relatively small amount of storage. The experiments in Section 5.1.5 indicate that CG more effective than classical iterative methods such as SOR.

The experiments conducted in Sections 5.1.4 and 5.2.2 indicate that the diagonal preconditioner is more effective than the $ILDL^T$ preconditioner for the systems for the Darcy velocity vector components, while the $ILDL^T$ preconditioner is more effective for the system which has to be solved for the pressure.

Chapter 6

Performance of Non-symmetric Solvers

In this chapter, the performance of some non-symmetric linear solvers (described in Section 2.3.2) on the system (3.23) are examined. As in the previous chapter, this is done by numerical experiment. The test cases are generated by the problems which were used to examine discretisations in Chapter 4.

In Chapter 5, which dealt with the performance of solvers for the linear systems with SPD matrices in the model, only CG was used (apart from the brief use of SOR for comparison purposes). Preconditioned CG is generally accepted as the most effective solution technique for large sparse SPD systems. When the class of matrices in question is large, sparse and non-symmetric, there appears to be no such obvious best (at the moment). Due to the success of CG in the SPD case, much recent research on solvers for large sparse non-symmetric systems has concentrated on Krylov subspace methods (for example the methods described in Section 2.3.2).

There have been many comparative studies on non-symmetric Krylov subspace methods in the literature. In [55], each of a selection of these methods is shown to have the best performance on some carefully constructed examples, and the worst on others. More empirically based studies, where various methods are compared in practical situations (e.g. plasma turbulence modelling [11], groundwater flow [61], semiconductor device modelling [66]), have also shown that the relative performance of these methods depends on the situation (and the hardware constraints, e.g. fast memory).

Since so many comparisons of the different methods already exist then, in the early part of this chapter, a representative of each of the two main classes of non-symmetric Krylov subspace methods (i.e. one possessing a minimisation property and one based on short term recurrences) are used on the non-symmetric linear system (3.23). The two methods compared are GMRES and Bi-CGSTAB. Comparisons with other methods can then be made by the use of results from the literature.

6.1 Comparison of Bi-CGSTAB and GMRES

In this section, the performance of GMRES (Algorithm 2.5) and Bi-CGSTAB (Algorithm 2.8) are compared on the linear systems arising in the solution of the discrete contaminant mass balance equation in the 1-D tracer test case from Section 4.1. As in Section 4.1, this 1-D test case is solved on a 2-D mesh. A mesh of linear triangles is used (a single column of rectangles with the same diagonal connected in each).

In all the tests involving the 1-D test case (i.e. the whole of this chapter apart from Section 6.6), the preconditioning matrix, Z, is the diagonal of the system matrix. As in the tests on the SPD solver in the previous chapter, convergence is taken to have occurred when

$$\|Z^{-1}\boldsymbol{r}_i\|_2 \leq \tau\|Z^{-1}\boldsymbol{b}\|_2,$$

and, for these tests, the tolerance is $\tau = 10^{-6}$.

The matrix in this problem is completely characterised by the Courant and Peclet numbers (which were defined in Section 3.4.3). The Courant number (Co) measures the proportion of the advection matrix to the mass matrix, while the Peclet number (Pe) measures the proportion of the advection matrix to the diffusion matrix. In order to examine the effects of varying these parameters, two different mesh sizes are used to give different Peclet numbers, and different (constant) time-steps are used to give different Courant numbers.

The test case is run from the initial condition for 10 time-steps and the average performance is assessed over this period. Table 6.1 shows the average number of matrix-vector multiplications used to reach convergence per time-step for the two solvers. For both solvers, the average number of matrix-vector multiplications increases with increasing Courant number i.e. as the mass matrix becomes less dominant.

			Average Mvs to convergence	
Pe	n	Co	Bi-CGSTAB	GMRES
1	802	0.5	13.2	11.5
		1	20.0	17.2
		5	61.4	54.6
5	162	0.5	10.2	8.3
		1	18.0	13.4
		5	96.4	53.9

Table 6.1: Average Mvs to convergence over first 10 time-steps

Bi-CGSTAB always requires more matrix-vector multiplications to reach convergence, the difference being greatest at high Courant number. This is as expected since GMRES is the optimal Krylov subspace method in terms of minimising the norm of the residual over a given number of iterations. Hence, if the selection of the non-symmetric solver is based purely on the matrix-vector multiplication count, GMRES is the best Krylov subspace method for use on non-symmetric systems. However, as already stated in Section 2.3.2, the amount of work and storage required in GMRES increases with each iteration (whereas the amount of work and storage per iteration remains fixed with Bi-CGSTAB) so the matrix-vector count is not the fairest comparison of the two methods.

In order to allow a fairer comparison to be made between GMRES and Bi-CGSTAB, the average CPU time in the solver per time-step for the set of problems from Table 6.1 is shown in Table 6.2. Due to the greater average work per iteration for GMRES, Bi-CGSTAB is consistently faster for these problems (and it requires a known, relatively small, fixed amount of storage).

Pe	n	Co	Average time in solver / seconds		
			Bi-CGSTAB	GMRES	GMRES(10)
1	802	0.5	0.68	0.76	0.71
		1	1.01	1.26	1.01
		5	3.03	7.13	3.21
5	162	0.5	0.13	0.14	0.14
		1	0.22	0.24	0.22
		5	1.08	1.63	0.80

Table 6.2: Average time in solver over first 10 time-steps

As already stated in Section 2.3.2, the restarted version of GMRES is used to control the storage required by the method and limit the average amount of work per iteration. In order to compare the performance of restarted GMRES and Bi-CGSTAB, Table 6.2 also shows the average time in the solver for GMRES(m) (Algorithm 2.6) with $m = 10$.

GMRES(10) and Bi-CGSTAB require approximately the same time to achieve convergence. GMRES(10) requires an external parameter, namely the restart period. The restart period used here ($m = 10$) is chosen so that the storage required to hold the search vectors is of the same order as the storage required to hold the matrix.) It is possible to get better performance from GMRES(m) by choosing an optimal restart period but this is not investigated here. The need for an external parameter is an undesirable feature of a solver.

Since it takes approximately the same amount of CPU time as GMRES(10) (and does not require an external parameter) then, from these simple tests, Bi-CGSTAB appears to be more promising for the systems that are generated by discretising the contaminant mass balance equation. However, Bi-CGSTAB does not have the stable, monotonic convergence properties of GMRES (or its restarted version). For this reason, there is a need to examine the performance of Bi-CGSTAB under harsher conditions to assess its robustness - this is the subject of the next section.

6.2 Tests on the Robustness of Bi-CGSTAB

In order to test the robustness of Bi-CGSTAB, the same test cases as in Section 6.1 are used, but an extreme tolerance is requested in the convergence criterion. The tolerance requested, τ, is the machine round-off unit.

Definition 6.1 *The* machine round-off unit, *ϵ_M, is the smallest floating point number such that,*

$$fl(x) := x(1 + \epsilon_M)$$

where fl denotes a finite precision operation.

For the machine and compiler used (SUN `f77` on a SPARCstation 1+) the round-off unit is

$$\epsilon_M \approx 2.22045 \times 10^{-16}$$

in double precision.

The condition number of a matrix measures the sensitivity of the solution of the system of linear equations to errors in the data. In finite precision, these errors are caused by machine rounding. A combination of the machine round-off unit and the condition number of the matrix indicate size of the smallest residual norm that can be achieved. Hence, if the matrix is ill-conditioned (i.e. has large condition number), it will be impossible to achieve the tolerance required in these tests (i.e. the machine round-off unit). However, for all the matrices in the robustness tests, the condition number is of $O(10^1)$ (e.g. for the matrix generated with $Co = 0.5$ and $Pe = 1$, the condition number[1] is $\kappa_2(A) = 4.2847$; while for the matrix with $Co = 20$ and $Pe = 5$, $\kappa_2(A) = 37.4391$. Hence it should be possible to achieve, or get close to, the required tolerance.

Some additional tests at larger Courant numbers are also included. Increasing the Courant number increases the contribution of the advection matrix (the source of the non-symmetry) to the overall properties of the matrix in the system.

Due to the harshness of these tests, there is a likelihood that the linear solver will fail to converge. The use of a non-converged solution in a subsequent time-

[1]The condition numbers quoted for these matrices was obtained using the `cond` function in MATLAB.

step would make analysis of the results difficult so, in order to avoid this scenario, the simulation period for the robustness tests is a single time step.

Also, to reduce the effect of initial stiffness in the test case, the tests are started from time $t_0 = 7.5 \times 10^6$ seconds. The initial profile is generated by the analytic solution, i.e. (4.2).

Table 6.3 shows the minimum residual which occurs during the iteration history, and the corresponding matrix-vector multiplication count at which this minimum occurs (Mv). A maximum of $2n$ matrix-vector multiplications are allowed.

Pe	n	Co	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
		0.5	4.66×10^{-15}	32
		1	5.39×10^{-13}	52
		2	1.86×10^{-12}	66
1	802	5	1.60×10^{-11}	126
		10	1.85×10^{-9}	166
		20	3.21×10^{-7}	200
		40	1.14×10^{-4}	236
		0.5	3.74×10^{-14}	30
		1	3.95×10^{-13}	42
		2	4.08×10^{-11}	60
5	162	5	1.12×10^{-7}	118
		10	8.55×10^{-6}	178
		20	2.43×10^{-3}	194
		40	4.13×10^{-2}	228

Table 6.3: Performance of Bi-CGSTAB in robustness tests

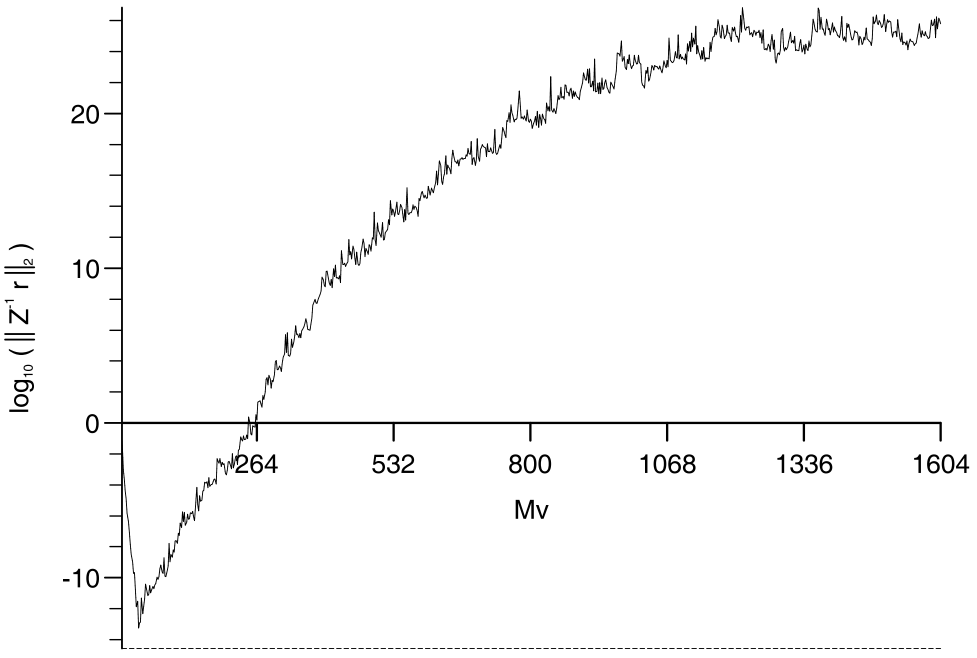
Convergence to the required tolerance is not achieved in any of the test cases. In all but two of the cases, the minimum residual norm achieved is not even within three orders of magnitude of convergence. The condition numbers of the matrices (as already quoted) suggest that a much smaller residual norm can be achieved in most cases.

Figure 6.1 shows the iteration histories for two of the cases in Table 6.3. In both graphs, the decrease in the residual norm is steady and (reasonably) fast in the early stages of the iteration. However, this good early convergence behaviour stops at some stage and the norm of the residual shows a general trend of increasing - this is representative of the behaviour in most of the robustness tests. Note that, as with the iteration histories in the previous chapter, the true residual is also shown in Figure 6.1, as a dotted line (not to be confused with the dashed line which represents the required convergence target). It is indistinguishable from the recurrence based residual.

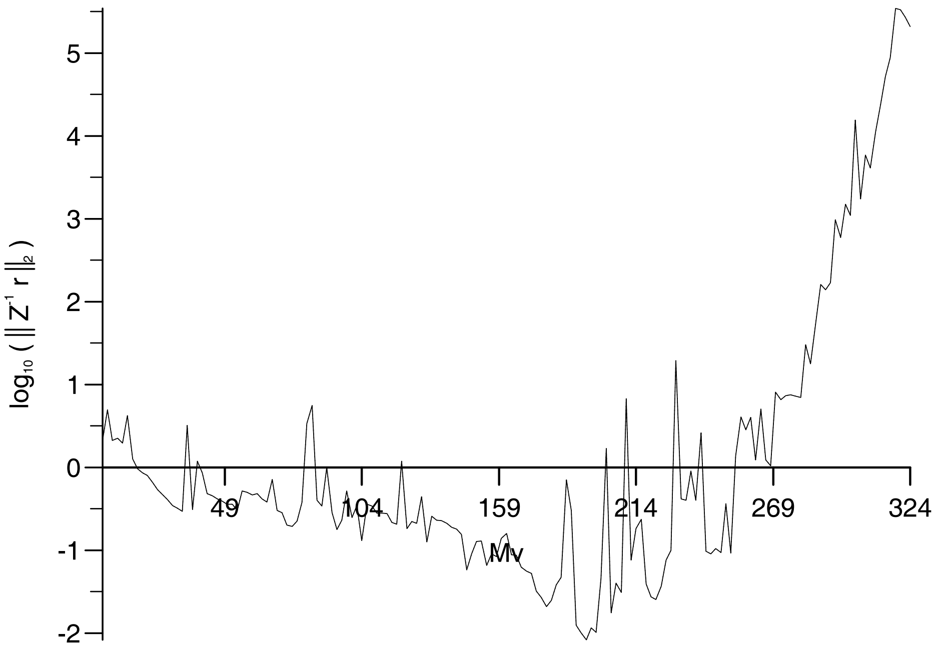
Table 6.4 shows the maximum relative error between the recursion-based preconditioned Bi-CGSTAB residuals, $Z^{-1}\boldsymbol{r}_i$, and the true preconditioned residuals, $Z^{-1}\boldsymbol{r}_i^{true}$ $(= Z^{-1}(\boldsymbol{b}-A\boldsymbol{x}_i)$), and also the matrix-vector multiplication count when this maximum occurs.

Pe	n	Co	$\max\limits_i \frac{\\|Z^{-1}(\boldsymbol{r}_i^{true}-\boldsymbol{r}_i)\\|_2}{\\|Z^{-1}\boldsymbol{r}_i^{true}\\|_2}$	Mv
1	802	0.5	5.30×10^{-2}	32
		1	8.64×10^{-4}	52
		2	4.47×10^{-4}	66
		5	1.55×10^{-4}	126
		10	2.47×10^{-6}	166
		20	2.37×10^{-8}	200
		40	7.66×10^{-11}	236
5	162	0.5	5.28×10^{-3}	30
		1	4.61×10^{-4}	44
		2	1.62×10^{-5}	60
		5	1.16×10^{-8}	118
		10	2.29×10^{-10}	178
		20	1.28×10^{-12}	194
		40	2.03×10^{-13}	290

Table 6.4: Relative error in residuals



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.1: Sample iteration histories for Bi-CGSTAB in robustness tests

Considering the size of the residuals involved, the relative errors are quite small. This indicates that a fatal or near-fatal breakdown (see Section 2.3.2) has not occurred in the underlying nonsymmetric Lanczos process. But the small discrepancy between the true and recursion residuals suggests that the recursion process has been spoiled, leading to the convergence difficulties. Indeed, comparing Tables 6.3 and 6.4, the onset of divergence coincides with the largest relative error in the computed residual (apart from one case).

Hence, the cause of the convergence difficulties must be the rounding errors that occur in finite precision arithmetic. Since Bi-CGSTAB is based on three-term recurrence relations and possesses no quasi-minimisation property, there are no bounds on its rate of the convergence. Because a finite precision phenomenon is occurring in this case, the lack of convergence theory is compounded.

As this point, due to the convergence problems with Bi-CGSTAB, an option is to return to GMRES(m) and accept the need for an external parameter in order to gain the monotone, robust convergence. Apart from the exact arithmetic theory on the convergence of GMRES (see Section 2.3.2), recent work [35] has developed theory for the finite precision behaviour of methods of this type.

Despite all the theory supporting GMRES, due to the promise shown by BiCGSTAB in Table 6.2, the option to use GMRES(m) is not taken here. Instead the convergence difficulties of Bi-CGSTAB are examined in an attempt to overcome them.

Current knowledge of the practical behaviour of Bi-CGSTAB falls into three main categories:

- Investigations on the effects of the presence of extreme (large, small and negative) eigenvalues in the eigenspectrum of the coefficient matrix (e.g. for a comparison of this type with Bi-CG and CG-S, see [12]). These studies tend to use matrices constructed to generate a particular eigenspectrum.

- Comparisons with other solvers on matrices arising from the solution of practical problems (e.g. Peters [61]).

- Examination of performance on a parameterised family of matrices, an example of which can be found in [73] where the parameters in a discretised reaction-diffusion equation are varied to control the eigenspectrum of the matrix, and the effect of asymmetry and dynamic instability (eigenvalues with both positive and negative real parts) on the performance of CG-S and Bi-CGSTAB is examined.

By necessity, due to the lack of convergence theory for Bi-CGSTAB, all these investigations rely on numerical experiment. The tests conducted in this work most closely resemble the last of these categories, the matrices used being parameterised by the Courant and Peclet numbers.

In the following section, techniques for improving the finite precision behaviour of Krylov subspace methods are investigated by numerical experiment on the set of problems used in the robustness test cases in this section.

6.3 Improving the Robustness of Bi-CGSTAB

In this section, techniques for improving the convergence behaviour of Krylov subspace methods are examined in an attempt to prevent the divergent behaviour in the robustness tests highlighted in Figure 6.1. Some techniques for improving convergence behaviour of Krylov subspace methods currently in the literature are:

- **residual smoothing** [90] - an auxiliary sequence of vectors, $\bar{\boldsymbol{x}}_i$, is generated from non-monotonic iterates, $\boldsymbol{x}_i$, by the recursion

$$\begin{aligned} \bar{\boldsymbol{x}}_0 &= \boldsymbol{x}_0 \\ \bar{\boldsymbol{x}}_i &= (1-\eta_i)\bar{\boldsymbol{x}}_{i-1} + \eta_i \boldsymbol{x}_i \quad (i = 1, 2, \ldots), \end{aligned}$$

  where each η_i is chosen to minimise

$$\|\boldsymbol{b} - A\left\{(1-\eta)\bar{\boldsymbol{x}}_{i-1} + \eta \boldsymbol{x}_i\right\}\|_2,$$

  over $\eta \in \mathbb{R}$. The parameter η_i is given explicitly by

$$\eta_i = -\frac{\boldsymbol{s}_{i-1}^T(\boldsymbol{r}_i - \boldsymbol{s}_{i-1})}{\|\boldsymbol{r}_i - \boldsymbol{s}_{i-1}\|_2^2},$$

  where $\boldsymbol{s}_{i-1} = \boldsymbol{b} - A\bar{\boldsymbol{x}}_{i-1}$. The vectors in the auxiliary sequence, $\bar{\boldsymbol{x}}_i$, are iterates with monotone non-increasing residual.

- **random initial iterate** - the Mismatch Theorem [80] indicates that a breakdown in the Lanczos process can be caused by "irregular" left- and right-eigenvector distributions in $\boldsymbol{r}_0$.

  Joubert [46] suggests that this problem can be overcome by using an appropriately scaled initial vector consisting of random entries. Since good initial iterates are often available in time-dependent and non-linear problems, it is more suitable in these cases to add a perturbation to the initial vector.

- **restarting** - if the recursion process is spoiled by rounding errors then, by restarting the iteration with a new initial iterate (e.g. the latest one), the current numerical Krylov subspace is discarded and with it all the rounding errors thus far. This method is advocated (although for other reasons) for Bi-CGSTAB in [84].

- **look-ahead Lanczos** - as already described in Section 2.3.2, the non-symmetric Lanczos process (which underpins the three-term recurrence relation for the methods described in Section 2.3.2) can suffer fatal or near-fatal breakdown. To remedy this, the look-ahead Lanczos process [27, 60] allows the use of block pivots in the iteration steps where the scalar pivots of the standard Lanczos process are expected to encounter difficulties. This approach is used in practical versions of the QMR method described in Chapter 2.

- **partial orthogonalisation** [13]. In the Lanczos process, after an eigenvector is accurately determined, loss of orthogonality (or bi-orthogonality in the non-symmetric case) due to rounding errors causes the creation of copies of the same eigenvector. These repetitions in the underlying Lanczos process slow the convergence of the iterative method. In order to overcome this problem, it is possible to explicitly impose full orthogonality by performing a Gram-Schmidt orthogonalisation on the vectors of the Krylov subspace as they are produced, e.g. GMRES. However this is expensive in terms of both computing time and storage, particularly if a large number of iterations are required.

In [13], a partial orthogonalisation approach is introduced and investigated for CG and Bi-CG. In this method, a new vector $\boldsymbol{r}_k$ is orthogonalised with respect to the previous normalised vectors $\boldsymbol{r}_j/\|\boldsymbol{r}_j\|_2$ $(j < k)$ of the base and added to this base until a given iteration. After this, each new vector is only orthogonalised with respect to the base without increasing its dimension. Note that this requires that, once constructed, the base is kept until the process is converged.

- variants of Bi-CGSTAB (e.g. Bi-CGSTAB2 [36], Bi-CGSTAB(ℓ) [74]) allow **quadratic polynomials** in the construction of $\tilde{\varphi}_i(A)$ in (2.19) rather than the linear components, $(1-\omega_j A)$, used in the original van der Vorst version. These methods attempt to avoid stagnation in the Bi-CGSTAB iteration history which occurs when the eigenvalues of the matrix are almost purely imaginary.

Look-ahead Lanczos is not investigated in this thesis as the convergence problems are not caused by fatal or near-fatal breakdown. The partial orthogonalisation looks promising and should be investigated for systems which suffer from the problems highlighted in this thesis, but this avenue of research has not been pursued here. Conversely, the random initial iterate approach is not expected to improve the convergence behaviour, but results are given for this approach for completeness.

6.3.1 Residual Smoothing

Rather than the residual smoothing algorithm already described, the following one (from [95]), which is the same in exact arithmetic but has better numerical rounding properties, is used in this section.

Algorithm 6.1 RESIDUAL SMOOTHING

This algorithm performs single parameter residual smoothing on the iterates produced by the Bi-CGSTAB method of Algorithm 2.8 to produce a sequence of iterates with monotone (non-increasing) residual norm.

```
$\bar{\boldsymbol{r}}_0 := \boldsymbol{r}_0$
```

```
$\bar{\boldsymbol{x}}_0 := \boldsymbol{x}_0$
$\Delta \boldsymbol{r} := \boldsymbol{0}$
$\Delta \boldsymbol{x} := \boldsymbol{0}$
for $i = 1, 2, \ldots,$
  Generate $\alpha$, $\omega$, $\boldsymbol{t}$, $\boldsymbol{v}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ by a Bi-CGSTAB iteration (Algorithm 2.8)
  $\Delta \boldsymbol{r} := \Delta \boldsymbol{r} + \alpha \boldsymbol{v} + \omega \boldsymbol{t}$
  $\Delta \boldsymbol{x} := \Delta \boldsymbol{x} + \alpha \boldsymbol{y} + \omega \boldsymbol{z}$
  $\eta := \dfrac{(\bar{\boldsymbol{r}}_{i-1}, \Delta \boldsymbol{r})}{(\Delta \boldsymbol{r}, \Delta \boldsymbol{r})}$
  $\bar{\boldsymbol{r}}_i := \bar{\boldsymbol{r}}_{i-1} - \eta \Delta \boldsymbol{r}$
  $\bar{\boldsymbol{x}}_i := \bar{\boldsymbol{x}}_{i-1} + \eta \Delta \boldsymbol{x}$
  $\Delta \boldsymbol{x} := (1 - \eta) \Delta \boldsymbol{x}$
  $\Delta \boldsymbol{r} := (1 - \eta) \Delta \boldsymbol{r}$
```

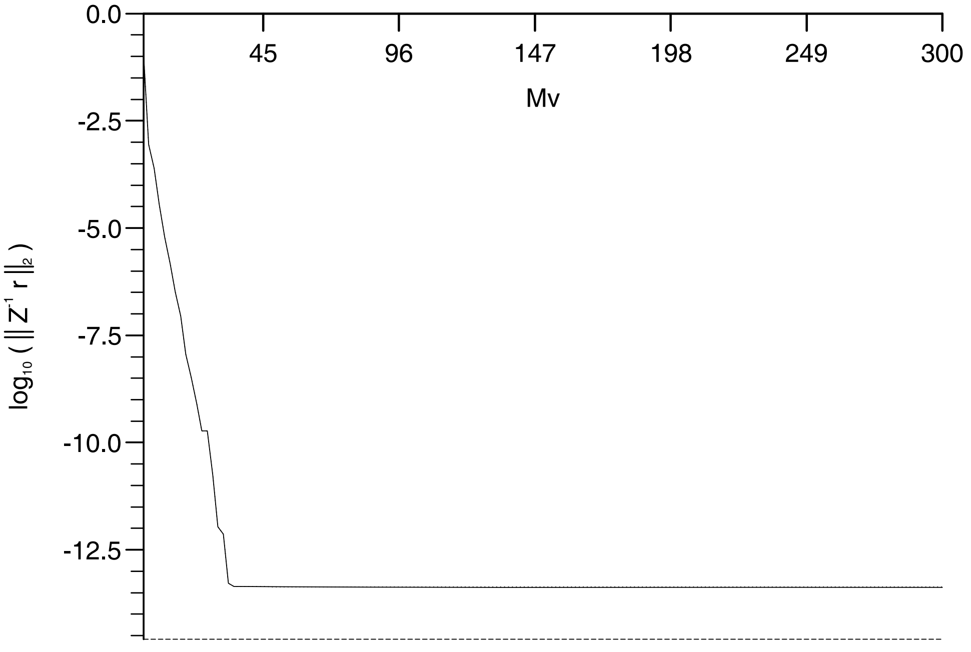
Table 6.5 is the residual smoothed equivalent of Table 6.3, showing the minimum preconditioned residual achieved.

Comparing the two tables, the residual smoothing produces residuals that are smaller (in the pre-conditioned 2-norm sense) than the Bi-CGSTAB residuals from which they are generated. However, in general, the improvement is not enough to warrant even the small amount of extra work involved.

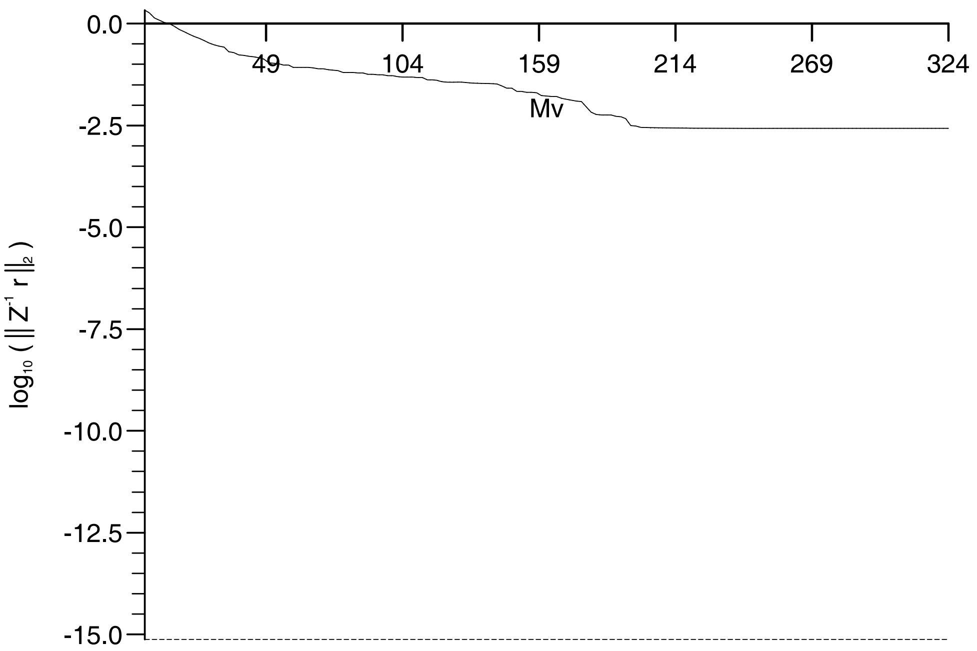
The residual smoothing does not improve convergence significantly because the original Bi-CGSTAB iterates on which the smoothing is based are still prone to the same problems as before; hence the residual smoothing algorithm stagnates (i.e. $\eta = 0$) when Bi-CGSTAB stops converging.

Figure 6.2 is the residual smoothed equivalent of Figure 6.1. An obvious advantage of the process is that the solution is monotone non-increasing, but the effect is only marginally better than the cheaper method of storing the iterate corresponding to the smallest residual norm thus far.

In [95], it is shown that the QMR method (described on page 38 of this thesis) is equivalent to a residual smoothing process applied to Bi-CG iterates. From [16, 28], using the notation already introduced for QMR, the quantity that is minimised in the QMR algorithm at the i^{th} iteration,



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.2: Sample iteration histories for residual smoothing in robustness tests

Pe	n	Co	$\min_i \frac{\|Z^{-1}\bar{\boldsymbol{r}}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$
1	802	0.5	3.59×10^{-15}
		1	3.08×10^{-13}
		2	1.09×10^{-12}
		5	4.44×10^{-12}
		10	1.47×10^{-9}
		20	2.05×10^{-7}
		40	4.66×10^{-5}
5	162	0.5	8.89×10^{-15}
		1	1.33×10^{-13}
		2	1.45×10^{-11}
		5	2.76×10^{-8}
		10	2.93×10^{-6}
		20	7.86×10^{-4}
		40	1.54×10^{-2}

Table 6.5: Robust test performance with residual smoothing

$$\boldsymbol{z}_i^{\mathrm{QMR}} = \tilde{\beta}_1 \boldsymbol{e}_1^i - H_{i+1}\boldsymbol{y}_i$$

is related to the Bi-CG residual at the i^{th} iteration, $\boldsymbol{r}_i^{\text{Bi-CG}}$, by

$$\|\boldsymbol{r}_i^{\text{Bi-CG}}\|_2 = \frac{\|\boldsymbol{z}_i^{\mathrm{QMR}}\|_2}{\sqrt{1 - \left(\|\boldsymbol{z}_i^{\mathrm{QMR}}\|_2 / \|\boldsymbol{z}_{i-1}^{\mathrm{QMR}}\|_2\right)^2}}$$

From this relationship, it can be seen that fast convergence of QMR corresponds to fast convergence of Bi-CG, and slow convergence of QMR corresponds to Bi-CG spikes (i.e. QMR stagnation corresponds to Bi-CG breakdown).

6.3.2 Random Initial Iterate

As the convergence difficulties are not caused by breakdown, it is not anticipated that the technique of using a random initial iterate will improve the convergence behaviour of Bi-CGSTAB. These tests are only included for completeness.

Since a good initial iterate, $\boldsymbol{x}_0$, is available in this problem, a vector of random entries ($\boldsymbol{x}_R$) is added to perturb it (to disrupt irregular left and right eigenvector distributions) and generate a new initial iterate $\tilde{\boldsymbol{x}}_0$. The size of the random perturbation is controlled by scaling and a factor, χ, i.e.

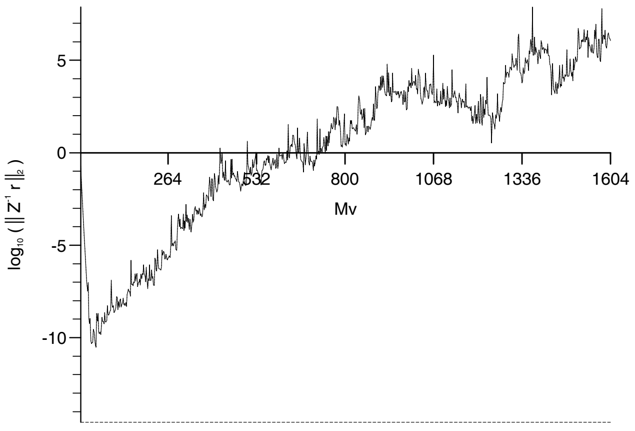
$$\tilde{\boldsymbol{x}}_0 = \boldsymbol{x}_0 + \chi \|\boldsymbol{x}_0\|_2 \frac{\boldsymbol{x}_R}{\|\boldsymbol{x}_R\|_2} \quad \text{with} \quad \boldsymbol{x}_R = \{x_i\}_{i=1,\dots,n} \quad \text{and} \quad -1 \leq x_i \leq 1.$$

Table 6.6 shows the minimum residual achieved and the corresponding matrix-vector multiplication count for the robustness tests with different perturbation sizes.

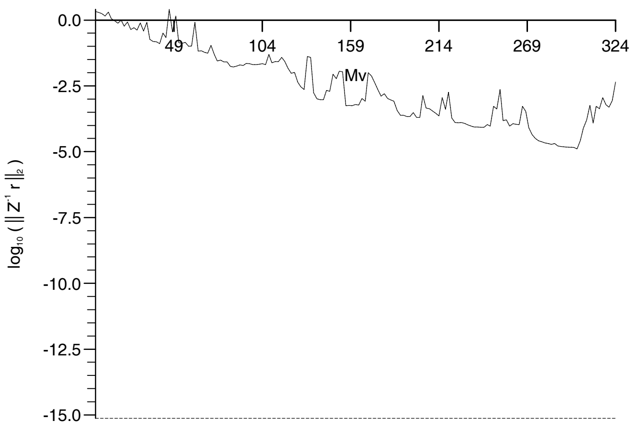
Pe	Co	$\chi = 10^{-3}$		$\chi = 10^{-2}$		$\chi = 10^{-1}$	
		$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
1	0.5	2.59×10^{-12}	46	5.03×10^{-11}	34	2.03×10^{-9}	26
	1	1.10×10^{-11}	40	5.47×10^{-10}	38	1.97×10^{-10}	44
	2	4.05×10^{-11}	72	9.68×10^{-11}	120	2.09×10^{-9}	122
	5	2.04×10^{-9}	128	7.52×10^{-9}	144	1.61×10^{-7}	146
	10	1.14×10^{-9}	230	4.21×10^{-8}	230	3.32×10^{-8}	230
	20	1.92×10^{-9}	404	1.81×10^{-8}	360	2.82×10^{-7}	342
	40	5.55×10^{-8}	474	4.63×10^{-7}	442	1.34×10^{-6}	488
5	0.5	8.34×10^{-13}	42	9.19×10^{-11}	34	1.65×10^{-10}	34
	1	2.83×10^{-11}	40	6.20×10^{-10}	32	3.19×10^{-9}	36
	2	6.73×10^{-10}	64	8.08×10^{-9}	50	1.03×10^{-7}	60
	5	2.74×10^{-8}	120	1.51×10^{-7}	146	3.95×10^{-7}	126
	10	4.43×10^{-7}	196	7.23×10^{-7}	204	1.31×10^{-6}	222
	20	3.75×10^{-6}	300	9.73×10^{-6}	300	2.06×10^{-5}	290
	40	8.86×10^{-5}	322	1.23×10^{-4}	312	3.36×10^{-4}	324

Table 6.6: Bi-CGSTAB robustness test performance with random initial iterate

If the results in Table 6.6 are compared with those in Table 6.3, it can be seen that applying a random perturbation to the initial vector improves the robustness of the Bi-CGSTAB iteration (in the sense that the minimum residual is closer to the robustness test convergence criterion) for high Courant numbers, but it has a detrimental effect for low Courant numbers. The most effective perturbation (in terms of achieving the lowest minimum residual) is the smallest, $\chi = 10^{-3}$ (i.e. 0.1 %).



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.3: Sample iteration histories for random initial iterate ($\chi = 10^{-3}$)

Figure 6.3 is the equivalent of Figure 6.1 with a random perturbation (with $\chi = 10^{-3}$) to the initial iterate. The underlying trends in the convergence histories in both these figures are the same. In general, as expected, this technique does not overcome the convergence difficulties in the robustness tests.

6.3.3 Restarting

The original Bi-CGSTAB paper [84] recommends that practical implementations of the method should monitor sensitive values and, if any of these become too small, the iteration should be stopped and Bi-CGSTAB should be restarted with a new choice of initial iterate. The suggested sensitive values are the scalars ρ_{i+1} and $(\hat{\boldsymbol{r}}_0, \boldsymbol{v})$ (see Algorithm 2.8). These values are highlighted because they can be close to zero without convergence having taken place, and are used as denominators in the algorithm (so are particularly sensitive to round-off error). If these sensitive values become equal to zero, they cause a division by zero which leads to a total breakdown in the algorithm - this is known as a *fatal breakdown*.

The logical choice of new initial iterate is either the last iterate produced before the restart, or the iterate corresponding to the smallest residual achieved thus far.

Note that, in practice, a trap must be included in the restart algorithm to ensure that, after a restart has been performed, another one is not attempted immediately as this would lead to a fixed cycle of producing the same iterate repeatedly.

Due to the cost of re-calculating the initial residual and initialising vectors, each restart requires approximately half the computational expense of a Bi-CGSTAB iteration. Restarting is often used in the literature (e.g. [45, 46, 84]) but the criteria tend to be heuristic and of the form,

$$\text{Restart if monitor } < \text{ tolerance.}$$

In most cases, no guidance is given on the origin of the monitor or the sensitivity of the criterion to the tolerance.

If the restart criterion is not severe enough, then restart is premature and good convergence behaviour can be interrupted and spoiled. This is similar to the problem that can arise in the practical implementation of GMRES (GMRES(m)) which, in order to decrease the storage requirements and average work per iteration, restarts after every m iterations. If the restart criterion is too lenient, rounding errors can build up and cause the divergent behaviour demonstrated in Figure 6.1. In this section, various forms of restart criteria are examined.

Restart monitors based on sensitive values

In the first criteria examined, the two sensitive values recommended in [84] are used as the monitors, and the round-off unit definition is the basis of the tolerance.

The operation in the Bi-CGSTAB algorithm (Algorithm 2.8) where ρ is used as a denominator is

$$\boldsymbol{p} := \boldsymbol{r}_{i-1} + \frac{\rho_i}{\rho_{i-1}} \frac{\alpha}{\omega} (\boldsymbol{p} - \omega \boldsymbol{v}).$$

From Definition 6.1, serious rounding errors are expected to occur in this particular operation if,

$$\left.\begin{array}{c} \left| \beta \dfrac{\{\boldsymbol{p}(j) - \omega \boldsymbol{v}(j)\}}{\boldsymbol{r}_{i-1}(j)} \right| < \epsilon_M \\ \text{or} \\ \left| \beta \dfrac{\{\boldsymbol{p}(j) - \omega \boldsymbol{v}(j)\}}{\boldsymbol{r}_{i-1}(j)} \right| > \dfrac{1}{\epsilon_M} \end{array} \right\} \quad (1 \leq j \leq n), \tag{6.1}$$

where $\beta = (\rho_i/\rho_{i-1})(\alpha/\omega)$ and $\boldsymbol{v}(j)$ is the j^{th} component of $\boldsymbol{v}$. The expression is undefined when an entry in the residual vector is zero so these cases must be excluded. Hence a more suitable form of (6.1) is

$$\boldsymbol{r}_{i-1}(j) \neq 0 \quad \text{and} \quad \left\{ \begin{array}{c} |\beta\{\boldsymbol{p}(j) - \omega \boldsymbol{v}(j)\}| < \epsilon_M |\boldsymbol{r}_{i-1}(j)| \\ \text{or} \\ |\beta\{\boldsymbol{p}(j) - \omega \boldsymbol{v}(j)\}| > \dfrac{1}{\epsilon_M} |\boldsymbol{r}_{i-1}(j)| \end{array} \right\} \quad (1 \leq j \leq n). \tag{6.2}$$

This restart criterion requires no external parameters other than the easily available round-off unit. However, experimental investigations show that in all the robustness tests, the monitor value is always well within the bounds in criterion (6.2). This is not surprising since an event such as (6.2) indicates a fatal or near-fatal breakdown, while the results from the previous robustness tests indicate that the convergence difficulties are caused by a gradual corruption rather than a total breakdown.

By performing numerical experiments and observing the monitor value over the set of robustness tests, it was noticed that the breakdown in convergence coincides with the monitor growing larger than 10^4 in many of the cases. This suggests that, in practice, the restart criterion (6.2) should be modified to

$$\left.\begin{array}{c} \boldsymbol{r}_{i-1}(j) \neq 0 \\ \\ \text{and} \\ \\ |\beta\{\boldsymbol{p}(j) - \omega \boldsymbol{v}(j)\}| > \dfrac{1}{10^{-4}}\, |\boldsymbol{r}_{i-1}(j)| \end{array}\right\} \quad (1 \leq j \leq n). \tag{6.3}$$

Table 6.7 shows the performance of Bi-CGSTAB restarted by criterion (6.3) for the robustness tests.

Pe	n	Co	$\min_i \dfrac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
		0.5	$< \epsilon_M$	42
		1	$< \epsilon_M$	70
		2	$< \epsilon_M$	118
1	802	5	$< \epsilon_M$	294
		10	$< \epsilon_M$	600
		20	$< \epsilon_M$	2326
		40	$< \epsilon_M$	6364
		0.5	$< \epsilon_M$	42
		1	$< \epsilon_M$	88
		2	$< \epsilon_M$	150
5	162	5	$< \epsilon_M$	310
		10	$< \epsilon_M$	578
		20	$< \epsilon_M$	978
		40	$< \epsilon_M$	1424

Table 6.7: Performance of Bi-CGSTAB restarted by criterion (6.3)

Convergence is achieved in all the cases. Hence it appears to be possible to monitor and control the gradual build up of the rounding errors with (6.3). However, the tolerance (10^4) is an external parameter so the overall restart criterion is not ideal.

Figure 6.4 shows the iteration histories for two of the cases in the table. On the graphs in this figure, the symbol ◯ denotes a restart due to criterion (6.3).

In the same way as the round-off unit definition is used to generate the possible restart criterion (6.1), the operation

$$\boldsymbol{s} := \boldsymbol{r}_{i-1} - \frac{\rho_i}{(\hat{\boldsymbol{r}}_0^T \boldsymbol{v})}\boldsymbol{v}$$

gives rise to

$$\boldsymbol{r}_{i-1}(j) \neq 0 \quad \text{and} \quad \left\{ \begin{array}{c} |\alpha \boldsymbol{v}(j)| < \epsilon_M |\boldsymbol{r}_{i-1}(j)| \\ \text{or} \\ |\alpha \boldsymbol{v}(j)| > \dfrac{1}{\epsilon_M}|\boldsymbol{r}_{i-1}(j)| \end{array} \right\} \quad (1 \leq j \leq n), \tag{6.4}$$

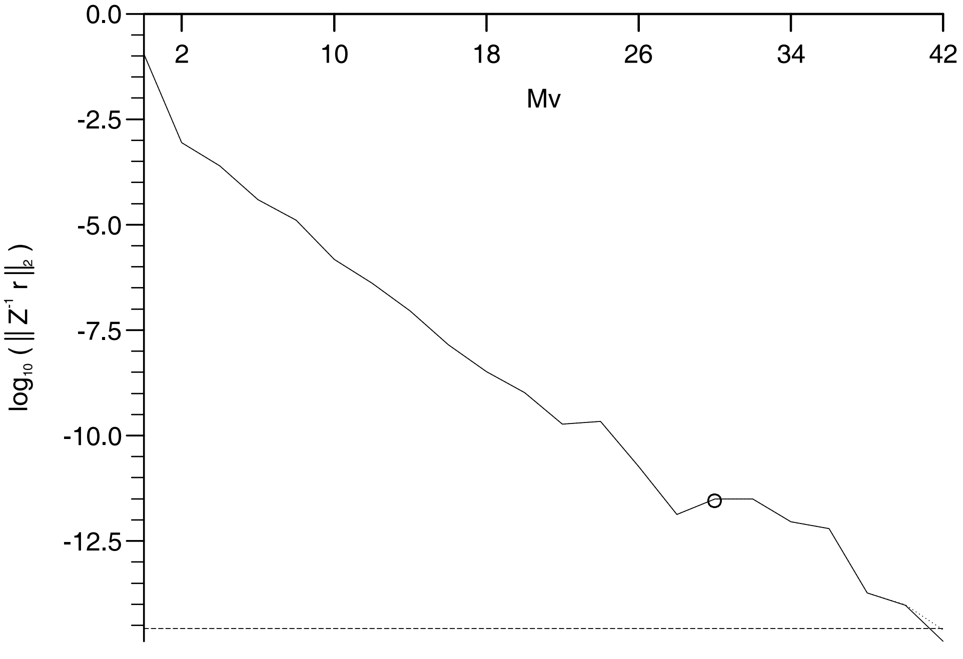
where $\alpha = \rho_i/(\hat{\boldsymbol{r}}_0^T \boldsymbol{v})$. Again (as expected since fatal or near-fatal breakdown is not the difficulty in these problems) experimental investigations show that this monitor is always well within the bounds for all the tests, and there are no significant features in the monitor near the onset of convergence difficulties, so this restart criterion is also not effective.

As before, by tracking the monitor value in numerical experiments, restarting based on the criterion,

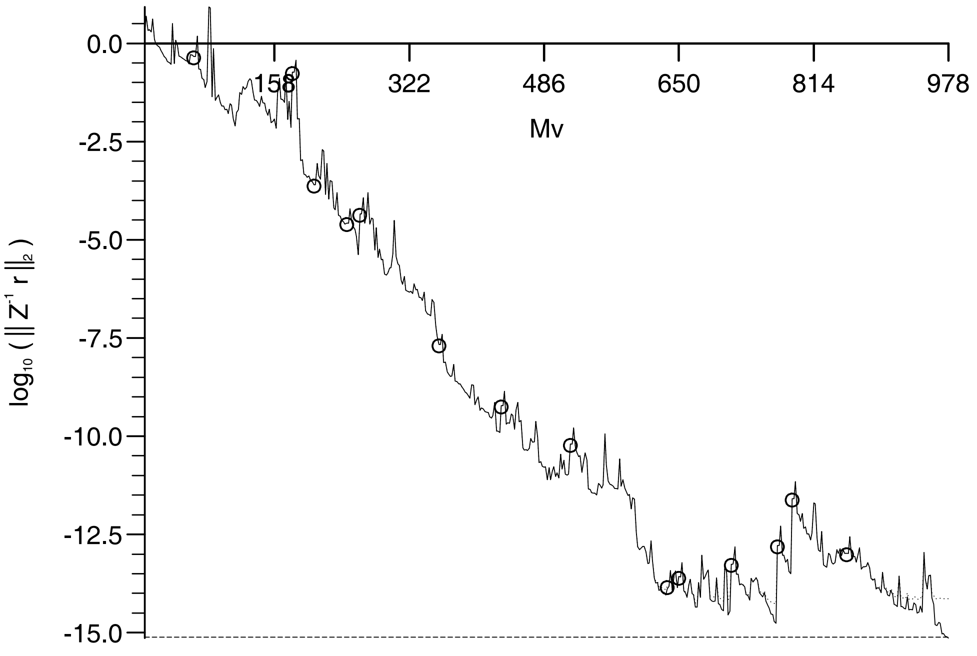
$$\left. \begin{array}{c} \boldsymbol{r}_{i-1}(j) \neq 0 \\ \text{and} \\ |\alpha \boldsymbol{v}(j)| > \dfrac{1}{5 \times 10^{-5}}|\boldsymbol{r}_{i-1}(j)| \end{array} \right\} \quad (1 \leq j \leq n), \tag{6.5}$$

in which the tolerance is again an external parameter, is found to have some success in adding to the robustness of the method. Table 6.8 shows the performance of Bi-CGSTAB restarted by criterion (6.5) for the robustness tests.

Comparing the results in Table 6.8 with those in Table 6.7, in most cases the criterion (6.5) is more effective than the restart criterion (6.3) as it leads to



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.4: Sample iteration histories of Bi-CGSTAB restarted by criterion (6.3)

Pe	n	Co	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
1	802	0.5	$< \epsilon_M$	44
		1	$< \epsilon_M$	116
		2	$< \epsilon_M$	98
		5	$< \epsilon_M$	192
		10	2.77×10^{-7} (fixed cycle)	116
		20	$< \epsilon_M$	660
		40	7.35×10^{-9} (fixed cycle)	534
5	162	0.5	$< \epsilon_M$	42
		1	$< \epsilon_M$	76
		2	$< \epsilon_M$	148
		5	$< \epsilon_M$	246
		10	$< \epsilon_M$	458
		20	$< \epsilon_M$	622
		40	$< \epsilon_M$	946

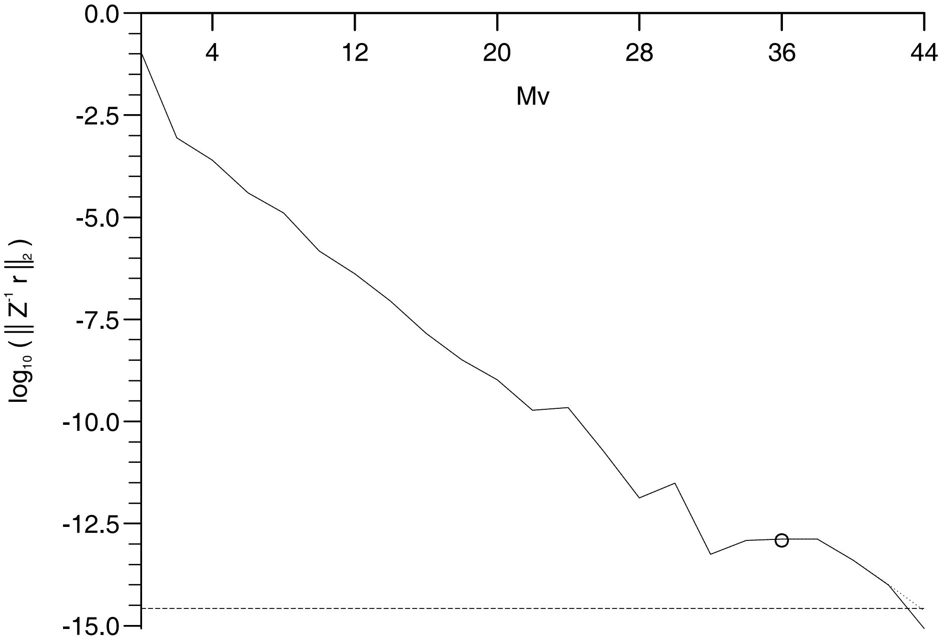
Table 6.8: Performance of Bi-CGSTAB restarted by criterion (6.5)

convergence in fewer matrix-vector multiplications. However, convergence is not achieved in two of the cases due to "fixed cycles", i.e. a restart is attempted on the iteration immediately following a restart.

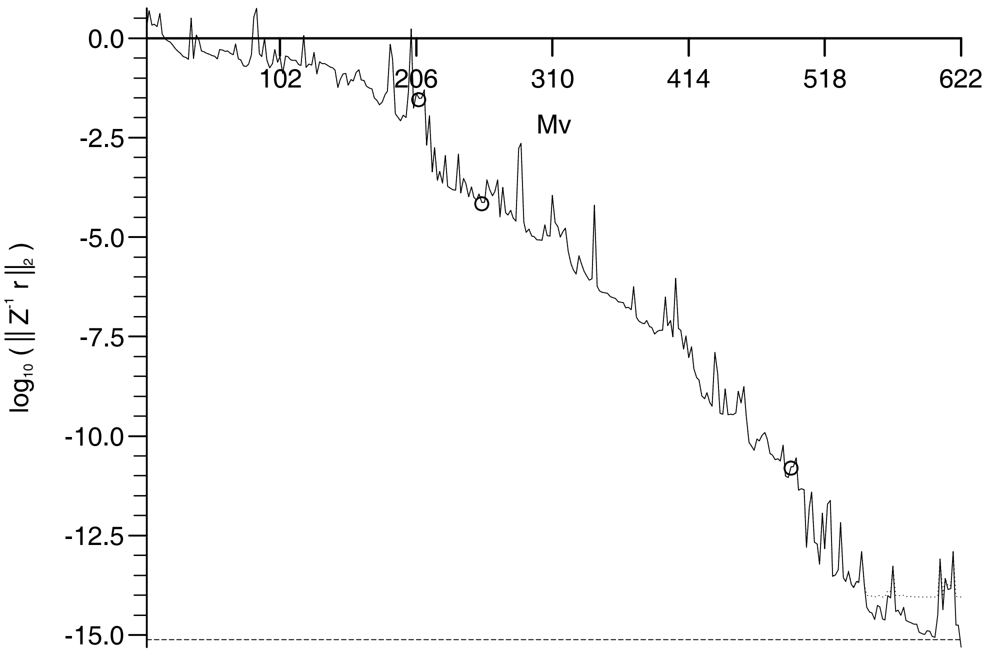
The graphs in Figure 6.5 show the iteration histories of two of the cases in Table 6.8 - again, ○ denotes a restart.

It is possible to use restart tests based on both criterion (6.3) and (6.5) at the same time. However, as has already been seen, it is not the breakdown of either monitor value that is tested for in the effective restart criteria; rather the monitor values are being used to monitor the build up of rounding error in the whole process. Hence there is little advantage to be gained by combining these restart criteria as each one performs precisely the same task.

In summary, restarting using monitors based on sensitive values can be used to improve the robustness of Bi-CGSTAB in these cases, but it is not an ideal



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.5: Sample iteration histories of Bi-CGSTAB restarted by criterion (6.5)

solution as fixed cycles occur and there is a requirement for an external parameter.

Restart monitors based on inner product denominators

In [45], restart criteria based on the occurrence of inner products as denominators in Bi-CG and CG-S are used. The general form of these criteria are that if $(\boldsymbol{a}, A\boldsymbol{b})$ is used as a denominator in the algorithm then restart if

$$\frac{|(\boldsymbol{a}, A\boldsymbol{b})|}{\|\boldsymbol{a}\|_2 \|A\boldsymbol{b}\|_2} \leq f(\epsilon_M).$$

In [45], the tolerance has the form, $f(\epsilon_M) = 10^a \epsilon_M^{\frac{1}{2}}$, where a is an integer.

Since the inner product $(\boldsymbol{r}_0, \boldsymbol{v})$ is used as a denominator in Bi-CGSTAB in the calculation of α (see Algorithm 2.8) then a possible restart criterion is

$$\frac{|(\hat{\boldsymbol{r}}_0, \boldsymbol{v})|}{\|\hat{\boldsymbol{r}}_0\|_2 \|\boldsymbol{v}\|_2} \leq \tau_1. \tag{6.6}$$

A Joubert inner product test can also be used as a possible restart criterion for the sensitive value $(\hat{\boldsymbol{r}}_0, \boldsymbol{t})$ which is generated during Algorithm 2.8. The form of this test is

$$\frac{|(\hat{\boldsymbol{r}}_0, \boldsymbol{t})|}{\|\hat{\boldsymbol{r}}_0\|_2 \|\boldsymbol{t}\|_2} \leq \tau_2. \tag{6.7}$$

By experiment, the monitors in (6.6) and (6.7) are found to become very small near the on-set of divergence. This indicates that these monitors can be used to signal when to restart in these problems.

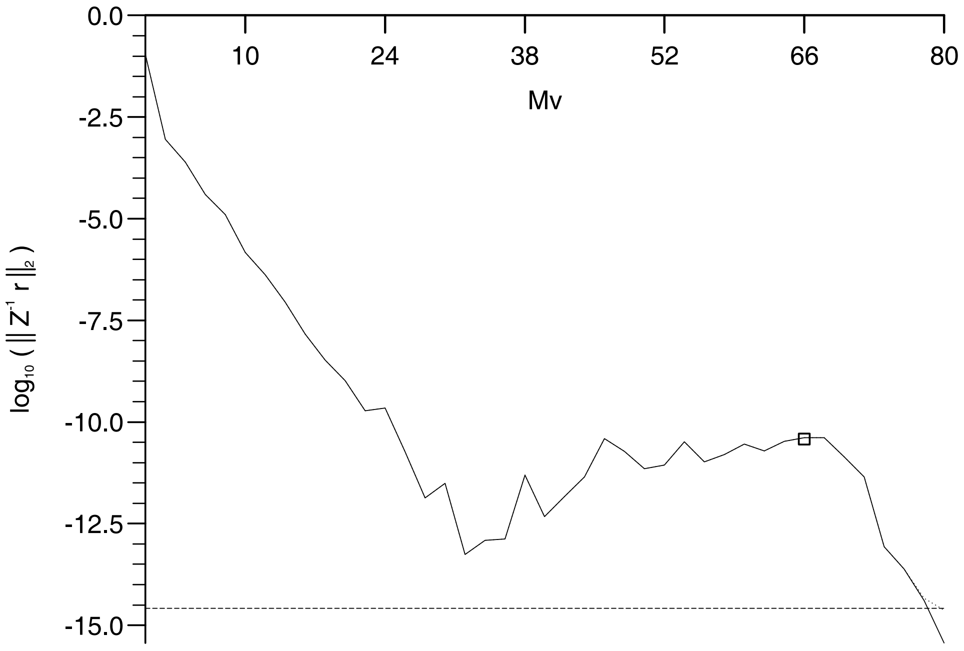
Table 6.9 shows the robustness test performance with (6.6) and (6.7) used as restart criteria and $\tau_1 = \tau_2 = 10^{-8}$ - the selection of these restart tolerances being based on experience gained by conducting numerical experiments. The required convergence tolerance is achieved in all cases.

As with the requirement for the user defined restart tolerances in the previous section, the requirement for the parameters τ_1 and τ_2 violates the desired property of no external parameters.

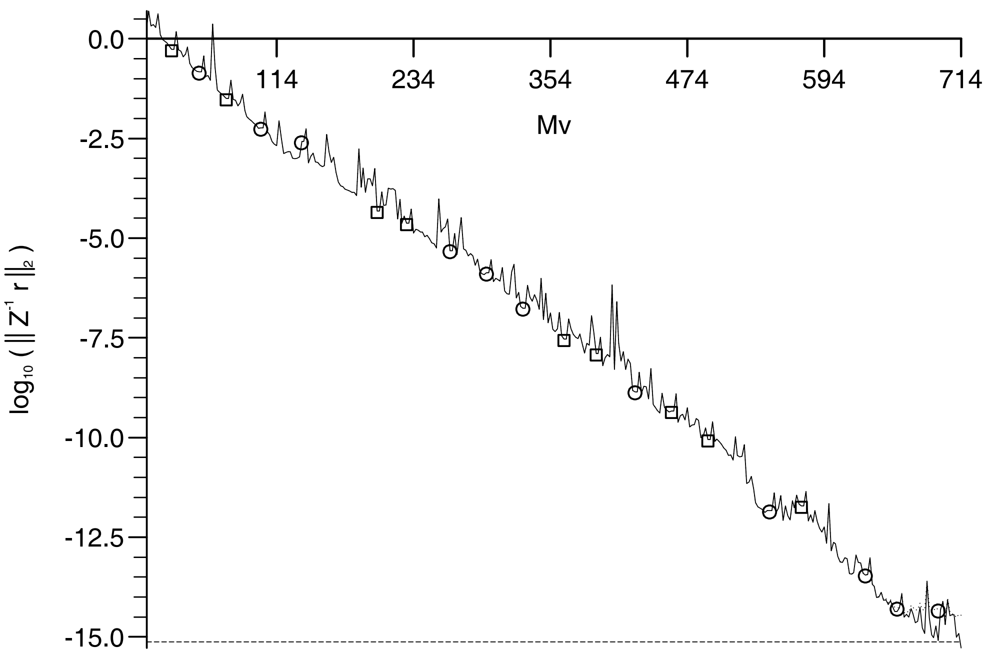
Pe	n	Co	$\min_i \frac{\lVert Z^{-1}\boldsymbol{r}_i\rVert_2}{\lVert Z^{-1}\boldsymbol{b}\rVert_2}$	Mv
1	802	0.5	$< \epsilon_M$	80
		1	$< \epsilon_M$	60
		2	$< \epsilon_M$	100
		5	$< \epsilon_M$	200
		10	$< \epsilon_M$	370
		20	$< \epsilon_M$	666
		40	$< \epsilon_M$	1102
5	162	0.5	$< \epsilon_M$	44
		1	$< \epsilon_M$	68
		2	$< \epsilon_M$	100
		5	$< \epsilon_M$	256
		10	$< \epsilon_M$	456
		20	$< \epsilon_M$	714
		40	$< \epsilon_M$	886

Table 6.9: Performance with inner product criteria based restarts

Figure 6.6 corresponds to Figure 6.1 with restarts based on the criteria (6.6) and (6.7). In these convergence histories, $\bigcirc$ denotes restart due to (6.6) and $\square$ denotes restart due to (6.7).



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.6: Sample iteration histories of Bi-CGSTAB restarted by criteria (6.6) and (6.7)

Fixed Period Restart

It is possible that the restart criterion used in the previous two sections are only effective because the tolerances are tuned to the problems so as to be triggered intermittently to flush out the rounding errors in the recurrence.

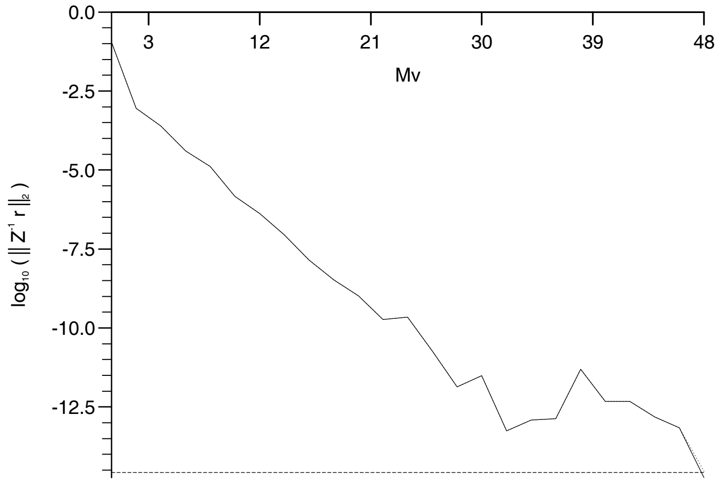
A primitive restart criterion, based on the idea of intermittently discarding the recurrences (and the associated rounding errors) is investigated in this section. In this restart criterion, the process is restarted periodically after a fixed number of iterations. This is similar to GMRES(m) but, where a restart is performed to avoid impractical storage requirements and work per iteration in that case, here it is performed at regular intervals to try to prevent the build-up of rounding errors spoiling the recursion process.

Table 6.10 shows the minimum residual achieved and the corresponding iteration number for the Bi-CGSTAB robustness tests with different fixed restart periods, k. The convergence criterion is met in all but one of the robustness tests.

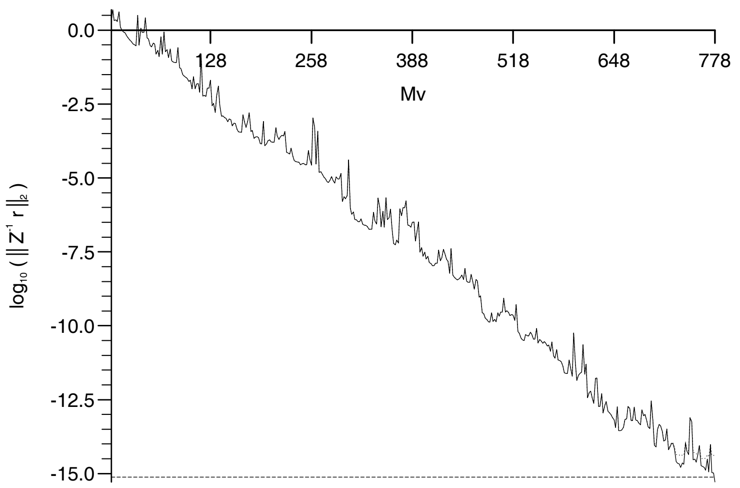
Pe	Co	$k = 5$		$k = 20$		$k = 40$	
		$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
	0.5	$< \epsilon_M$	44	$< \epsilon_M$	48	$< \epsilon_M$	96
	1	$< \epsilon_M$	66	$< \epsilon_M$	60	$< \epsilon_M$	106
	2	$< \epsilon_M$	116	$< \epsilon_M$	96	$< \epsilon_M$	114
1	5	$< \epsilon_M$	320	$< \epsilon_M$	204	$< \epsilon_M$	200
	10	$< \epsilon_M$	752	$< \epsilon_M$	370	$< \epsilon_M$	370
	20	$< \epsilon_M$	1270	$< \epsilon_M$	730	$< \epsilon_M$	734
	40	$< \epsilon_M$	2230	$< \epsilon_M$	1540	$< \epsilon_M$	1204
	0.5	$< \epsilon_M$	44	$< \epsilon_M$	50	$< \epsilon_M$	92
	1	$< \epsilon_M$	68	$< \epsilon_M$	54	$< \epsilon_M$	102
	2	$< \epsilon_M$	160	$< \epsilon_M$	96	$< \epsilon_M$	126
5	5	$< \epsilon_M$	282	$< \epsilon_M$	258	$< \epsilon_M$	238
	10	$< \epsilon_M$	442	$< \epsilon_M$	490	$< \epsilon_M$	470
	20	$< \epsilon_M$	766	$< \epsilon_M$	778	$< \epsilon_M$	732
	40	3.93×10^{-16}	1030	$< \epsilon_M$	1080	$< \epsilon_M$	1038

Table 6.10: Robustness test performance with fixed restart period

Comparing the number of iterations required for convergence in Tables 6.9 and 6.10, the restart strategy based on (6.6) and (6.7) is more effective over the range of cases than a strategy based on any of the fixed restart periods. This is not surprising since the inner product based restart is effectively an adaptive strategy - only restarting when it senses the need for such action - while the fixed period strategy restarts regardless of whether it is required or not.



(a) $Co = 0.5,\ Pe = 1$



(b) $Co = 20,\ Pe = 5$

Figure 6.7: Sample iteration histories for fixed restart period in robustness tests

The value of the restart period k has a significant effect on the number of iterations required to achieve convergence - practical implementations of a fixed restart period should choose k based on the expected susceptibility of the recursion process to corruption by rounding errors.

Generally, larger linear systems are more susceptible to rounding errors due to the number of computational operations involved in each iteration. From Table 6.10, the optimum restart period decreases as Co is decreased, indicating that the more non-symmetric the matrix, the more sensitive the process is to rounding errors. Thus, the optimum value of k for this problem is expected to decrease with n, Co and Pe (the latter is included because it also affects the amount of non-symmetry in the matrix).

Again, the requirement for the value k violates the desired property that the solver should require no external parameters (see Section 2.3.2).

Figure 6.7 corresponds to Figure 6.1 for a restart of fixed period $k = 20$.

6.3.4 Quadratic Bi-CGSTAB Polynomials

In the problems examined so far in this chapter, when the advective part of the matrix is dominant, the complex part of the eigenspectrum becomes comparable in magnitude to the real part. This can be seen in Figure 6.8 where Graph (a) has a relatively low Courant number and hence is the low advection case and Graph (b) is the high advection case.

The (exact arithmetic) performance of Krylov sub-space methods depends on the eigenspectrum of the matrix. From [12], the Bi-CGSTAB error polynomial is

$$\boldsymbol{r}_i = \tilde{\varphi}_i(A)\phi_i(A)\boldsymbol{r}_0 = \sum_{k=1}^{n} \rho_k \tilde{\varphi}_i(\lambda_k) \prod_{j=1}^{i} \left(\frac{\lambda_j - \lambda_k}{\lambda_j} \right) \boldsymbol{v}_k$$

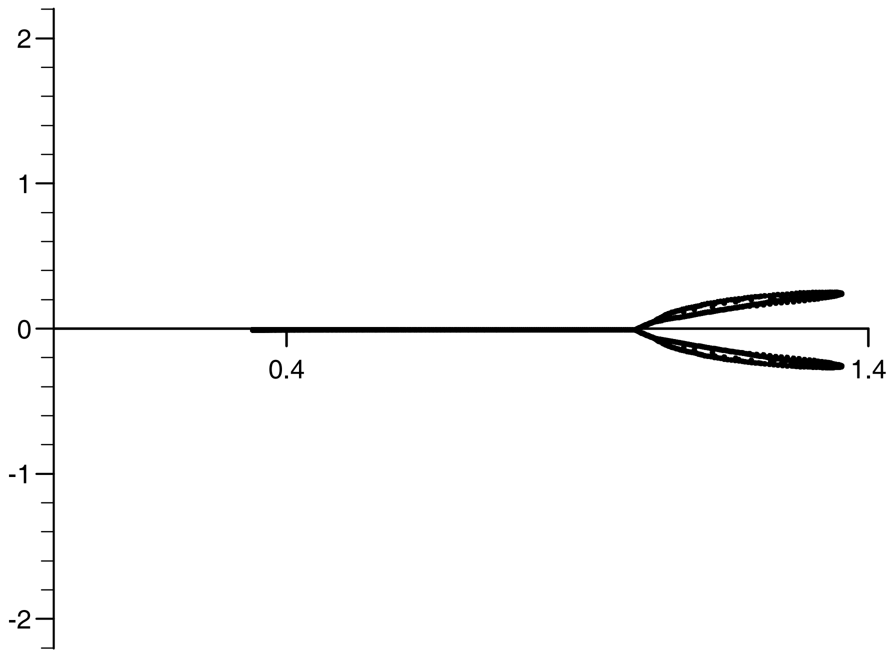
where $A\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$ and $\boldsymbol{r}_0 = \sum_{k=1}^{n} \rho_k \boldsymbol{v}_k$. This means that, for the error mode in the direction $\boldsymbol{v}_k$ to be damped by the iteration process, either λ_j or a root of $\tilde{\varphi}_i$ must be close to λ_k. The $\tilde{\varphi}_i$ defined in (2.19) can only produce real roots, i.e. $\omega_1^{-1}, \omega_2^{-1}, \ldots, \omega_i^{-1}$. Hence it cannot contribute to the convergence of the method when the eigenvalues of the matrix A have a significant complex part. In this case, Bi-CGSTAB computes ω_i which are close to zero and therefore very sensitive to rounding error.

In [36], the problem of Bi-CGSTAB not being capable of representing complex eigenvalues is tackled by the use of a quadratic polynomial for the building blocks of $\tilde{\varphi}_i$ in (2.19) i.e.

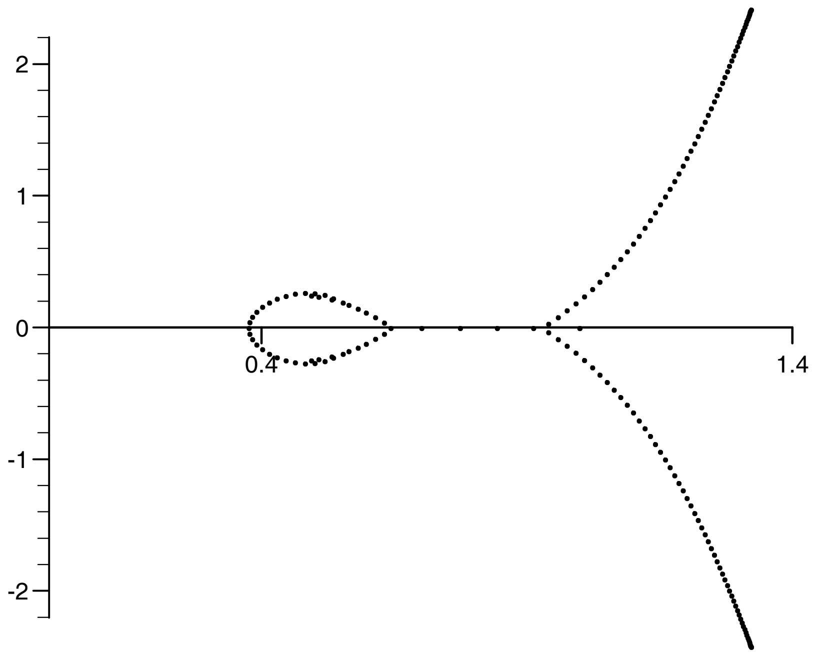
$$\tilde{\varphi}_i(x) = (1 - \omega_i x - \gamma_i x^2)\tilde{\varphi}_{i-1}(x), \quad \omega_i, \gamma_i \in \mathbb{R}.$$

Each factor can have complex roots so the Bi-CGSTAB part of the error polynomial can have roots near complex eigenvalues. This method is known as Bi-CGSTAB2. In practice, an iteration with this method consists of two stages; in the first, a linear polynomial such as that in Bi-CGSTAB is used, while in the second stage, the linear polynomial from the first stage is corrected to a quadratic polynomial. This approach does not address the problem of a breakdown in the convergence history arising during the first stage of each iteration.

A more general approach for producing complex eigenvalues is taken in [74]. In this approach, a general degree ℓ factor is used to construct the $\tilde{\varphi}_i$ in (2.19). The resulting method is known as Bi-CGSTAB(ℓ). For $\ell = 2$ this approach



(a) $Co = 0.5$, $Pe = 1$, (gives $0.3419 \leq (Re) \leq 1.3542$, $|(Im)| \leq 0.2590$)



(b) $Co = 20$, $Pe = 5$, (gives $0.3768 \leq (Re) \leq 1.3227$, $|(Im)| \leq 2.2051$)

Figure 6.8: Eigenspectra of preconditioned matrices in sample robustness tests

gives the same results as Bi-CGSTAB2 (in the absence of rounding errors), but the Bi-CGSTAB(2) implementation is numerically more stable (never using the usual Bi-CGSTAB linear factors) and more efficient, requiring 14 SAXPYs and 9 vector-vector products per 4 matrix-vector multiplications, as opposed to the 22 SAXPYs and 11 vector-vector products of Bi-CGSTAB2. Due to its better efficiency and stability properties (and ease of extension to even higher order polynomials) this implementation is the one used here. The code used in the numerical experiments on this method was written by Diederik Fokkema and obtained from Gerard Sleijpen.

The method used in the preliminary tests is Bi-CGSTAB(2) - Bi-CGSTAB based on quadratic polynomials - since it is the lowest polynomial required to generate complex roots.

Pe	n	Co	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
1	802	0.5	$< \epsilon_M$	36
		1	$< \epsilon_M$	56
		2	$< \epsilon_M$	96
		5	$< \epsilon_M$	196
		10	$< \epsilon_M$	372
		20	$< \epsilon_M$	632
		40	$< \epsilon_M$	924
5	162	0.5	$< \epsilon_M$	36
		1	$< \epsilon_M$	52
		2	$< \epsilon_M$	88
		5	$< \epsilon_M$	180
		10	$< \epsilon_M$	256
		20	$< \epsilon_M$	336
		40	$< \epsilon_M$	368

Table 6.11: Performance of Bi-CGSTAB(2) in robustness tests

Table 6.11 shows the performance of Bi-CGSTAB(2) in the robustness tests.

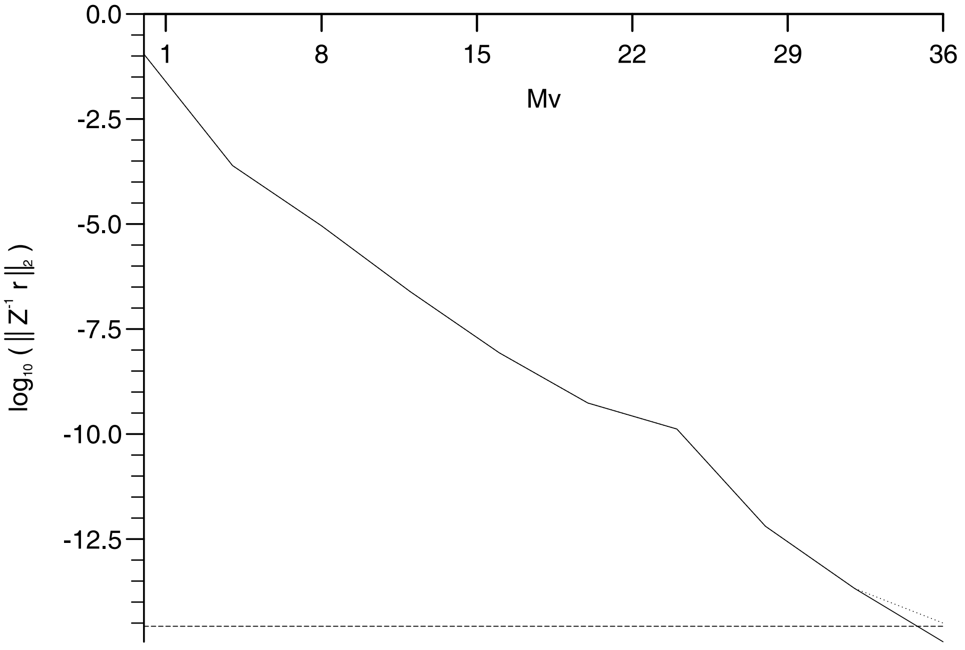
Convergence appears to be fully robust, i.e. convergence is achieved in all cases. Also, comparing these results with those from the restart criteria in Section 6.3.3, it can be seen that the Bi-CGSTAB(2) convergence is appreciably faster, even taking into account the small amount of extra work required to perform the two parameter minimisation in the quadratic steepest descent step.

Figure 6.9 shows the iteration histories for two of the cases in Table 6.11. Since a two parameter local steepest descent step is performed in Bi-CGSTAB(2) the resulting iteration history is smoother than standard Bi-CGSTAB (c.f. Figure 6.1).

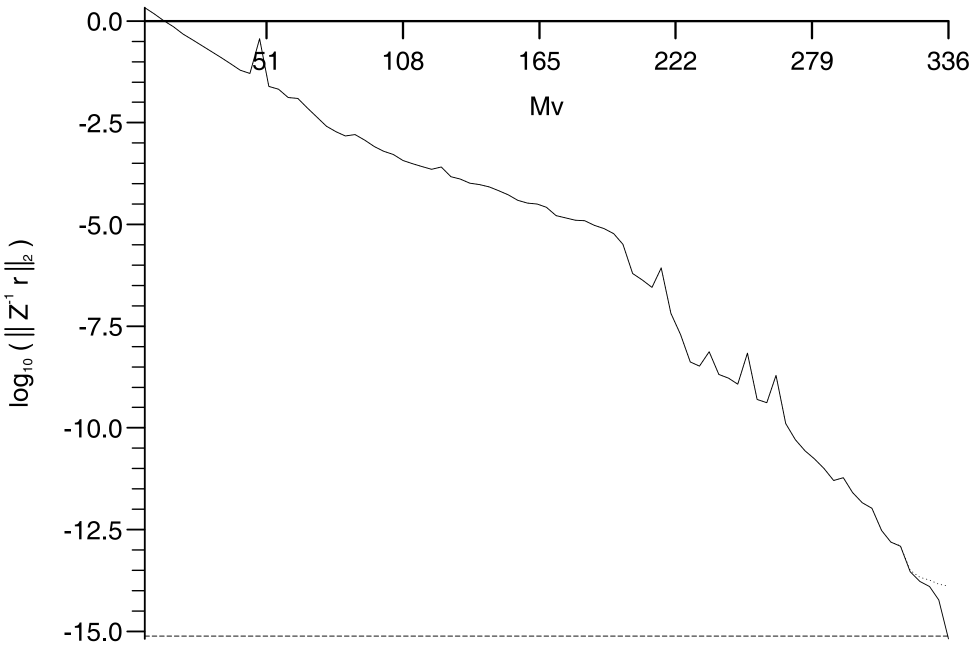
In [74], the use of "higher-than-quadratic" order polynomials is advocated as an efficient acceleration method. Table 6.12 shows the performance of Bi-CGSTAB(ℓ) based on cubic and quartic polynomials.

			Bi-CGSTAB(3)		Bi-CGSTAB(4)	
Pe	n	Co	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$	Mv
1	802	0.5	$< \epsilon_M$	36	$< \epsilon_M$	40
		1	$< \epsilon_M$	60	$< \epsilon_M$	56
		2	$< \epsilon_M$	90	$< \epsilon_M$	88
		5	$< \epsilon_M$	192	$< \epsilon_M$	192
		10	$< \epsilon_M$	366	$< \epsilon_M$	360
		20	$< \epsilon_M$	642	$< \epsilon_M$	624
		40	$< \epsilon_M$	882	$< \epsilon_M$	888
5	162	0.5	$< \epsilon_M$	36	$< \epsilon_M$	40
		1	$< \epsilon_M$	48	$< \epsilon_M$	48
		2	$< \epsilon_M$	84	$< \epsilon_M$	80
		5	$< \epsilon_M$	174	$< \epsilon_M$	168
		10	$< \epsilon_M$	252	$< \epsilon_M$	264
		20	$< \epsilon_M$	324	$< \epsilon_M$	344
		40	$< \epsilon_M$	372	$< \epsilon_M$	424

Table 6.12: Performance of Bi-CGSTAB(ℓ) ($\ell = 3, 4$) in robustness tests



(a) $Co = 0.5,\ Pe = 1$



(b) $Co = 20,\ Pe = 5$

Figure 6.9: Sample iteration histories of Bi-CGSTAB(2) in robustness tests

Again, the convergence is smooth, fast and robust in all the cases presented in Table 6.12. However, on comparison with the results in Table 6.11, it can be seen that the extension to higher order polynomials does not result in a general increase in the rate of convergence for these cases examined.

Back-tracking slightly, since CG-S (the Bi-CGSTAB predecessor) does not use linear factors in the underlying polynomials, it should also perform well on advection dominated problems where Bi-CGSTAB fails. Table 6.13 shows the performance of CG-S on the robustness test cases. By comparison with the matrix-vector multiplication count in Table 6.11, CG-S requires approximately the same amount of work to achieve convergence as Bi-CGSTAB(2).

Pe	n	Co	$\min_i \frac{\|Z^{-1}\boldsymbol{r}_i\|_2}{\|Z^{-1}\boldsymbol{b}\|_2}$ (true value in brackets)		Mv
1	802	0.5	$< \epsilon_M$	$(< \epsilon_M)$	40
		1	$< \epsilon_M$	(6.49×10^{-16})	66
		2	$< \epsilon_M$	(7.03×10^{-16})	100
		5	$< \epsilon_M$	(2.45×10^{-15})	228
		10	$< \epsilon_M$	(1.37×10^{-13})	502
		20	$< \epsilon_M$	(4.04×10^{-11})	716
		40	$< \epsilon_M$	(5.05×10^{-5})	882
5	162	0.5	$< \epsilon_M$	$(< \epsilon_M)$	42
		1	$< \epsilon_M$	$(< \epsilon_M)$	48
		2	$< \epsilon_M$	(3.77×10^{-16})	92
		5	$< \epsilon_M$	(1.44×10^{-13})	196
		10	$< \epsilon_M$	(3.70×10^{-13})	250
		20	$< \epsilon_M$	(2.81×10^{-10})	304
		40	$< \epsilon_M$	(1.89×10^{-8})	338

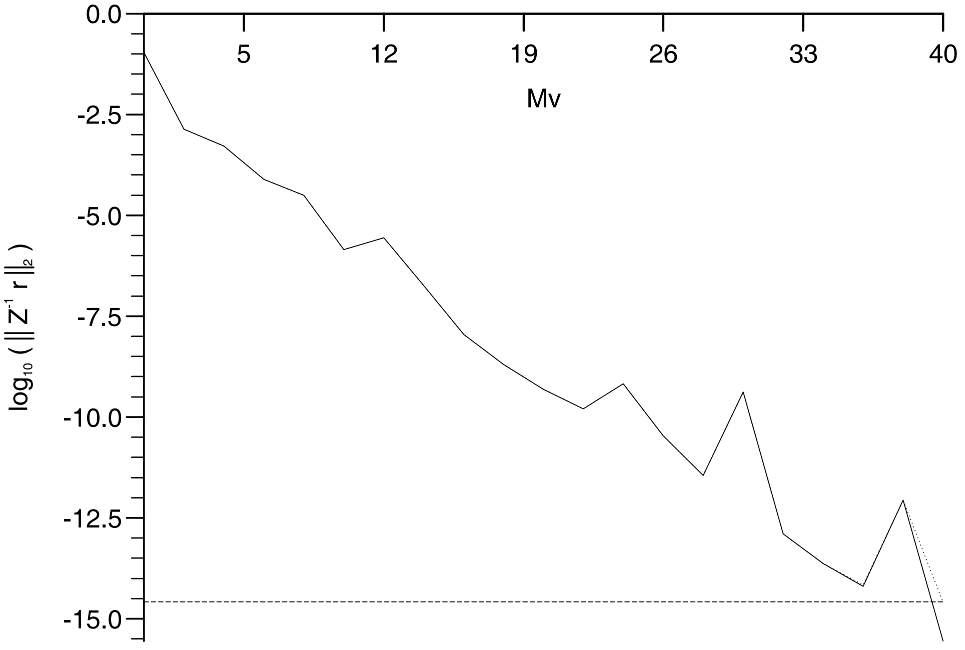
Table 6.13: Performance of CG-S in robustness tests

The 2-norm of the true preconditioned residuals at convergence are also shown in Table 6.13. In many of the cases, this norm is a long way from the converged recurrence value. Note that for all the previously converged examples with other iterative methods, the norm of the true preconditioned residual is at most $O(10^{-15})$.

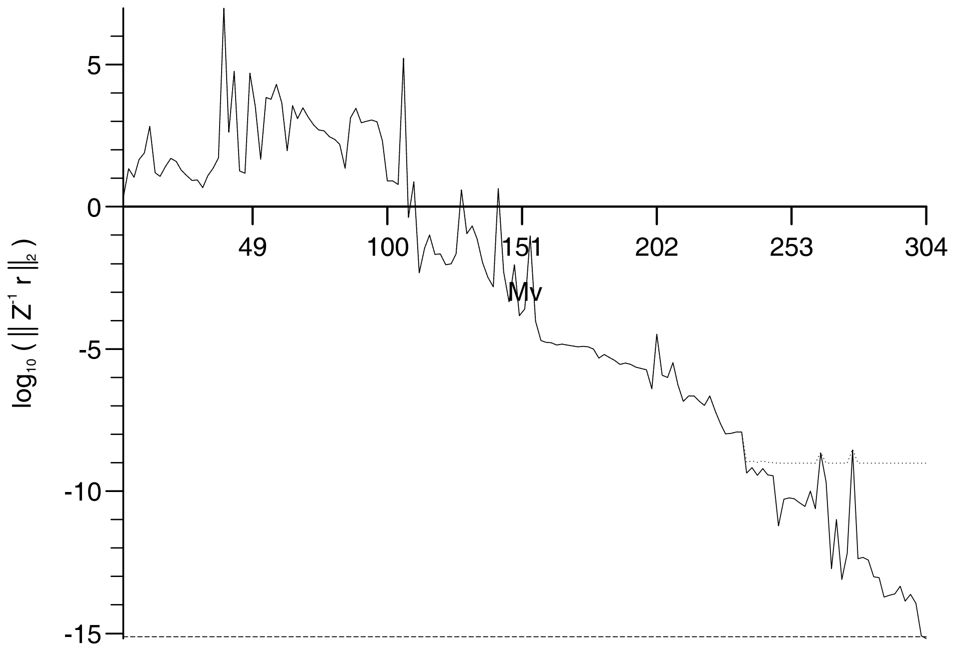
Figure 6.10 shows iteration histories for CG-S on some of the robustness test cases. The second graph gives an indication of the cause of the difference between the true and recurrence residuals at convergence. The norm of the true preconditioned residual (dotted line) stagnates at $\approx 9.57 \times 10^{-10}$ and owing to its irregular convergence behaviour, CG-S produces a peak of $\approx 9.68 \times 10^{6}$ earlier in the iteration history. In all the cases where the true residual stagnates, the ratio of the minimum residual norm achieved to the maximum one generated is of the order of the machine round-off unit, ϵ_M. This coincides with the theory for the finite precision behaviour of other methods given in [33].

Owing to its irregular convergence behaviour (i.e. generating large spikes in the iteration history) causing the true residual norm to stagnate before convergence while the recurrence based residual continues to decrease, CG-S is not as robust as Bi-CGSTAB(2).

Note, spikes also occur in some of the iteration histories of restarted Bi-CGSTAB when the restart tolerance is too tight, but this does not lead to a stagnation of the true residual for that method as the spikes, although part of the iteration history, are not part of the recurrence relation after a restart.



(a) $Co = 0.5$, $Pe = 1$



(b) $Co = 20$, $Pe = 5$

Figure 6.10: Sample iteration histories for CG-S in robustness tests

6.4 Robustness Tests on a Similar Matrix

In order to confirm the theory that the divergent behaviour apparent in Figure 6.1 is caused by rounding errors corrupting sensitive values that are generated when Bi-CGSTAB cannot cope with complex eigenvalues, tests are carried out on a diagonal matrix which is similar to one from the robustness tests.

Definition 6.2 *Two matrices A and B are* similar *if*

$$B = WAW^{-1}$$

for some nonsingular matrix W. Similar matrices have identical eigenvalues.

Given the eigenspectrum of a matrix $\{\lambda_k\}$, it is relatively trivial to generate a similar 2×2 block diagonal matrix by setting the diagonal entry of the similar matrix to λ_k for $\lambda_k \in \mathbb{R}$ and placing a 2×2 block

$$\begin{pmatrix} a_k & b_k \\ -b_k & a_k \end{pmatrix}$$

on the diagonal for $\lambda_k = a_k \pm ib_k \in \mathbb{C}$.

In this section, such a matrix is generated to be similar to the matrix with the eigenspectrum shown in Figure 6.8(b). The divergent behaviour of Bi-CGSTAB for the tracer system possessing this eigenspectrum has already been shown in Figure 6.1(b).

As the similar matrix is a (2×2 block) diagonal matrix, hardly any rounding errors occur in the underlying Lanczos process, so the divergent behaviour should not occur with Bi-CGSTAB, even though the matrix has an eigenspectrum with a strong complex component.

A dummy system is used to test this. The matrix is the 2×2 block diagonal similar matrix, the right-hand side vector is a vector of ones i.e. $\boldsymbol{b} = [1, 1, 1, \ldots, 1]^T \in \mathbb{R}^n$ and the initial iterate is taken to be the vector $\boldsymbol{x}_0 \in \mathbb{R}$ with entries $x_0(j) = j^{1/4}$. The same preconditioner and convergence tolerance as for the previous robustness tests is used (i.e. the diagonal preconditioner and the round-off unit respectively).

The iteration history for Bi-CGSTAB on the similar matrix robustness test is shown in Figure 6.11. The residual norm stagnates but does not diverge. The

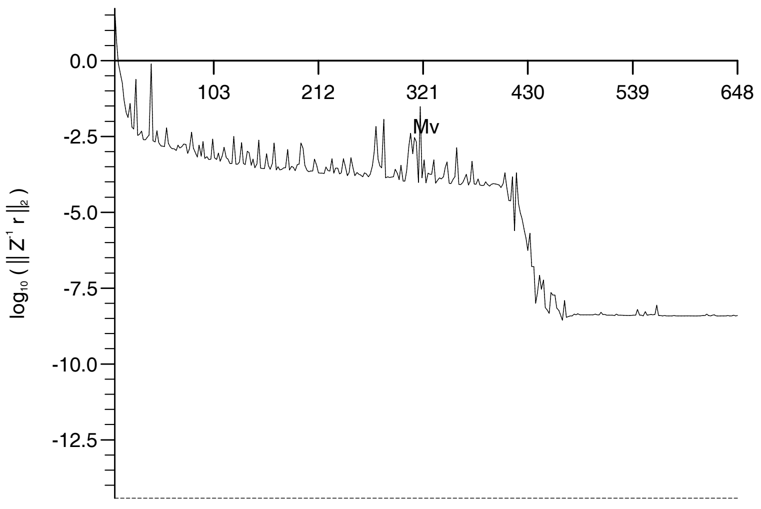


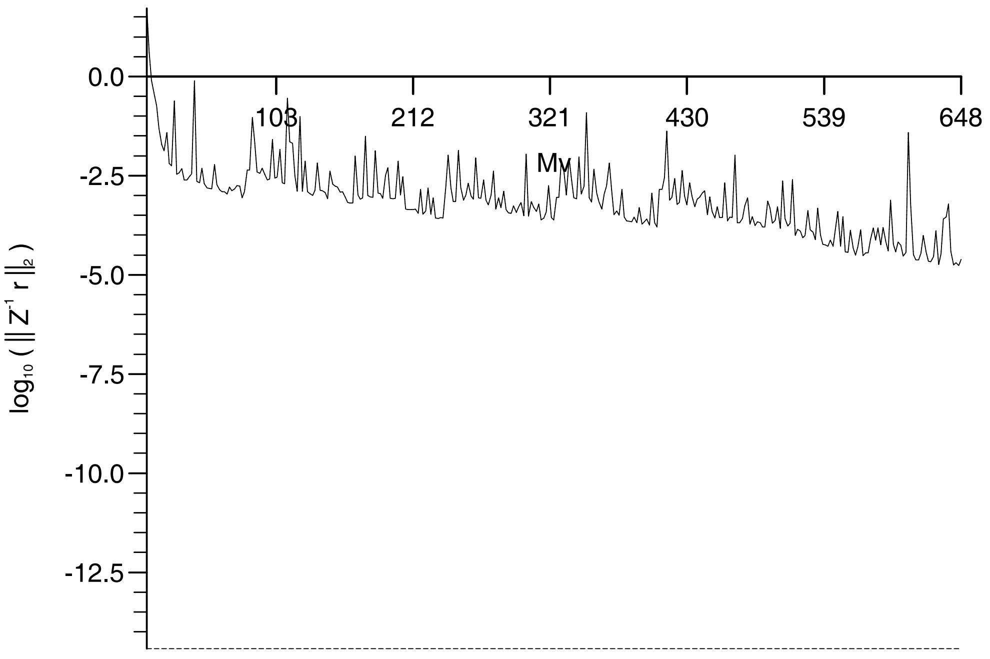
Figure 6.11: Iteration history for Bi-CGSTAB on 2×2 block diagonal matrix

stagnation shows that Bi-CGSTAB still cannot cope with the complex components in the eigenspectrum. The lack of divergence suggests that any round-off errors which occur are far less critical when the matrix is diagonal.

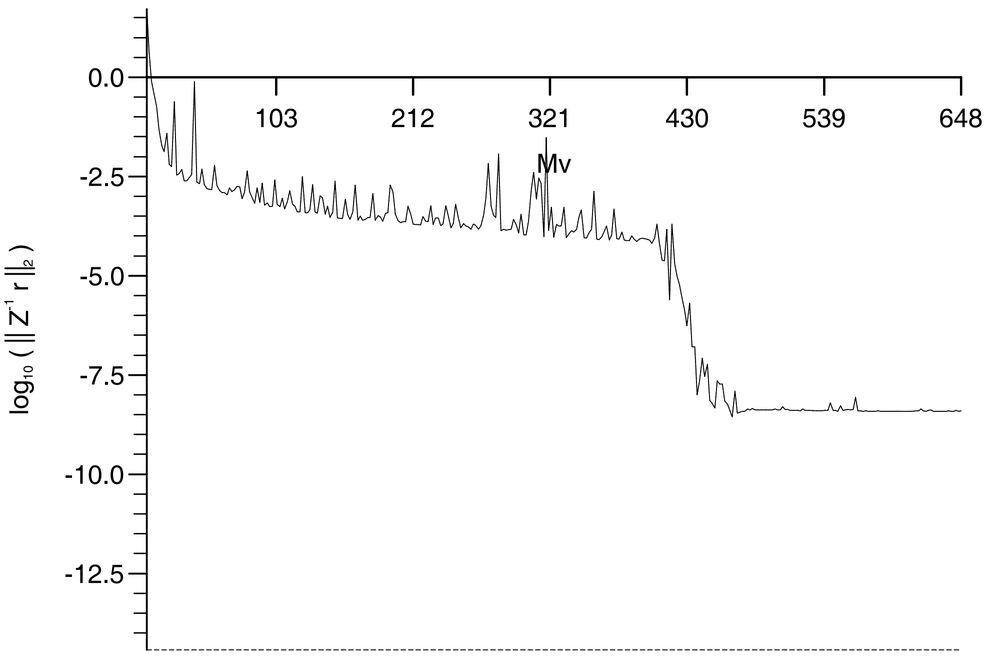
Since rounding errors do not appear to be a problem in this case, restarting (i.e. discarding the Krylov subspace to flush out any rounding errors in the recurrence relation) is not expected to be of any benefit. In fact, it is likely to cause a degradation in the performance as it discards a good Krylov subspace at each restart. The iteration histories for Bi-CGSTAB restarted after a fixed number of iterations (as in Section 6.3.3) are shown in Figure 6.12 with two different restart periods. The restart has, as expected, a detrimental effect which, at best, leaves the iteration history unaffected.

If the stagnation in Figure 6.11 is due to Bi-CGSTAB not being able to produce approximations to complex eigenvalues, Bi-CGSTAB(2) and CG-S should both converge without problems. The iteration histories for these methods are shown in Figure 6.13, both perform well as expected.

The results in this section reinforce the theory that the cause of the divergence is sensitive values (which occur when there is a large complex component in the eigenspectrum) being corrupted by rounding error and perturbing the recurrence relations.

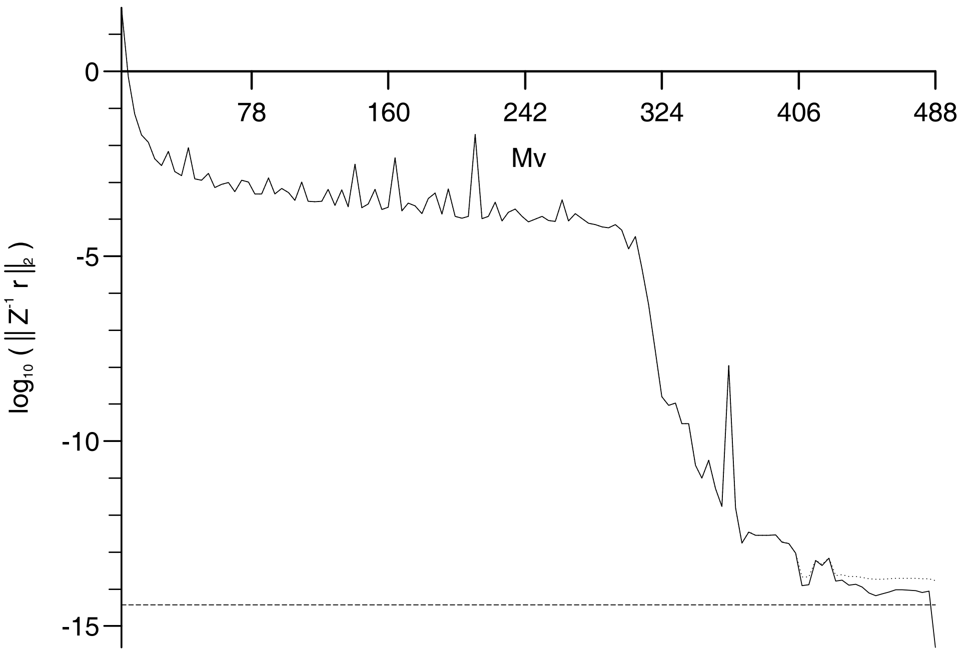


(a) Bi-CGSTAB($k = 40$)

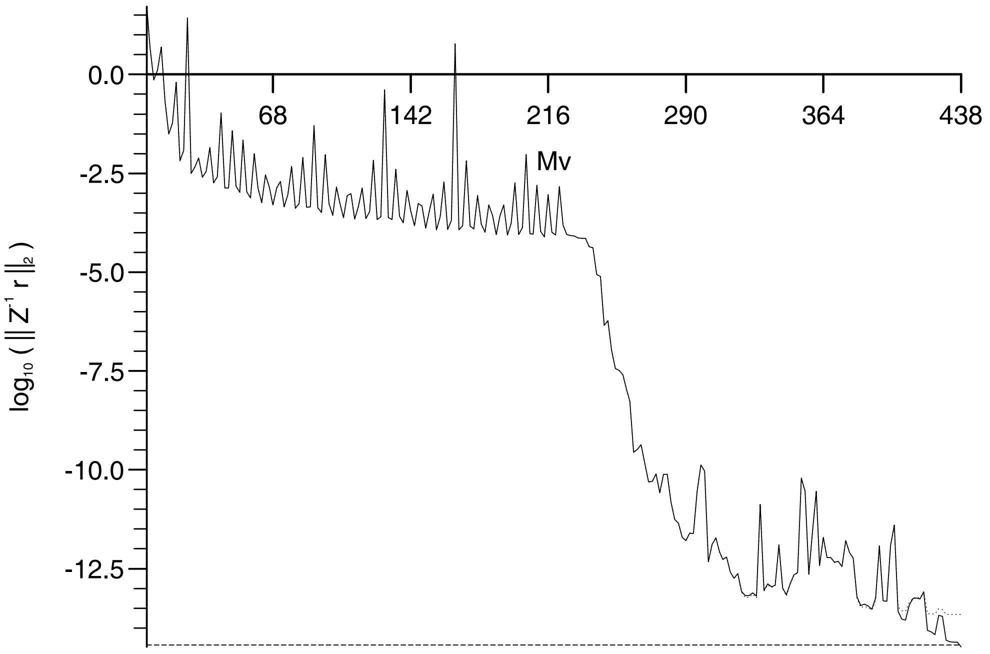


(b) Bi-CGSTAB($k = 500$)

Figure 6.12: Iteration histories for restarted Bi-CGSTAB on 2×2 block diagonal matrix



(a) Bi-CGSTAB(2)



(b) CG-S

Figure 6.13: Iteration histories for Bi-CGSTAB(2) and CG-S on 2×2 block diagonal matrix

6.5 Efficiency of Bi-CGSTAB(2)

Of all the Bi-CGSTAB extensions described and investigated in this section, the Bi-CGSTAB(2) method is considered to be the most effective at overcoming the divergence problems encountered in the robustness tests.

In order to compare the computational efficiency of this method with that of Bi-CGSTAB, it is applied to the tests cases from Section 6.1.

As before, the test cases are run from the initial condition for 10 time-steps and the average performance is assessed over this period. The same grids, time-step sizes and convergence tolerance as in Section 6.1 are used. The average number of Mvs to convergence and the average time spent in the solver routine are both shown in Table 6.14.

Pe	n	Co	Mvs to convergence	Time in solver / seconds
1	802	0.5	14.0	0.72
		1	20.0	0.99
		5	61.2	2.86
5	162	0.5	12.0	0.13
		1	17.6	0.18
		5	61.2	0.56

Table 6.14: Average Mvs to convergence and time in solver

The results for Bi-CGSTAB(2) in Table 6.14 can be compared with those for Bi-CGSTAB, GMRES and GMRES(10) in Tables 6.1 and 6.2. Bi-CGSTAB(2) takes approximately the same number of Mvs to achieve convergence as Bi-CGSTAB, and spends less time in the solver than both Bi-CGSTAB and GMRES(10). Hence Bi-CGSTAB(2) is more robust than Bi-CGSTAB and is also more efficient on these test problems.

6.6 Preconditioner for Bi-CGSTAB(2)

As with the CG method used to solve the linear systems with SPD matrices in Chapter 5, the preconditioning technique used is an important feature of the overall performance of the solver. Selection of the appropriate preconditioner can lead to an acceleration of the method (both in terms of the number of iterations and the required computational time), and also allow mesh independent convergence to be achieved.

In the 1-D problem used to test robustness and efficiency in this chapter, the structure of the matrix is such that there is no fill-in with Gaussian elimination. So there is not much flexibility to examine preconditioning matrices for the method on the 1-D problem since a preconditioner as basic as the incomplete factorisation is, in effect, the inverse.

Hence, in order to examine preconditioning of the Bi-CGSTAB(2) solver, the constant dispersion coefficient Henry problem test case from the tests on the performance of the SPD solver in Chapter 5 is used. A representative matrix is generated with the same conditions as in the SPD tests but time-step is changed to 300s (so that the maximum Courant number on the linear mesh is typical of that which would be encountered in practice).

As in Section 5.1.3, three grids (of varying distortion) are used to investigate the dependence of the convergence of the preconditioned method on the mesh.

The performance of Bi-CGSTAB(2) with diagonal and $ILDM^T$ factorisation preconditioners on the representative matrix generated at the first coupling iteration of the 3rd time-step is shown in Table 6.15.

Mesh type	Co_{max}	Pe_{max}	Number of Mvs		Time in solver / seconds	
			Diagonal	$ILDM^T$	Diagonal	$ILDM^T$
Linear	3.58	12.80	36	12	0.68	0.71
Quadratic	70.43	14.31	112	12	2.01	0.71
Cubic	5047.23	26.37	260	12	4.67	0.72

Table 6.15: Comparison of diagonal and $ILDM^T$ preconditioners

As with the matrix from the fluid continuity equation, the convergence depends on the mesh when the diagonal preconditioner is used. The theory from [89] (described in Section 5.2.1) for bounding the condition number of the diagonally preconditioned matrix (and hence the convergence rate) does not apply to non-symmetric matrices. However, the convergence rate does appear to be mesh-independent with the $ILDM^T$ preconditioner - requiring 12 Mvs to achieve convergence regardless of the amount of distortion. Unlike the results with the $ILDL^T$ preconditioner used with CG on the matrix from the fluid continuity equation in Section 5.1.4, this apparent mesh independence applies as the number of points in the mesh varies (e.g. the problem on an 11×6 mesh needs 8 Mvs for convergence, on a 21×11 mesh it needs 12, while on a 41×21 mesh it still only needs 12).

For the cases considered, the use $ILDM^T$ preconditioner is generally more efficient (in terms of less time in the solver) than the diagonal preconditioner. This, combined with the apparent mesh independent convergence, indicates that the $ILDM^T$ preconditioner is the better preconditioner for the Bi-CGSTAB(2) method.

6.7 Concluding Remarks

Two non-symmetric iterative solvers, Bi-CGSTAB and GMRES, have been compared. Bi-CGSTAB performs as well as GMRES in terms of the CPU time required to achieve convergence. However, in harsher tests (i.e. when the non-symmetric part of the matrix dominates), Bi-CGSTAB does not converge due to sensitive values in the iteration process being corrupted by rounding errors. These sensitive values are generated due to the lack of a facility for representing complex eigenvalues in the method.

Of the remedies examined, only restarting and the use of higher order polynomials have any degree of success.

Restarting keeps the build-up of rounding errors in check. However, all the restart criteria tested in this thesis require the use of an external parameter to be effective. Another drawback is that restarting treats the problem by addressing its symptom, it does not eliminate the cause.

Quadratic polynomials, i.e. Bi-CGSTAB(2), allows the complex eigenvalues to be represented. Hence the sensitive values do not occur and the convergence problem is solved. As this remedy treats the cause of the problem, it is the approach selected in this thesis. An added advantage is that Bi-CGSTAB(2) is smoother and converges quicker than (the linear polynomial based) Bi-CGSTAB.

The numerical experiments in Section 6.6 indicate that the $ILDM^T$ preconditioner is more effective, both in terms of computational effort and the mesh independence of convergence, than the diagonal preconditioner for Bi-CGSTAB(2).

Chapter 7

Coupling of the Governing Equations

The governing equations described in Section 3.1 are coupled together and cannot be solved independently of each other. The source of the coupling is the dependency of the fluid density (ρ) on the dimensionless contaminant concentration (c) in the constitutive equation (3.6). Two approaches for solving such a system of coupled equations are (i) to use a coupling iteration and (ii) to solve the full system.

In a coupling iteration, the equations are solved individually and sequentially. An initial approximation is generated for one of the variables (the iteration variable) at a particular instant in time. By careful selection of the order in which the equations are solved (and which variables are solved for in these equations), it is possible to remove any non-linearity in the system and generate a new approximation for the iteration variable. One possible coupling iteration approach is given in Algorithm 3.2. That particular algorithm is the method used to solve the coupled system of equations in the previous chapters of this thesis.

In the full system approach, the equations are solved simultaneously - all the unknown variables are treated as a single unknown vector and the system is written in matrix operator form. A full system approach is used in [72] for the fluid mass balance equation and Darcy's law. There are many ways of writing the full system for the governing equations in the system. One of these is shown in (7.1).

$$\begin{pmatrix} 1 & 0 & 0 & -\epsilon \\ \begin{pmatrix} \phi\dfrac{\partial}{\partial t} \\ -\boldsymbol{\nabla}.\underline{\boldsymbol{K}}(\boldsymbol{\nabla} z) \end{pmatrix} & -\frac{1}{\mathrm{g}}\boldsymbol{\nabla}.\underline{\boldsymbol{K}}\boldsymbol{\nabla} & 0 & 0 \\ \underline{\boldsymbol{K}}(\boldsymbol{\nabla} z) & \frac{1}{\mathrm{g}}\underline{\boldsymbol{K}}\boldsymbol{\nabla} & \rho & 0 \\ 0 & 0 & 0 & \begin{pmatrix} \rho\phi\dfrac{\partial}{\partial t} + \rho\boldsymbol{q}.\boldsymbol{\nabla} \\ -\boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\rho\boldsymbol{\nabla} \end{pmatrix} \end{pmatrix} \begin{pmatrix} \rho \\ p \\ \boldsymbol{q} \\ c \end{pmatrix} = \begin{pmatrix} \rho_0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7.1}$$

When discretised, this system is $3d+1$ (where d is the dimension of the physical problem) times the size of the individual systems solved in the coupling iteration approach.

As the governing equations are non-linear, the full system is invariably non-linear. Hence a large sparse non-linear solver is required.

Non-linear solvers operate by iteration. These can have a quadratic rate of convergence, e.g. the Newton method which uses the Jacobian of the matrix in the system, but if this is not computable (as is the case in (7.1)), then approximate derivatives must be used and the convergence rate is closer to linear. Hence it is just as reasonable to use the coupling iteration approach. The coupling iteration resembles a non-linear Gauss-Seidel-type iteration applied to the full system, with any variables causing non-linearity in the matrix updated (to the most recently calculated value) as they are required.

The full system approach is not investigated in this thesis. Instead, some coupling approaches are investigated in the remainder of this chapter.

7.1 Coupling Iteration Approaches

The coupling iteration approach, apart from decreasing the size of the discrete systems being solved, and removing the non-linearity, allows the properties of the discrete systems for the individual equations to be exploited. An example of this is the use of the CG solver with different preconditioners for the SPD systems arising from the discretisation of the fluid continuity equation and the Darcy velocity vector components.

There are other options for the coupling iteration, i.e. the governing equations can be solved in a different order and with a different subject variable. In order to decide which coupling iteration option is the most effective, analysis is needed on the rate of convergence of the various coupling iteration approaches. However, such theory is difficult due to the nature of the governing equations e.g. the lack of a maximum principle for many of the differential operators involved. Such analysis is not attempted here - the coupling iteration given in Algorithm 3.2 is used throughout without consideration given to the other options; no convergence theory is given for this algorithm for the reasons already given in this paragraph, it is assumed to converge as it is the method used in the literature e.g. [30].

In the physical systems considered in this thesis, the maximum fluid density is not very different from freshwater density ρ_0 i.e. in (3.6) $\epsilon = 0.02499\rho_0$ so $\rho = 1.02499\rho_0$ when $c = 1$. Hence, the coupling in the system is of a relatively weak nature. An insight into the behaviour of the system can be gained by examining the tracer case where the fluid density is unaffected by the contaminant concentration (i.e. $\epsilon = 0$).

In the tracer case, the governing equations reduce to

$$\boldsymbol{\nabla}.\boldsymbol{q} = 0, \tag{7.2}$$

$$\boldsymbol{q} = -\underline{\boldsymbol{K}}\left(\frac{\boldsymbol{\nabla} p}{\rho_0 g} + \boldsymbol{\nabla} z\right), \tag{7.3}$$

$$\phi\frac{\partial c}{\partial t} + \boldsymbol{q}.\boldsymbol{\nabla} c - \boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\boldsymbol{\nabla} c = 0, \tag{7.4}$$

so that the full system can be written as

$$\begin{pmatrix} \boldsymbol{\nabla}. & 0 & 0 \\ \rho_0 g & \underline{\boldsymbol{K}}\boldsymbol{\nabla} & 0 \\ 0 & 0 & \left(\phi\frac{\partial}{\partial t} + \boldsymbol{q}.\boldsymbol{\nabla} - \boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\boldsymbol{\nabla}\right) \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ p \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ -\rho_0 g\underline{\boldsymbol{K}}\boldsymbol{\nabla} z \\ 0 \end{pmatrix}.$$

This system can be seen to be reducible. The leading principal 2×2 matrix can be used to form an independent system - this reflects the fact that the governing equation for the contaminant does not influence the governing equations for the fluid. However, the reverse is not true as the governing equations for the fluid influence the governing equation for the contaminant through the Darcy velocity field they generate.

Due to the reducibility of the system, (7.2) and (7.3) can be solved in isolation to give the Darcy velocity (in fact (7.2) is only needed in order to incorporate the Dirichlet pressure boundary conditions). If the boundary conditions are independent of time, then the Darcy velocity is independent of time so this only needs to be calculated once at the beginning of the simulation. After this has been done, the tracer concentration can be obtained at all required times by solving (7.4). This is essentially the approach used in the 1-D case examined in Section 4.1.

Returning to the case of interest in this thesis (non-passive transport), the full system can not be reduced in the same way as the tracer system due to the non-zero off-diagonal entry in the constitutive equation part of (7.1). However, as already stated, the coupling is of a relatively weak nature, so this value is nearly zero. Hence the non-passive case in this thesis is "close" in some sense to the (reducible) tracer case. This suggests a spectrum of approaches for the coupling iteration which mimic, to varying degrees, the tracer system approach.

A selection of these approaches follows, with the cheapest approach being first and the most computationally demanding approach last.

1. *No coupling.* In this approach, the contaminant is treated as a tracer, the fluid density is taken as the freshwater one at all times, and the system (7.2), (7.3) & (7.4) is solved rather than (3.1), (3.2) & (3.4).

2. *Segol coupling* [72]. In Segol coupling, a large (or macro-) time-step (of size ΔT) is performed on (3.1) and (3.2) using the most recent concentration and assuming the transport to be passive. Then smaller time-steps (of size Δt where $\Delta T = m\Delta t$ for some positive integer m) are performed on (3.4) using the most recently calculated fluid density and values interpolated from the Darcy velocity vector at the macro-time-step.

3. *Partial coupling* [30]. In this approach, the transport is treated as passive within a time-step so that, once the fluid density is approximated at the beginning of the coupling process, it is assumed to be correct during the current time-step. This corresponds to Algorithm 3.2 with Stage 8 omitted. In this approach, the dimensionless concentration calculated at the end of the time-step, and the density for the time-step (calculated at the beginning

of the time-step) do not match.

4. *Full coupling.* This approach is Algorithm 3.1 in its entirety. It is the only coupling iteration technique which guarantees a valid solution state (i.e. one in which the fluid density and dimensionless contaminant concentration match) at the end of all time-steps. This is the coupling approach that has been used so far throughout this thesis.

The uncoupled and Segol coupling approaches are not investigated in this thesis. The use of no coupling has been investigated briefly in [30] where it is shown to lead to unreliable results. Segol coupling is used in [72] but it is not compared with any other approaches. A comparison of partial and full coupling is made in the remainder of this chapter.

7.2 Comparison of Partial and Full Coupling

This work is presented last because it involves comparisons that use timing data from numerical experiments on long term simulations, so the details of the methods used to solve the individual governing equations had to be finalised first. These details were examined in Chapters 4-6.

The purpose of this part of the thesis is to look briefly at the strength and effect of the dependency of ρ on c in the coupling iteration. Logically, the fully coupled approach (assuming convergence occurs) should lead to the most accurate and reliable results, but any loss in accuracy incurred by only using partial coupling may be compensated by the lower computational expense involved.

Partial and full coupling are compared in [30] where partial coupling is demonstrated to give results which are acceptably close to those from full coupling. However, that work does not give a quantitative comparison of the two approaches in terms of either accuracy or computational cost. In order to allow such a comparison in this thesis, an "exact" solution is generated for the dimensionless contaminant concentration in the Henry problem (Section 4.2) at time $t = 500$s.

A true exact solution is not feasible. Instead, a solution on a much finer spatial grid (a 81×41 linear mesh) and with a very small time-step ($\Delta t = 1$s throughout the simulation) is generated, the relative convergence tolerance for all the linear solvers is $\tau = 10^{-15}$ and a full coupling iteration is used with a pointwise convergence tolerance of 10^{-15}. This solution is very expensive to generate; its only purpose is to provide a benchmark in which as many of the sources of error as possible have been minimised. A sample time of 500s is used because it is relatively early in the simulation, hence it is sufficiently far from the steady state solution to give a representation of the temporal errors.

After the "exact" solution is generated, more computationally feasible solutions are generated for comparison. These are less accurate solutions, they are obtained on coarser meshes, with the tolerances for the linear solvers and the full coupling iteration being 10^{-8}. In order to allow comparison with the "exact" results, adaptive time-stepping strategies are not used so the time step is kept constant during the simulation.

As already stated, the fully coupled approach is expected to be more accurate as it reduces the coupling error. The difference between the concentration at the end of a simulation and the "exact" solution is measured by the relative error. This is obtained as follows. The nodal values of the "exact" solution, u_J^{fine}, are projected onto the coarse grid where the nodes coincide, the other nodal values from the "exact" solution are ignored. Then the 2-norm of the difference between the projected nodal values and the ones from the simulation on the coarse grid, u_J^{coarse}, is calculated. This is normalised by the 2-norm of the projected "exact" solution to give the relative error. i.e.

$$\text{Relative error} = \frac{\sqrt{\sum_{J}^{coarse\ nodes} \left(u_J^{coarse} - u_J^{fine}\right)^2}}{\sqrt{\sum_{J}^{coarse\ nodes} \left(u_J^{fine}\right)^2}}.$$

This measure discards the information in the "exact" solution at nodes that do not coincide the coarse mesh nodes. Rather than just examining the difference between nodal values, a better approach would be to examine the difference in the continuous finite element solutions over the whole domain, i.e.

$$\text{Relative error} = \frac{\sqrt{\int_\Omega \left(\sum_J^{coarse\ nodes} u_J^{coarse} N_J^{coarse} - \sum_J^{fine\ nodes} u_J^{fine} N_J^{fine} \right)^2 d\Omega}}{\sqrt{\int_\Omega \left(\sum_J^{fine\ nodes} u_J^{fine} N_J^{fine} \right)^2 d\Omega}},$$

where N_J are the finite element basis functions introduced in Section 3.3.1. Although this second measure is better, the first approach for measuring the relative error is used in these tests due to its simplicity. Another issue concerning the measurement of the error is which norm should be used, e.g. an energy norm may be a better representation of the error. This issue is not addressed here and only the 2-norm is used.

Table 7.1 contains the results for simulations on the constant dispersion coefficient case with a 21×11 linear mesh. Recorded in this table are the size of the time-step used for that particular simulation, the maximum Courant and Peclet numbers that occurred during the simulation and, with both partial and full coupling, the total CPU time required for the simulation program and a measure of the error when compared to the exact solution.

Δt / sec	Co_{max}	Pe_{max}	CPU time[1]/ sec		Relative error	
			Partial	Full	Partial	Full
125	1.52	13.01	8.8	25.6	1.35×10^{-1}	4.82×10^{-1}
100	1.21	"	10.7	30.6	1.30×10^{-1}	4.57×10^{-1}
50	0.61	"	20.2	49.1	1.26×10^{-1}	4.29×10^{-1}
25	0.30	"	39.1	81.0	1.25×10^{-1}	4.24×10^{-1}
20	0.24	"	47.1	97.6	4.41×10^{-2}	1.50×10^{-1}
10	0.12	"	91.8	181.3	4.52×10^{-2}	1.53×10^{-1}
5	0.06	"	175.3	344.0	4.53×10^{-2}	1.53×10^{-1}
1	0.01	"	845.8	1263.0	4.57×10^{-2}	1.54×10^{-1}

Table 7.1: 21×11 mesh - constant dispersion coefficient

[1]These and all subsequent simulation timings were carried out using the SUN OS command `time` on a SPARCstation 1+.

The simulations using partial coupling are between 1.5 and 3 times faster than those using full coupling. This indicates that the full coupling iteration takes, on average, between 1.5 and 3 iterations to converge - a similar figure is noted in [30].

Surprisingly, partial coupling appears to give generally lower relative errors than full coupling for this problem. This is probably because the errors introduced by not performing full coupling counteract some of the error caused by using a coarser discretisation and looser tolerances than those used to obtain the "exact" solution.

The values in the Table 7.1 indicate that, with either coupling approach, it is pointless to use a small time-step as this does not drastically reduce the errors. This suggests that the temporal error is small compared to the other errors in the system. It is likely that the spatial error is the dominant source of error for the particular discretisation sizes used in space and time.

Table 7.2 contains the same information as Table 7.1, but for the velocity dependent dispersion coefficient Henry problem.

Δt / sec	Co_{max}	Pe_{max}	CPU time / sec		Relative error	
			Partial	Full	Partial	Full
125	1.52	6.36	8.7	27.1	1.52×10^{-1}	1.56×10^{-1}
100	1.21	″	10.4	30.8	1.52×10^{-1}	1.54×10^{-1}
50	0.61	″	19.5	49.2	1.52×10^{-1}	1.51×10^{-1}
25	0.30	″	37.3	89.4	1.51×10^{-1}	1.51×10^{-1}
20	0.24	″	46.5	96.7	1.51×10^{-1}	1.51×10^{-1}
10	0.12	″	88.9	179.7	1.51×10^{-1}	1.51×10^{-1}
5	0.06	″	176.5	338.8	1.51×10^{-1}	1.52×10^{-1}
1	0.01	″	845.3	1284.4	1.51×10^{-1}	1.52×10^{-1}

Table 7.2: 21×11 mesh - velocity dependent dispersion coefficient

The results in this table show the same trends as the results from the constant dispersion case. There is no noticeable loss of accuracy by using partial coupling and this approach is between 2 and 3 times faster for a given time step. Again, the error is not reduced as the time-step is refined so the use of a small time-step is pointless. As before, it is likely that spatial errors dominate.

The lack of variation in the relative errors suggest that the temporal component of the error is dominated by the other errors in the process for the range of time-steps investigated. Hence the optimum time-step for this problem on this mesh (i.e. the one which balances the temporal discretisation errors with the rest of the errors in the process) is greater than 125 seconds and corresponds to a maximum Courant number which is greater than unity. Thus an implicit discretisation scheme, as used here, is more appropriate for this problem than an explicit one due to its stability in this flow regime.

In order to investigate the role of the spatial errors, the simulation tests are repeated with a finer mesh. Tables 7.3 and 7.4 correspond to Tables 7.1 and 7.2 but on a 41×21 linear mesh.

Δt / sec	Co_{max}	Pe_{max}	CPU time / sec		Relative error	
			Partial	Full	Partial	Full
125	3.80	8.14	40.2	140.9	7.17×10^{-2}	8.49×10^{-2}
100	3.04	"	49.0	173.7	4.98×10^{-2}	6.32×10^{-2}
50	1.52	"	91.1	267.1	2.05×10^{-2}	2.68×10^{-2}
25	0.76	"	173.7	435.6	1.30×10^{-2}	1.46×10^{-2}
20	0.61	"	211.1	529.5	1.20×10^{-2}	1.30×10^{-2}
10	0.30	"	395.1	845.5	1.19×10^{-2}	1.21×10^{-2}
5	0.15	"	747.8	1524.8	1.25×10^{-2}	1.26×10^{-2}
1	0.03	"	3465.2	5318.5	1.33×10^{-2}	1.33×10^{-2}

Table 7.3: 41×21 mesh - constant dispersion coefficient

The pattern of the results is similar to that observed on the 21×11 mesh. The use of partial coupling does not noticeably degrade the accuracy of the results and leads to a solution 1.5 - 4 times quicker than full coupling.

The reduced spatial error on this finer mesh allows the variation in the temporal error as the time-step is refined to be observed. For these simulations, there is little point in using a time step of less than 50 seconds as the temporal errors are dominated by the other errors in the process. Hence there is no benefit to

Δt / sec	Co_{max}	Pe_{max}	CPU time / sec		Relative error	
			Partial	Full	Partial	Full
125	3.80	4.26	41.7	188.5	1.16×10^{-1}	6.02×10^{-2}
100	3.04	″	49.3	197.2	6.61×10^{-2}	4.27×10^{-2}
50	1.52	″	90.8	293.9	1.69×10^{-2}	1.99×10^{-2}
25	0.76	″	168.6	469.1	9.93×10^{-3}	1.13×10^{-2}
20	0.61	″	202.4	556.9	7.97×10^{-3}	8.84×10^{-3}
10	0.30	″	383.3	899.1	8.80×10^{-3}	8.91×10^{-3}
5	0.15	″	736.2	1545.0	9.31×10^{-3}	9.34×10^{-3}
1	0.03	″	3434.1	5550.1	9.92×10^{-3}	9.93×10^{-3}

Table 7.4: 41×21 mesh - velocity dependent dispersion coefficient

be gained by the extra computational work involved in the use of a smaller time-step. At $\Delta t = 50$ seconds, the maximum Courant number is 1.52 so, as with the 21×11 mesh, an implicit discretisation is more appropriate than an explicit one.

7.3 Concluding Remarks

The tests in this chapter suggest that partial coupling is a reasonable approach for this problem. For a particular size of time step, partial coupling is between 1.5 and 3 times faster than full coupling. The use of partial coupling does not lead to any noticeable degradation in the quality of the solution. Based on the success of partial coupling, it is reasonable to suggest that the next weakest coupling in the spectrum of approaches (Segol coupling) should be investigated for this problem. Such an investigation has not been carried out in this work.

However for more complicated flows, e.g. ones with more dynamically changing boundary conditions or systems in which the fluid properties are more sensitive to the contaminant concentration, the results on the coupling iteration approach may not be true.

As the optimal time step gives a maximum Courant number which exceeds unity in all the cases presented here, these results vindicate the use of a method that has no restriction on the size of the time-step for stability.

Chapter 8

Summary and Suggestions for Further Work

In this thesis, some of the implications of using an implicit Galerkin discretisation approach in the numerical solution of the governing equations for contaminant transport in porous media have been examined.

Conceptually this is a relatively simple discretisation approach. Theoretically, it is unconditionally stable and second order accurate in both space and time. In Chapter 4, this discretisation was shown both qualitatively and quantitatively to be in practical agreement with theory. The problem of unphysical extrema in the approximate solution which is caused by the discretisation used was highlighted. These extrema can have disastrous consequences if the resulting solution is used to drive a chemical or physical process in the model. Some techniques for the control of unphysical extrema (including mesh refinement and flux corrected transport) were given and demonstrated to be effective.

The results on the Henry problem test cases in Chapter 4 were shown qualitatively to be in agreement with those obtained by other authors who used different discretisation approaches, e.g. Lagrangian and mixed explicit-implicit methods.

Most of these other approaches generate linear systems with symmetric positive definite matrices for which an established cheap and reliable method of solution is available - namely the preconditioned conjugate gradient method. The discretisation method used in this work has notable positive features (e.g. simplicity, accuracy, stability), but it generates a non-symmetric matrix; there is no established equivalent of PCG for non-symmetric matrices. However, due to recent developments in applied linear algebra, there are some solvers available for non-symmetric systems which appear to be just as effective without possessing all the benefits of PCG. Examples of these are Bi-CGSTAB and GMRES.

The original purpose of this thesis was to re-appraise the non-symmetric approach in the light of these solvers and compare this approach with some of the techniques which give symmetric matrices for all aspects of the solution process e.g. accuracy, stability, memory requirements, CPU time. However, in the course of the study on the non-symmetric solvers, a problem with the robustness of Bi-CGSTAB was un-covered. At this stage, the purpose of the work became one of investigating this problem.

The cause of the lack of robustness was identified as being rounding errors corrupting sensitive values. The accumulation of these rounding errors spoils the underlying recursion process. These sensitive values are generated because the Bi-CGSTAB method is not capable of representing the eigenvalues of the matrix when these have a significant imaginary part.

Methods for overcoming the robustness problem were given and tested and two were found to be effective. The first of these involves restarting the iteration process at points indicated by various monitor functions and tolerances, which has the effect of discarding rounding errors in the process so that they do not accumulate and lead to convergence difficulties. The second remedy avoids the generation of the sensitive values by modifying the underlying recursion process so that it can generate complex eigenvalues. This second approach, known as Bi-CGSTAB(2), is preferred here because, unlike the method of restarting, it does not require any external parameters and treats the cause, not the symptom. In Section 6.5, the Bi-CGSTAB(2) method was demonstrated to give convergence histories that are smoother than those of Bi-CGSTAB. It was also shown to be at least as efficient in practice.

Other possible remedies for the problem with the robustness of Bi-CGSTAB include partial orthogonalisation and the use of a quasi-minimal residual type approach. These methods were outlined in Chapter 6 but they were not investigated. The deficiencies of the solver these methods concentrate on suggest that they show enough promise to warrant investigation in the context of Bi-CGSTAB robustness.

As the robustness of Bi-CGSTAB is being discussed, it should be noted that even with Bi-CGSTAB(2), the problem of a fatal or near-fatal breakdown of the underlying Lanczos process is not addressed. For the solver to be truly robust, a look-ahead Lanczos approach must also be incorporated as a contingency against such a breakdown (and even this method fails in the very special case of an “incurable breakdown” [80]).

The tests on the performance of the linear solvers used for each of the governing equations suggest the use of the following. The $ILDL^T$ preconditioned conjugate gradient method for the solution of the discrete fluid continuity equation, the diagonally preconditioned conjugate gradient method for the discrete Darcy velocity vector component equations, and the $ILDM^T$ preconditioned Bi-CGSTAB(2) method for the discrete contaminant mass balance equation.

A breakdown of the CPU time typically spent in each of these linear solvers over a single step of the coupling iteration is shown in Table 8.1.

Origin of linear system	Time in solver / sec
Fluid continuity equation	0.40
Darcy x-component equation	0.14
Darcy z-component equation	0.14
Contaminant mass balance equation	0.12

Table 8.1: Breakdown of time in solvers over a single step of a coupling iteration

From the results in this table, it can be seen that the current bottle-neck in the whole process is the solution of the stiffness matrix system arising from the discretisation of the fluid continuity equation. The solver for this system consumes as much CPU time as the total for the solvers for the other three systems. This suggests that a more powerful approach is required for the discrete fluid continuity equation e.g. a more effective preconditioner or a fast elliptic solver. A candidate for the fast elliptic solver is multigrid which has already been used successfully on problems from flow in porous media, e.g. [71, 81].

The governing equations for the fluid and the contaminant cannot be solved separately due to the dependency of the fluid density on the contaminant concentration. In this thesis, a coupling iteration is used to allow the (inter-linked) governing equations to be solved individually. Since the dependency of the fluid density on the contaminant concentration is relatively weak, it is possible to use a partial coupling approach which does not iterate between the equations to couple them within a time step. This approach was compared with full coupling in Chapter 7 and shown quantitatively to give results which show no appreciable loss of accuracy but are obtained at approximately a quarter to a half of the computational expense of the fully coupled solution.

There is no theory given for the convergence of the coupling iteration in this work - such theory needs to be provided before the method can be considered fully trustworthy.

In order to make the work on the effectiveness of preconditioners in Chapters 5 and 6 more applicable to practical problems involving flows in porous media, the investigations on preconditioners needs to be extended to include problems where, due to impermeable formations in the porous medium for instance, there are rapidly changing coefficients in the governing equations.

Another aspect of practical problems involving flows in porous media is that the unsaturated region is of interest also. Hence, the mathematical model should be extended to include unsaturated (and combined saturated-unsaturated) flows. The extension of the model is a relatively simple step - the porosity is replaced by a moisture content variable. The value of this is bounded between zero and the porosity, and it varies with the fluid pressure. However, the introduction of this variable results in highly non-linear equations for the fluid, making the solution procedure more difficult and the requirement for a fast solver for the discrete fluid continuity equation even more of a necessity. The effect of the presence of this extra flow-dependent variable on the coupling iteration would have to be determined by performing tests similar to those in Chapter 7.

Apart from the further work already suggested in this chapter, it is noted that a comparison with the symmetric approach is still needed to determine if the non-symmetric discretisation is a viable alternative.

References

[1] M. Arioli, I. Duff, and D. Ruiz. "Stopping criteria for iterative solvers". Technical Report RAL-91-057, Rutherford Appleton Laboratory, 1991.

[2] S.F. Ashby, M.J. Holst, T.A. Manteuffel, and P.E. Saylor. "The role of the inner product in stopping criteria for conjugate gradient iterations". Technical Report UCRL-JC-112586, Numerical Mathematics Group, Computing and Mathematics Research Division, Lawrence Livermore National Laboratory, 1992.

[3] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1994.

[4] O. Axelsson and G. Lindskog. "On the rate of convergence of the preconditioned conjugate gradient method". *Numer. Math.*, **48**:499–523, 1986.

[5] R. Barrett et al. *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods.* SIAM, 1993.

[6] J. Bear and A. Verruijt. *Modeling Groundwater Flow and Pollution.* Reidel, Holland, 1987.

[7] A.A. Becker. *The Boundary Element Method in Engineering : A Complete Course.* McGraw-Hill, 1992.

[8] J.G. Blom, J.G. Verwer, and R.A. Trompert. "A comparison between direct and iterative methods to solve the linear systems arising from a time-dependent 2D groundwater flow model". Technical Report NM-R9205, CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, 1992.

[9] J.P. Boris and D.L. Book. "Flux-corrected transport. I SHASTA, a fluid transport algorithm that works". *J. Comp. Phys.*, **11**:38–69, 1973.

[10] L. Brusa, G. Gentile, L. Nigro, D. Mezzani, and R. Rangogni. "A 3-D finite element code for modelling salt intrusion in aquifers". In G. Gambolati, A. Rinaldo, C.A. Brebbia, W.G. Gray, and G.F. Pinder, editors, *Computational Methods in Subsurface Hydrology,* Proc. 8th Int. Conf. on Comp. Meth. in Water Res., Venice, pages 335–340. Springer-Verlag, June 1990.

[11] G. Brussino and V. Sonnad. "A comparison of direct and iterative techniques for sparse, unsymmetric systems of linear equations". *Int. J. Numer. Meth. Engng.*, **28**:801–815, 1989.

[12] F.F. Campos,filho and J.S. Rollet. "Study of convergence behaviour of conjugate gradient-type methods for solving unsymmetric systems". OUCL Report 92/24, Oxford University Computing Laboratory, 1992.

[13] F.F. Campos,filho and J.S. Rollet. "The use of partial orthogonalisation for solving large sparse linear systems". OUCL Report 93/11, Oxford University Computing Laboratory, 1993.

[14] S. Chandler. "Using Green's functions to improve conjugate gradient methods for the semi-conductor equations". Technical Report AM-93-04, School of Mathematics, University of Bristol, 1993.

[15] P.M. Cohn. *Algebra.* Wiley, 2nd edition, 1982.

[16] J. Cullum and A. Greenbaum. "Residual relationships within three pairs of iterative algorithms for solving $Ax = b$". CS Tech. Rept. 623, Courant Institute, New York, February 1993.

[17] E. Custodio and A. Galofré, editors. *Study and Modelling of Saltwater Intrusion into Aquifers.* CIMNE, Barcelona, February 1993. (Proceedings of the Saltwater Intrusion Meeting, 1-6 November 1992, Barcelona, Spain).

[18] A.D. Daus, E.O. Frind, and E.A. Sudicky. "Comparative error analysis in finite element formulations of the advection-dispersion equation". *Advances in Water Resources*, **8**:86–95, June 1985.

[19] A.J. Davies. *The Finite Element Method : A First Approach.* Oxford Applied Mathematics and Computing Science Series. Clarendon Press, Oxford, 1980.

[20] J. Donea. "A Taylor-Galerkin method for convective transport problems". *Int. J. Num. Meths. Engng.*, **20**:101–119, 1984.

[21] J. Donea, S. Giuliani, H. Laval, and L. Quartapelle. "Time-accurate solution of advection-diffusion problems by finite elements". *Comp. Meths. App. Mech. Engng.*, **45**:123–145, 1984.

[22] I. Duff and J. Reid. *ACM Trans. Math. Software*, **5**:18–35, 1979.

[23] I. Duff and J. Reid. *SIAM J. Sci. Stat. Comput.*, **5**:633–641, 1984.

[24] V. Faber and T. Manteuffel. "Necessary and sufficient conditions for the existence of a conjugate gradient method". *SIAM J. Numer. Anal.*, **21**(2):352–362, April 1984.

[25] R. Fletcher. "Conjugate gradient methods for indefinite systems". *Lecture Notes in Math.*, **506**:73–89, 1976.

[26] R.W. Freund, G.H. Golub, and N.M. Nachtigal. "Iterative solution of linear systems". *Acta Numerica*, pages 57–100, 1991.

[27] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal. "An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices". Technical Report 91.09, RIACS, NASA Ames Research Center, Moffet Field, 1991.

[28] R.W. Freund and N.M. Nachtigal. "QMR : A quasi-minimal residual method for non-hermitian linear systems". *Numer. Math.*, **60**:315–339, 1991.

[29] E.O. Frind. "Simulation of long-term transient density-dependent transport in groundwater". *Adv. Water. Res.*, pages 73–88, 1982.

[30] G. Galeati, G. Gambolati, and S.P. Neuman. "Coupled and partially coupled Eulerian-Lagrangian model of freshwater-seawater mixing. *Water Res. Res.*, **28**(1):149–165, January 1992.

[31] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1st edition, 1983.

[32] A. Greenbaum. "Behaviour of slightly perturbed Lanczos and conjugate-gradient recurrences". *Lin. Alg. Appl.*, **113**:7–63, February 1989.

[33] A. Greenbaum. "Accuracy of computed solutions from conjugate-gradient-like methods". In M. Natori and T. Nodera, editors, *Advances in Numerical Methods for Large Sparse Sets of Linear Equations,* Proc. PCG Meeting '94, Keio University, Yokohama, Japan, pages 126–138, 1994.

[34] A. Greenbaum and Z. Strakos. "Predicting the behaviour of finite precision Lanczos and conjugate gradient computations". *SIAM J. Matrix Anal. Appl.*, **13**(1):121–137, January 1992.

[35] A. Greenbaum and L.N. Trefethen. "GMRES/CR and Arnoldi/Lanczos as matrix approximation problems". *SIAM J. Sci. Comput.*, **15**(2):359–368, March 1994.

[36] M.H. Gutknecht. "Variants of BiCGSTAB for matrices with complex spectrum". *SIAM J. Sci. Stat. Comput.*, **14**(5):1020–1033, 1993.

[37] L.A. Hageman and D.M. Young. *Applied Iterative Methods.* Academic Press, 1981.

[38] H.R. Henry. "Effects of dispersion on salt encroachment in coastal aquifers". Water Supply Paper 1613-C:C71-C84, U.S. Geol. Survey, 1964.

[39] M.R. Hestenes and E. Stiefel. "Methods of conjugate gradients for solving linear systems". *J. Res. Natl. Bur. Stand.*, **49**:409–436, 1952.

[40] P.S. Huyakorn, P.F. Andersen, J.W. Mercer, and H.O. White. "Saltwater intrusion in aquifers : Development and testing of a three-dimensional finite element model". *Water Res. Res.*, **23**(2):293–312, 1987.

[41] A. Jameson. "Acceleration of transonic potential flow calculations on arbitrary meshes by the multiple grid method". Technical Report 79-1458, AIAA, New York, 1979.

[42] A. Jameson, W. Schmidt, and E. Turkel. “Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time stepping”. Technical Report 81-1259, AIAA, 1981.

[43] O.G. Johnson, C.A. Micchelli, and G. Paul. “Polynomial preconditioners for conjugate gradient calculations”. *SIAM J. Numer. Anal.*, **20**(2):362–376, April 1983.

[44] P. Joly and R. Eymard. “Preconditioned bi-conjugate gradient methods for numerical reservoir simulation”. *J. Comp. Phys.*, **91**:298–309, 1990.

[45] W. Joubert. *Generalized Conjugate Gradient and Lanczos Methods for the Solution of Nonsymmetric Systems of Linear Equations.* PhD thesis, Center for Numerical Analysis, University of Texas, Austin, TX, January 1990.

[46] W. Joubert. “Lanczos methods for the solution of nonsymmetric systems of linear equations”. *SIAM J. Matrix Anal. Appl.*, **13**(3):926–943, July 1992.

[47] D.W. Kelly, S. Nakazawa, O.C. Zienkiewicz, and J.C. Heinrich. “A note on upwinding and anisotropic balancing dissipation in finite element approximations to convective diffusion problems”. *Int. J. Numer. Meth. Engng.*, **15**:1705–1711, 1980.

[48] D.S. Kershaw. ”The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations”. *J. Comp. Phys.*, **26**:43–65, 1978.

[49] C. Lanczos. “Solution of systems of linear equations by minimized iterations”. *J. Res. Natl. Bur. Stand.*, **45**:255–282, 1952.

[50] H.M. Leismann and E.O. Frind. “A symmetric-matrix time integration scheme for the efficient solution of advection-dispersion problems”. *Water Res. Res.*, **25**(6):1133–1139, 1989.

[51] T.A. Manteuffel. “An incomplete factorization technique for positive definite linear systems”. *Math. Comp.*, **34**(150):473–497, April 1980.

[52] B.S. Massey. *Mechanics of Fluids.* Van Nostrand Reinhold, 5th edition, 1983.

[53] J.A. Meijerink and H.A. van der Vorst. "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix". *Math. Comp.*, **137**(6):148–162, January 1977.

[54] J.A. Meijerink and H.A. van der Vorst. "Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems". *JCP*, **44**:134–155, 1981.

[55] N.M. Nachtigal, S.C. Reddy, and L.N. Trefethen. "How fast are nonsymmetric matrix iterations ?". *SIAM J. Matrix Anal. Appl.*, **13**(3):778–795, July 1992.

[56] K.J. Neylon. "Application of the Taylor-Galerkin method to transport problems in subsurface hydrology". Numerical Analysis Report 5/93, Maths. Dept., Univ. of Reading, 1993.

[57] Y. Notay. "On the convergence of the conjugate gradients in presence of rounding errors". *Numer. Math.*, **65**:301–317, 1993.

[58] A. Ogata and R.B. Banks. "A solution of the differential equation of longitudinal dispersion in porous media". Professional Paper 411-A, U.S. Geol. Survey, 1961.

[59] C.C. Paige and M.A. Saunders. "Solution of sparse indefinite systems of linear equations". *SIAM J. Numer. Anal.*, **12**(4):617–629, September 1975.

[60] B.N. Parlett, D.R. Taylor, and Z.A. Liu. "A look-ahead Lanczos algorithm for unsymmetric matrices". *Math. Comp.*, **44**(169):105–124, January 1985.

[61] A. Peters. "CG-like algorithms for linear systems stemming from the FE discretization of the advection-dispersion equation". In T.F. Russell, R.E. Ewing, C.A. Brebbia, W.G. Gray, and G.F. Pinder, editors, *Numerical Methods in Water Resources,* Proc. 9th Int. Conf. on Comp. Meth. in Water Res., University of Colorado, Denver, USA, pages 511–518, June 1992.

[62] A. Peters, M. Eiermann, and H. Daniels. "Symmetric versus non-symmetric matrix techniques : Implementation of two Galerkin-FE approaches for the advection-dispersion equation". In *Proc. 2nd Int. Conf. on Applications of Supercomputers in Engineering,* MIT Cambridge, USA, pages 387–402, August 1991.

[63] G.F. Pinder and H.H. Cooper Jr. "A numerical technique for calculating the transient position of the saltwater front". *Water Res. Res.*, **6**(3):875–882, 1970.

[64] G.F. Pinder and S. Stothoff. "Can the sharp interface salt-water model capture transient behaviour ?". In Celia et al., editors, *Computational Methods in Water Resources,* Vol. 1, Proc. 7th Int. Conf. on Comp. Meth. in Water Res., June 1988, pages 217–222, July 1988.

[65] A. Priestley. "A quasi-conservative version of the semi-Lagrangian advection scheme". *Monthly Weather Review*, **121**:621–629, 1993.

[66] G. Radicati, Y. Robert, and S. Succi. "Iterative algorithms for the solution of nonsymmetric systems in the modelling of weak plasma turbulence". *J. Comp. Phys.*, **80**:489–497, 1989.

[67] J.K. Reid. "On the method of conjugate gradients for the solution of large sparse systems of linear equations". In J.K. Reid, editor, *Large Sparse Sets of Linear Equations*, pages 231–254, 1971.

[68] R.D. Richtmyer and K.W. Morton. *Difference Methods for Initial-Value Problems.* Number 4 in Tracts in Mathematics. Interscience, 2nd edition, 1967.

[69] Y. Saad. "Krylov subspace methods on supercomputers". *SIAM J. Sci. Stat. Comput.*, **10**(6):1200–1232, 1989.

[70] Y. Saad and M.H. Schultz. "GMRES : A generalized minimum residual algorithm for solving non-symmetric linear systems". *SIAM J. Sci. Stat. Comput.*, **7**(3):856–869, 1986.

[71] G.H. Schmidt and E. de. Sturler. "Multigrid for porous medium flow on locally refined three dimensional grids". Technical Report 1014, Shell Research B.V., Rijswijk, The Netherlands, December 1990. (Also presented at the Third European Conference on Multigrid Methods, Oct. 1-4, 1990, Bonn, Germany).

[72] G. Segol, G.F. Pinder, and W.G. Gray. "A Galerkin-finite element technique for calculating the transient position of the saltwater front". *Water Res. Res.*, **6**(3):875–882, 1975.

[73] R.B. Simpson. "Testing for effects of asymmetry and instability on preconditioned iterations of conjugate gradient type". *IMA J. Numer. Anal.*, **14**(1):1–25, 1993.

[74] G.L.G. Sleijpen and D.R. Fokkema. "BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum". *Electronic Transactions on Numerical Analysis*, **1**:11–32, 1993.

[75] G.D. Smith. *Numerical Solution of Partial Differential Equations : Finite Difference Methods.* Oxford Applied Mathematics and Computing Science Series. Clarendon Press, Oxford, 3rd edition, 1985.

[76] P. Sonneveld. "CG-S : A fast Lanczos-type solver for non-symmetric linear systems". *SIAM J. Sci. Stat. Comput.*, **10**(1):36–52, 1989. (Also Report 84-16, Dept. of Math., Delft University, The Netherlands, 1984).

[77] Z. Strakos. "On the real convergence rate of the conjugate gradient method". *Lin. Alg. Appl.*, **154**:535–549, August 1991.

[78] G. Strang and G.L. Fix. *An Analysis of the Finite Element Method.* Series in Automatic Computation. Prentice-Hall, 1973.

[79] E. Süli and A.F. Ware. "A spectral method of characteristics for first-order hyperbolic equations". *SIAM J. Numer. Anal.*, **28**(2):423–445, 1991.

[80] D.R. Taylor. *Analysis of the look ahead Lanczos algorithm.* PhD thesis, Department of Mathematics, University of California, Berkeley, CA, November 1982.

[81] R. Teigland. *On Multilevel Methods for Numerical Reservoir Simulation.* PhD thesis, Department of Mathematics, University of Bergen, July 1991.

[82] R.A. Trompert. "A note on singularities caused by the hydrodynamic dispersion tensor". Technical Report NM-R9302, Department of Numerical Mathematics, CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, January 1993.

[83] A. van der Sluis and H.A. van der Vorst. "The rate of convergence of conjugate gradients". *Numer. Math.*, **48**:543–560, 1986.

[84] H.A. van der Vorst. "BI-CGSTAB : A fast and smoothly convergent variant of BI-CG for the solution of nonsymmetric linear systems". *SIAM J. Sci. Stat. Comput.*, **13**(2):631–644, 1992.

[85] M. Th. van Genuchten. "On the accuracy and efficiency of several numerical schemes for solving the convective-dispersive equation". In W.G. Gray, G.F. Pinder, and C.A. Brebbia, editors, *Finite Elements in Water Resources,* Proc. 1st Int. Conf. on Finite Elements in Water Res., Princeton, USA, July 1976, pages 1.71–1.90, London, 1977. Pentech.

[86] M. Th. van Genuchten and W.G. Gray. "Analysis of some dispersion corrected numerical schemes for solution of the transport equation". *Int. J. Numer. Meth. Engng.*, **12**:387–404, 1978.

[87] R.S. Varga. *Matrix Iterative Analysis.* Prentice-Hall International, London, 1962.

[88] C.I Voss and W.R. Souza. "Variable density flow and solute transport simulation of regional aquifers containing a narrow freshwater-saltwater transition zone". *Water Res. Res.*, **23**(10):1851–1866, 1987.

[89] A.J. Wathen. "Realistic eigenvalue bounds for the Galerkin mass matrix". *IMA J. Numer. Anal.*, **7**:449–457, 1987.

[90] R. Weiss and W. Schönauer. "Accelerating generalized conjugate gradient methods by smoothing". In *Proceedings of IMACS '91, Brussels*, 1991.

[91] R.S. Wikramaratna and W.L. Wood. "Control of spurious oscillations in the salt water intrusion problem". *Int. J. Numer. Meth. Engng.*, **19**:1243–1251, 1983.

[92] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[93] G-T. Yeh. "On the computation of Darcian velocity and mass balance in the finite element modeling of groundwater flow". *Water Res. Res.*, **17**(5):1529–1534, October 1981.

[94] S.T. Zalesak. "Fully multidimensional flux-corrected transport algorithms for fluids". *J. Comp. Phys.*, **31**:335–362, 1979.

[95] L. Zhou and H.F. Walker. "Residual smoothing techniques for iterative methods". Research report, Dept. of Maths. and Stats., Utah State University, December 1992.

[96] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method. Volume 1 : Basic Formulation and Linear Problems.* McGraw-Hill, 4th edition, 1989.