UNI ERSITY 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

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.

tural 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 f r, and Requirements f, the Mathematical M del

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 quan

1. Mathematical M dels f Saline Intrusi n

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.

ariable density transport models are well documented in the literature. They essentially consist of four main components : a *uid 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, *arcy'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 S luti n f the G verning Equati ns

sation methods are described in Section 3.3, and discrete forms of the governing equations for saline intrusion are given in Section 3.4.

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], an

ematical 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 Meth ds

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 $i = k+1, \ldots, n$
    =

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 br=^^Tareation^^TD=TD=^Tc^Tcreqnmif^^TD=

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 Meth ds

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*

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

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	$L+U$	

- 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. Kryl v Subspace Meth ds

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

$$\begin{aligned} A^{-1} &= \frac{-1}{c_n}(c_{n-1}I + c_{n-2}A + \ldots + c_2 A^{n-3} + c_1 A^{n-2} + A^{n-1}) \\ &= \tilde{c}_0 I + \tilde{c}_1 A + \ldots + \tilde{c}_{n-3} A^{n-3} + \tilde{c}_{n-2} A^{n-2} + \tilde{c}_{n-1} A^{n-1}. \end{aligned} \tag{2.6}$$

Hence

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

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),

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

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 $\quad _i(A, \boldsymbol{r}) \supset \quad _{i-1}(A, \boldsymbol{r})$. If d is the smallest integer such that

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

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

$$\dim\{\quad _i(A, \boldsymbol{r})\} = \begin{cases} i & \text{for} \quad i \leq d \\ d & \text{for} \quad i \geq d. \end{cases}$$

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

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

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 $\quad_d(A, \boldsymbol{r})$ 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,

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

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

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

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

with $\quad_i(0) = 1$.

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

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

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

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

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

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

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 $\|\boldsymbol{w} \in$

vector-vector products and 3 SAXPYs[2].

Convergence Criteria

The term "until satisfied" appears in Algorithm 2.2 (and all subsequent iterative

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

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

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

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) ov

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 $\quad$. Pre-multiplying the system in this way is termed "preconditioning from the left",

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

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

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

Only preconditioning from the left is used in this thesis. The ideal preconditioning matrix gives $\quad ^{-1}A$ to be the identity matrix (since $\kappa_2(I) = 1$), in which case, $\quad ^{-1} = A^{-1}$. 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, $\quad ^{-1}$ 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

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

are solved at each iteration to produce $\boldsymbol{u} = \quad ^{-1}\boldsymbol{v}$. 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, $\quad \in \mathbb{R}^{n\times n}$, 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

$\quad \boldsymbol{z}_{i-1} := \quad ^{-1}\boldsymbol{r}_{i-1}$

$\quad \beta_k := (\boldsymbol{z}_{i-1}^T \boldsymbol{r}_{i-1})/(\boldsymbol{z}_{i-2}^T \boldsymbol{r}_{i-2}) \qquad (\beta_1 := 0)$

$\quad \boldsymbol{p}_i := \boldsymbol{z}_{i-1} + \beta_i \boldsymbol{p}_{i-1} \qquad (\boldsymbol{p}_1 := \boldsymbol{r}_0)$

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

$\boldsymbol{x}_i := \boldsymbol{x}_{i-1} + \alpha \boldsymbol{p}_i$

$\boldsymbol{r}_i := \boldsymbol{r}_{i-1} - \alpha A \boldsymbol{p}_i$

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 givenarer

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, ^{-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 $p = 1, \ldots$

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]. □

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 $\quad_k(A, \boldsymbol{r}_0)$ 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

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

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

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

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, $\boldsymbol{x}_0 \in \mathbb{R}^n$

$\hat{\boldsymbol{v}}_{j+1} := A\boldsymbol{v}_j - \sum_{i=1}^{j} h_{i,j}\boldsymbol{v}_i$

$h_{j+1,j} := \|\hat{\boldsymbol{v}}_{j+1}\|_2$

$\boldsymbol{v}_{j+1} := \hat{\boldsymbol{v}}_{j+1}/h_{j+1,j}$

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 $_k(A, \boldsymbol{r}_0)$ is that all the vectors in the basis m

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 $\quad_m(A, \boldsymbol{r}_0)$. 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.

$\boldsymbol{r}_0 :=$

$\boldsymbol{x}_m := \boldsymbol{x}_0 + V_k \boldsymbol{y}_m$ 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,

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

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

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

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 tw

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

$\beta := \rho_i / \rho_{i-1}$

$\boldsymbol{u} = \boldsymbol{r}_{i-1} + \beta \boldsymbol{q}$

$\boldsymbol{p} := \boldsymbol{u} + \beta(\boldsymbol{q} + \beta \boldsymbol{p})$

$\boldsymbol{y} := \quad^{-1}\boldsymbol{p}$

$\boldsymbol{v} := A\boldsymbol{y}$

$\alpha := \rho_i / (\hat{\boldsymbol{r}}_0^T \boldsymbol{v})$

$\boldsymbol{q} := \boldsymbol{u} - \alpha \boldsymbol{v}$

$\boldsymbol{z} := \quad^{-1}(\boldsymbol{u} + \boldsymbol{q})$

$\boldsymbol{x}_i := \boldsymbol{x}_{i-1} + \alpha \boldsymbol{z}$

$\boldsymbol{r}_i := \boldsymbol{r}_{i-1} - \alpha A \boldsymbol{z}$

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, $\tilde{\varphi}_i(\cdot$

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

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.

.1 The G verning Equati ns

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 as iha major test case in this thesis. From this poin

variations is negligible.

From [6], a mathematical model for the t

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

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

where $\underline{\boldsymbol{D}} = \underline{\boldsymbol{D}}(\boldsymbol{q})$ is the dispersion tensor ($\mathrm{L^2T^{-1}}$). 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}/\ $), by

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

where α_T is the transverse dispersivity (L)

α_L is the longitudinal dispersivity (L)

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

$\delta_{ij} = 1$ 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 $\boldsymbol{\nabla}.\ \underline{\boldsymbol{D}}\boldsymbol{\nabla}(\rho c)$ 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.

.2 C upling f the G verning Equati ns

Due to the nature of the interaction between the fluid density (ρ

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.

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

$(x_1 \quad _1$

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

$$= \mathrm{O}(\Delta x).\mathrm{TcffO}(\Delta\mathrm{ffO}(\Delta\mathrm{ffabce},\Gamma\mathrm{rtheГrtruncationTfferrorГrtendsTfftoГTfffDffГTcwГTfff$$

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 co

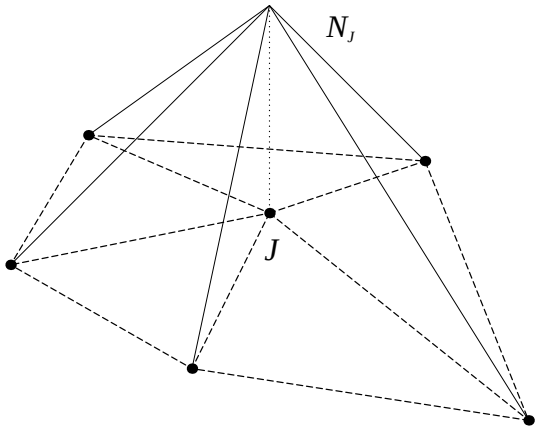


Figure 3.1: A linear basis function at node J in a 2-D mesh of triangles

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

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

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

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

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,

$$\int_\Omega (v\boldsymbol{\nabla}.\boldsymbol{\nabla}w + \boldsymbol{\nabla}v.\boldsymbol{\nabla}w)d \quad = \int_\Gamma v\boldsymbol{\nabla}w.\boldsymbol{n} \qquad \textbf{\textit{aanDcaaanaaa}}$$

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

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

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 $\int_\Omega \delta_I d \quad = 1$. This choice results in the *collocation* method.

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

  $$\omega_I = N_I(x, TDδTjδTTfTJδTDδ,nδTTcδTjδToTfδeρne\ aTDδhoiceTjδn$$

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

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

$$\frac{U_J^{n+1} - U_J^n}{\Delta t} + (1 - \theta)$$

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

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 poin

.4 Discretisati n f the G verning Equati ns

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, , i.e.

$$\int_{\Omega} \left(\frac{\partial \rho}{} \right.$$

(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

$$K^{n+1}\boldsymbol{p}^{n+1} = g\boldsymbol{F}^{n+1} - \quad gt$$

3.4.2

3.4.3

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

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

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,

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

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),

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

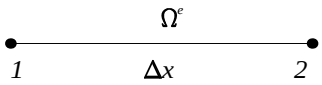
The physical Courant and Peclet numbers are not used in this work. Instead, Co and Pe are used to indicate the discrete versions of these numbers (e.g. Co indicates the proportion of the discrete advection term to the discrete inertia term). In these discrete versions, the representative length is the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

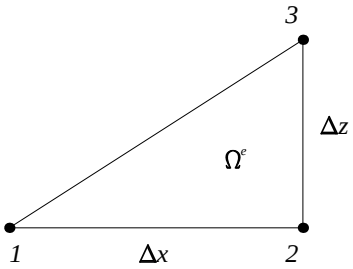
The origin of the non-symmetry is the advection term. This is the term which contains first spatial derivatives of the solution variable. A term of this form is not present in either the fluid continuity equation (when solved for p) or Darcy's law (when solved for $\boldsymbol{q}$).

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).



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

(a) Linear 1-D



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

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

sation 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

$$M = \{M_{IJ}\}_{I,J=1,\ldots,nodes} \qquad \boldsymbol{c}^i = \{c^i_J\}_{J=1,\ldots,nodes}$$

$$V^i = \{V^i_{IJ}\}_{I,J}$$

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 then

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 T

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 Δz

removing or controlling them are described.

4.1.3 Unphysical Oscillations and their Control

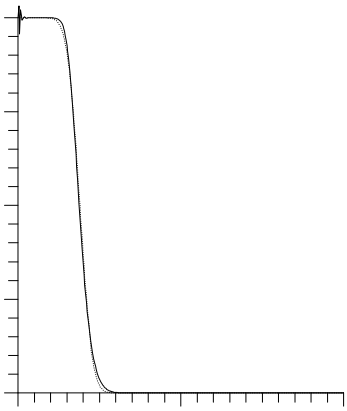
The steady-state form of the 1-D advection-diffusion equation (4.1) is

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

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 ($Co \leq 1$, $Pe \leq$



solution.

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

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

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 a aTTfiff.)Tjfffe(iTjiffTffs

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.

Case I: 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,

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

Case II: 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,

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

Case III: 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 b

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

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

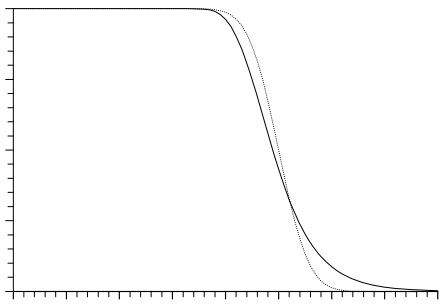
where $d_J^n=c_{J+1}^n-c_J^n$. The matrix system for the differences is

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

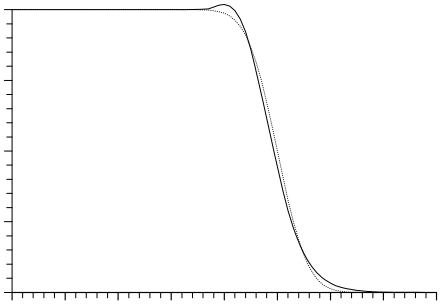
where typically, the J^{th} equation, with $Co=v_z\frac{\Delta t}{\Delta z}$ and $\frac{Co}{Pe}=D_{zz}\frac{\Delta t}{(\Delta z)^2}$, is

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

Scheme	



(a) Low order scheme



(b) High order scheme

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

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.

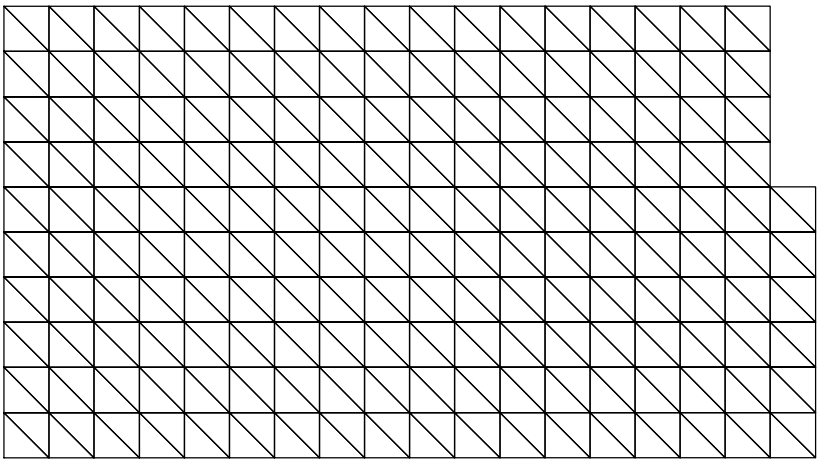
In the second method, the position of z_1 is determined dynamically (i.e. while the time-stepping is being performed) according to

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

where q_x is the horizontal componen

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

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



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,

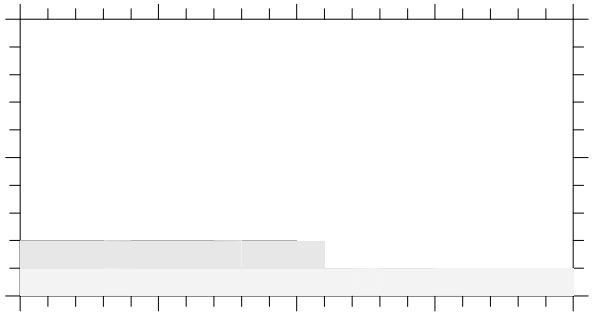
$$\Delta t_{i+1} = \min(1.2\Delta t_i, 600\text{s}) \qquad \Delta t_0 = 15\text{s}$$

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.

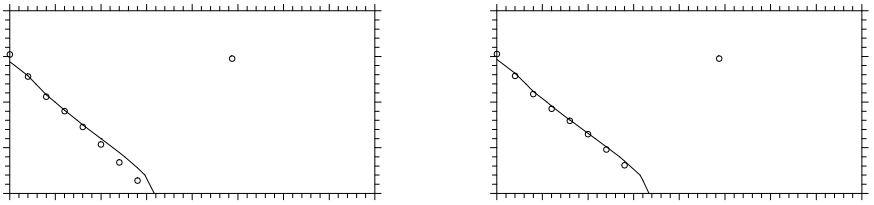
As it is the properties of the discretisations that are being examined in this chapter, the tolerance used in the coupling convergence criterion (Step 8 of the coupling iteration given in Algorithm 3.2) is set impractically low. The purpose of this is to remove any errors the iteration introduces. The 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 discretisations described in Chapter 3 is examined for the constant dispersion case.

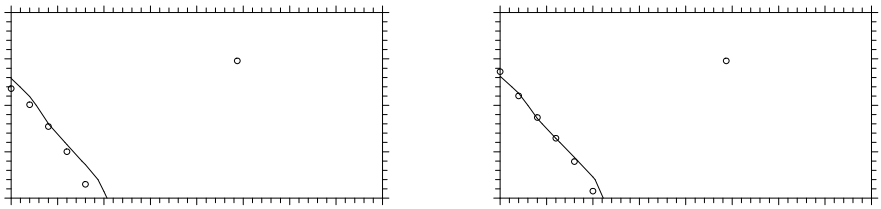


Comparisons of the position of the 0.5 isochlor at various times with results from [29, 72] are shown in Figures 4.9 and 4.10 - there is good agreement, showing the transient accuracy of the whole discretisation approach for this problem.

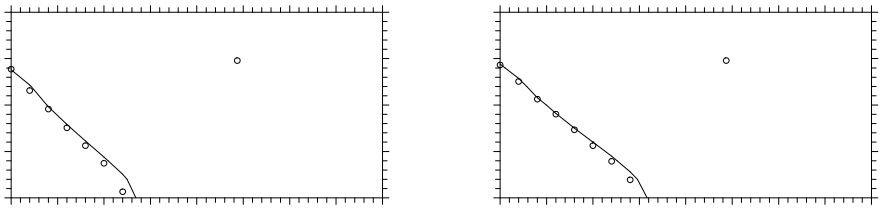


(i) t=100 min. (ii) At equilibrium (t = 300 min.)

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



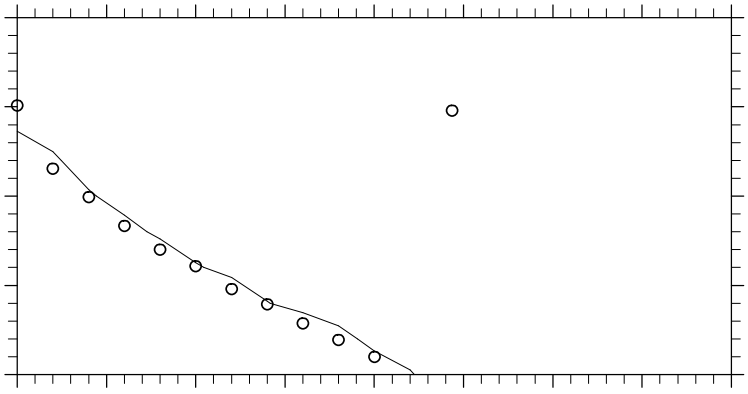
(i) t=25 min. (ii) t=30 min.



(iii) t

4.2.3 Results for 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]

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 De nite 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. ariations 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 fr m Fluid C ntinuity Equati n

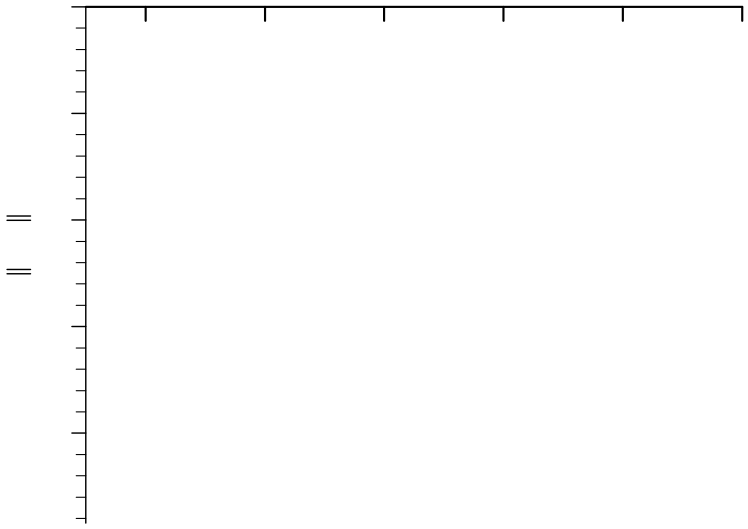
The matrix in the discrete fluid continuity equation (3.19) is the finite element stiffness matrix. This is symmetric and semi-definite but, with the imposition of the physical boundary conditions associated with the problem, is SPD. It is therefore a candidate for solution by CG.

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

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

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



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 multiplic

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.

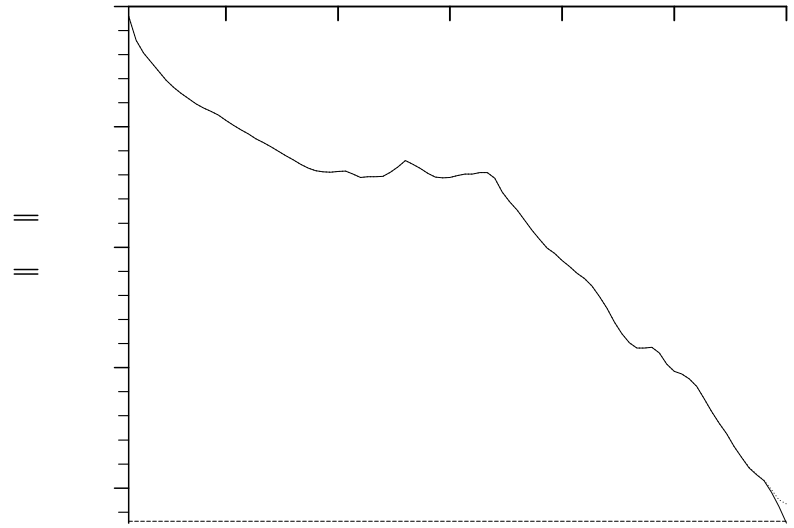


Figure 5.3: Iteration history for CG solver with high tolerance

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.

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.

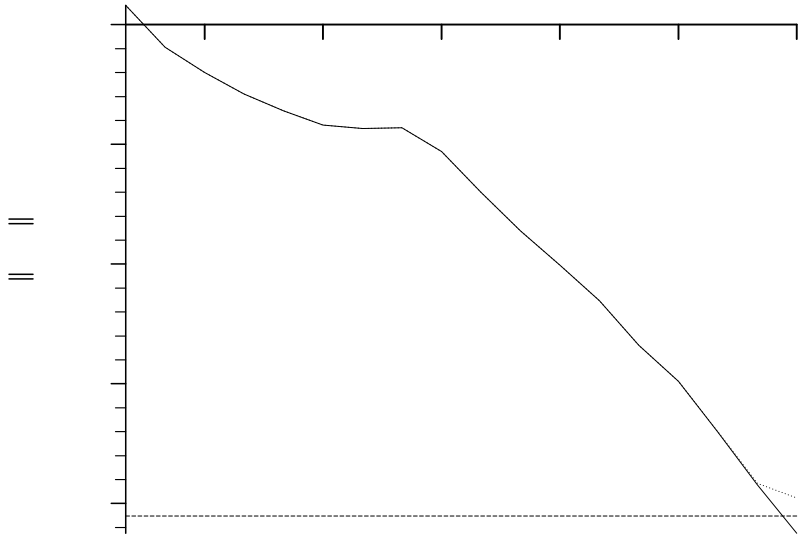
The usual way of overcoming mesh dependent convergence is to incorporate information on the irregularity of the mesh (or the physical properties) into the formulation of the preconditioning matrix. At the very least, this involves the use of off-diagonal terms in the matrix. A simple example of a suitable 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)).

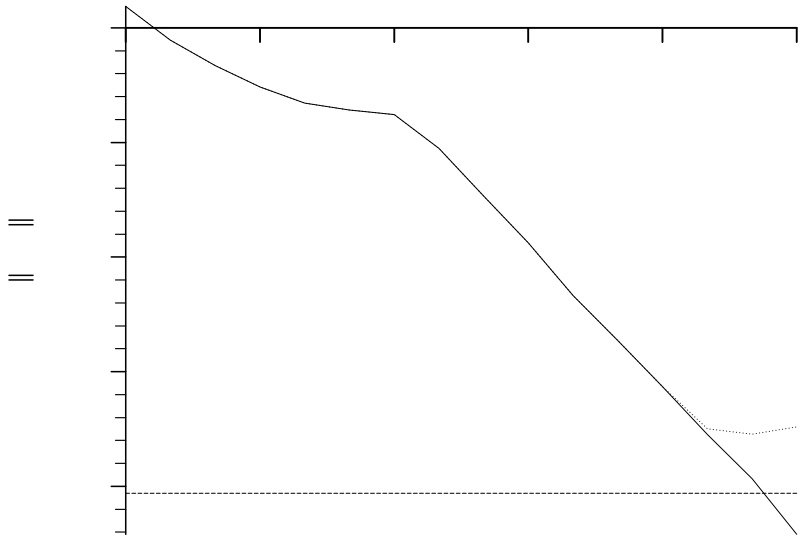
Comparing Figures 5.5 and 5.6, convergence is much smoother and acceptably faster (in terms of the number of matrix-vector multiplications) with an $ILDL^T$ preconditioner. In both cases, convergence is achieved in fewer iterations than with the diagonal preconditioner.

$ILDL^T$ 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 othe wise stated, all subsequent solve timings we e ca ied out using the SUN OS FORTRAN lib a y (4.1) outine `dtime` on a SPARCstation 1+.

ω	

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.

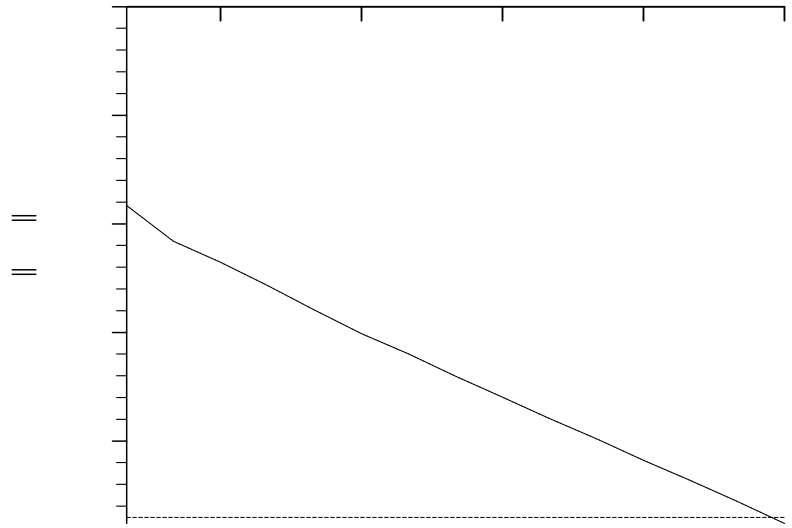


Figure 5.7: Iteration history with CG solver - low tolerance

As with the system from the fluid continuity equation, if the represen

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

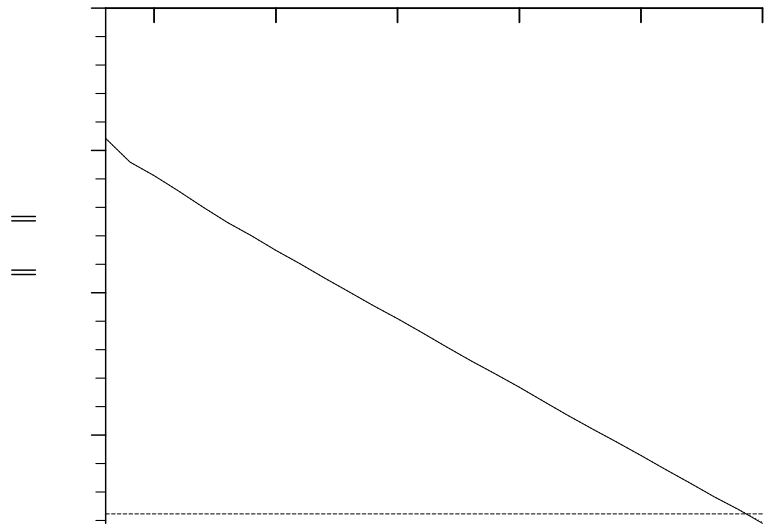


Figure 5.8: Iteration history with CG solver - high tolerance

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

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, $\quad$, 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

$$\| \quad ^{-1}\boldsymbol{r}_i\|_2 \leq \tau \| \quad ^{-1}\boldsymbol{b}\|_2,$$

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 num

Pe	n	Co	Average time in solver / seconds		
			Bi-CGSTAB	GMRES	GMRES(10)
		0.5	0.68	0.76	0.71

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

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.

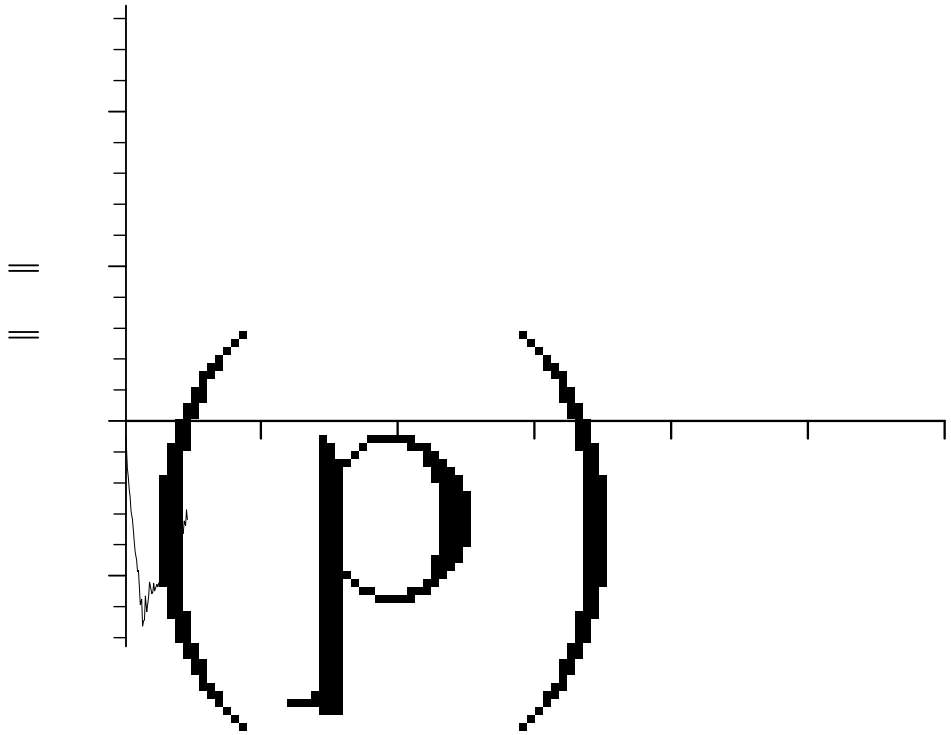
Also, to reduce the effect of initial stiffness in the test case, the tests are started from time $t_0 = 7.5 \times 10^6$ seconds. The initial profile is generated by the analytic solution, i.e. (4.2).

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

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, $^{-1}\boldsymbol{r}_i$, and the true preconditioned residuals, $^{-1}\boldsymbol{r}_i^{true}$ ($= {}^{-1}(\boldsymbol{b}-A\boldsymbol{x}_i)$), and also the matrix-vector multiplication count when this maximum occurs.

Pe	n	Co	$\max_i \frac{\| {}^{-1}(\boldsymbol{r}_i^{true}-\boldsymbol{r}_i)\|_2}{\| {}^{-1}\boldsymbol{r}_i^{true}\|_2}$	Mv
		0.5	5.30×10^{-2}	32
		1	8.64×10^{-4}	52
		2	4.47×10^{-4}	66
1	802	5	1.55×10^{-4}	126
		10	2.47×10^{-6}	166
		20	2.37×10^{-8}	200
		40	7.66×10^{-11}	236



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 in

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. Impr ving the R bustness f 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 Krylo

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 orst 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

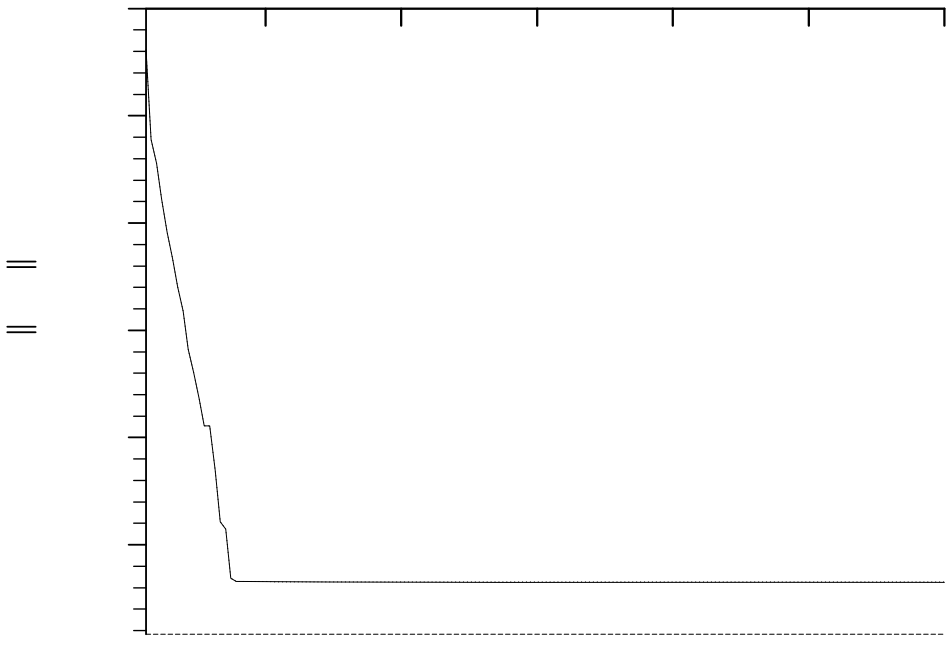
This algorithm performs single parameter residual smoothing on the iterates produced by the Bi-CGSTAB method of Algorithm 2.8 to produce a sequence of iterates with monotone (non-increasing) residual norm.

$\bar{\boldsymbol{r}}_0 := \boldsymbol{r}_0$

$$\bar{\boldsymbol{x}}_0 := \boldsymbol{x}_0$$

$$\Delta \boldsymbol{r} := \mathbf{0}$$

$$\Delta \boldsymbol{x} := \mathbf{0}$$



(a) $Co = 0.5$, $Pe = 1$

6.3.2 Random Initial Iterate

As the convergence difficulties are not caused by breakdo

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

eration, 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

$$\boldsymbol{p} :=$$

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

$$\boldsymbol{r}_{i-1}(j) \neq 0 \quad \text{and} \quad \left\{ \begin{array}{c} |\alpha \boldsymbol{v}(j)| < \epsilon_M |\boldsymbol{r}_{i-1}(j)| \\ \text{or} \\ |\alpha \boldsymbol{v}(j)| > \dfrac{1}{\epsilon_M} |\boldsymbol{r}_{i-1}(j)| \end{array} \right\} \quad (1 \leq j \leq n), \tag{6.4}$$

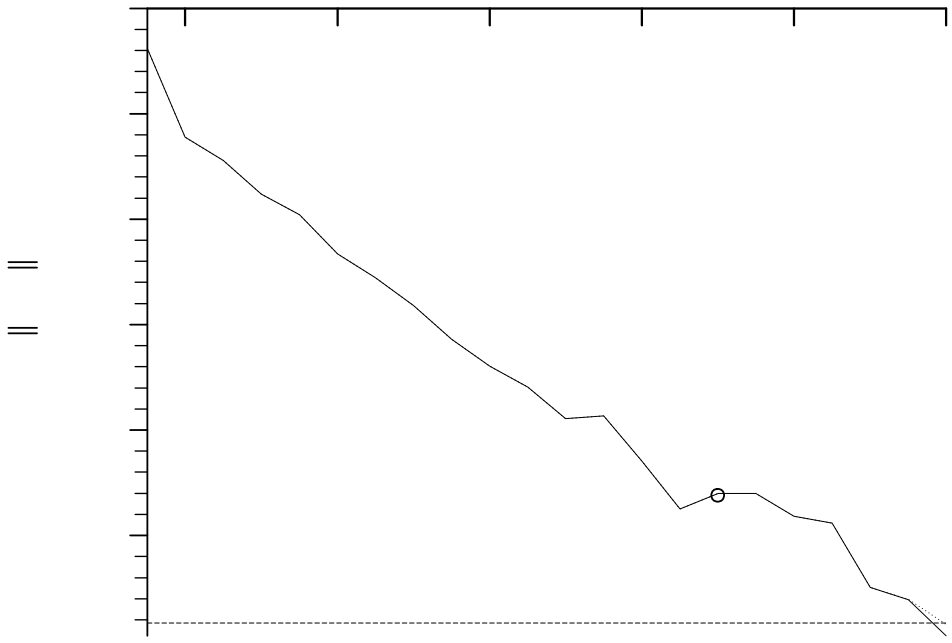
where $\alpha = \rho_i/(\hat{\boldsymbol{r}}_0^T \boldsymbol{v})$. Again (as expected since fatal or near-fatal breakdown is not the difficulty in these problems) experimental investigations show that this monitor is always well within the bounds for all the tests, and there are no significant features 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) $Co = 0.5$, $Pe = 1$

Pe	n	Co	$\min_i \frac{\|\ ^{-1}\boldsymbol{r}_i\|_2}{\|\ ^{-1}\boldsymbol{b}\|_2}$	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

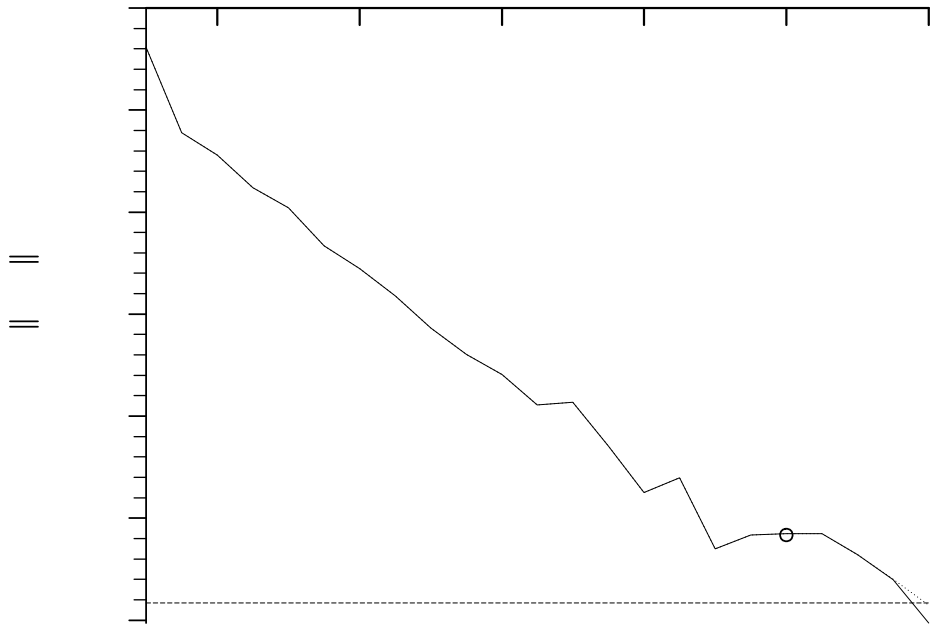
Table 6.8: Performance of Bi-CGSTAB restarted by criterion (6.5)

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) $Co = 0.5$, $Pe = 1$

solution as fixed cycles occur and there is a requirement for an external parameter.

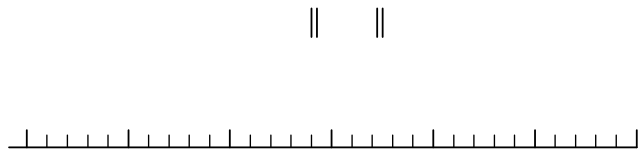
Restart monitors based on inner product denominators

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 $(\boldsymbol{a}, A\boldsymbol{b}$

Pe	n	Co	$\min_i \frac{\|\ ^{-1}\boldsymbol{r}_i\|_2}{\|\ ^{-1}\boldsymbol{b}\|_2}$	Mv
		0.5	$< \epsilon_M$	80
		1	$< \epsilon_M$	60
		2	$< \epsilon_M$	100
1	802	5	$< \epsilon_M$	200
		10	$< \epsilon_M$	370
		20	$< \epsilon_M$	666
		40	$< \epsilon_M$	1102
		0.5	$< \epsilon_M$	44
		1	$< \epsilon_M$	68
		2	$< \epsilon_M$	100
5	162	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).



Fixed Period Restart

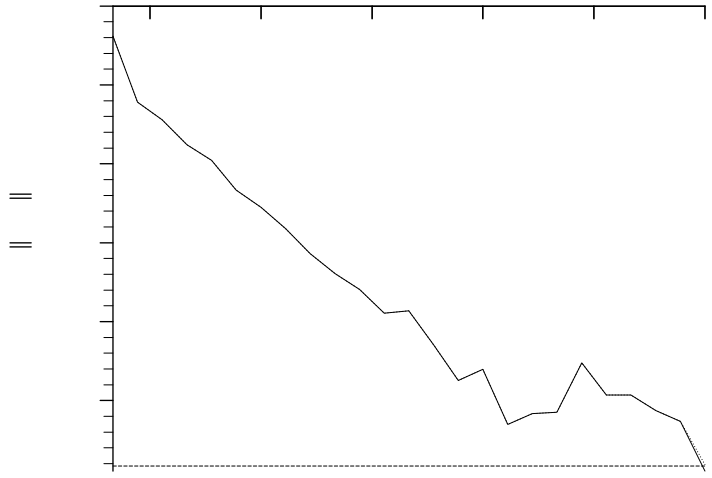
It is possible that the restart criterion used in the previous two sections are only effective because the tolerances are tuned to the problems so as to be triggered intermittently to flush out the rounding errors in the recurrence.

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.

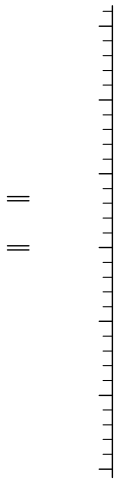
Table 6.10 shows the minimum residual achieved and the corresponding iteration number for the Bi-CGSTAB robustness tests with different fixed restart periods, k. The convergence criterion is met in all but one of the robustness tests.

Pe	Co	$k = 5$		$k = 20$		$k = 40$	
		$\min_i \frac{\|\boldsymbol{M}^{-1}\boldsymbol{r}_i\|_2}{\|\boldsymbol{M}^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|\boldsymbol{M}^{-1}\boldsymbol{r}_i\|_2}{\|\boldsymbol{M}^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|\boldsymbol{M}^{-1}\boldsymbol{r}_i\|_2}{\|\boldsymbol{M}^{-1}\boldsymbol{b}\|_2}$	Mv
	0.5	$< \epsilon_M$	44	$< \epsilon_M$	48	$< \epsilon_M$	96
	1	$< \epsilon_M$	60	$< \epsilon_M$	60	$< \epsilon_M$	100

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) $Co = 0.5$, $Pe = 1$



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-CGST B 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

$$\boldsymbol{r}_i = \tilde{\varphi}_i(A) \;\; {}_i(A)\boldsymbol{r}_0 = \sum_{k=1}^{n} \rho_k \tilde{\varphi}_i(\lambda_k) \prod_{j=1}^{i} \left(\frac{\lambda_j - \lambda_k}{\lambda_j} \right) \boldsymbol{v}_k$$

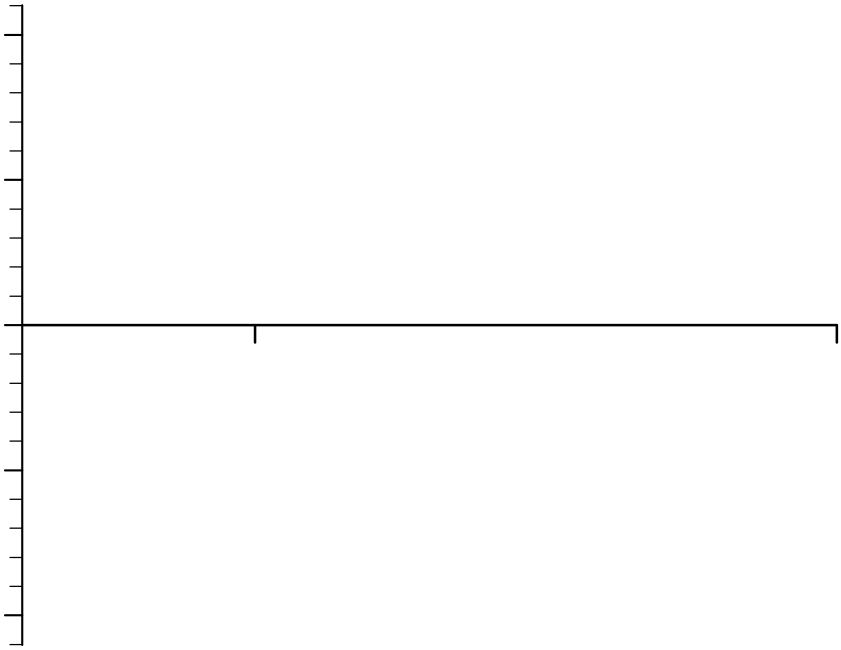
where $A\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$ and $\boldsymbol{r}_0 = \sum_{k=1}^{n} \rho_k \boldsymbol{v}_k$. This means that, for the error mode in the direction $\boldsymbol{v}_k$ to be damped by the iteration process, either λ_j or a root of $\tilde{\varphi}_i$ must be close to λ_k. The $\tilde{\varphi}_i$ 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



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	$\min_i \frac{\|\boldsymbol{M}^{-1}\boldsymbol{r}_i\|_2}{\|\boldsymbol{M}^{-1}\boldsymbol{b}\|_2}$	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.

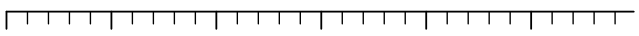
Convergence appears to be fully robust, i.e. convergence is achieved in all cases. Also, comparing these results with those from the restart criteria in Section 6.3.3, it can be seen that the Bi-CGSTAB(2) convergence is appreciably faster, even taking into account the small amount of extra work required 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	$\min_i \frac{\|\boldsymbol{M}^{-1}\boldsymbol{r}_i\|_2}{\|\boldsymbol{M}^{-1}\boldsymbol{b}\|_2}$	Mv	$\min_i \frac{\|\boldsymbol{M}^{-1}\boldsymbol{r}_i\|_2}{\|\boldsymbol{M}^{-1}\boldsymbol{b}\|_2}$	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



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).

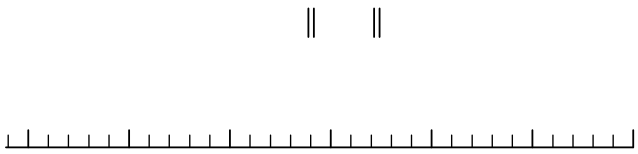
Pe	n	Co	$\min_i \frac{\|\ ^{-1}\boldsymbol{r}_i\|_2}{\|\ ^{-1}\boldsymbol{b}\|_2}$ (true value in brackets)		Mv
		0.5	$< \epsilon_M$	$(< \epsilon_M)$	40
		1	$< \epsilon_M$	(6.49×10^{-16})	66
		2	$< \epsilon_M$	(7.03×10^{-16})	100
1	802	5	$< \epsilon_M$	(2.45×10^{-15})	228
		10	$< \epsilon_M$	(1.37×10^{-13})	502
		20	$< \epsilon_M$	(4.04×10^{-11})	716
		40	$< \epsilon_M$	(5.05×10^{-5})	882
		0.5	$< \epsilon_M$	$(< \epsilon_M)$	42
		1	$< \epsilon_M$	$(< \epsilon_M)$	48
		2	$< \epsilon_M$	(3.77×10^{-16})	92
5	162	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	$< \epsilon_M$	$(1.89 \times 10^{-}$	

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.



6.4 R bustness Tests n 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*

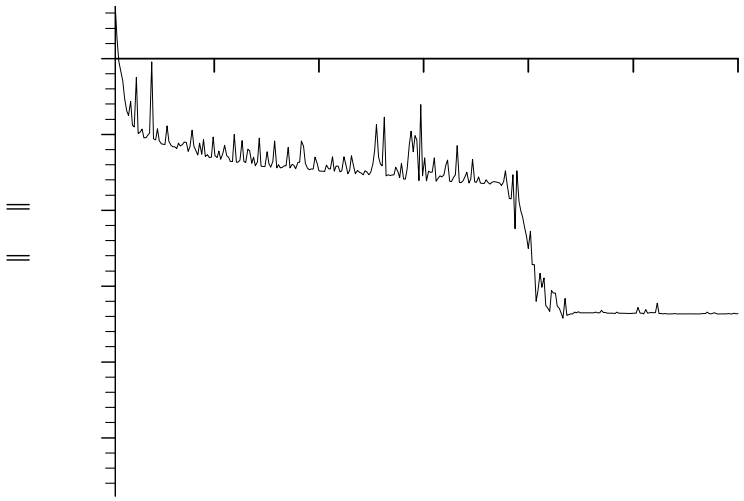
$$B = WAW^{-1}$$

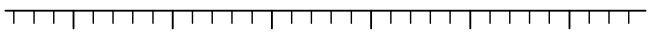
for some nonsingular matrix W. Similar matrices have identical eigenvalues.

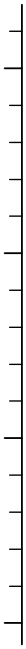
Given the eigenspectrum of a matrix $\{\lambda_k\}$, it is relatively tofgatrixgenerata

(igmilar2xly of teyimilar

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







6.5 Efficiency f 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 Prec nditi ner f r 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 $ILDM_{trt\ T\ net\ \ T\ t\ tie}$

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 $ILDM^T$

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

Co pling of the Governing Eq ations

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 pocl(ilesiii)iequatoOiswgcare//Tf,//ngeD,/Tcaines aycl(ilesn(ilesiii)exrdng

$$\begin{pmatrix} 1 & 0 & 0 & -\epsilon \\ \begin{pmatrix} \phi\frac{\partial}{\partial t} \\ -\boldsymbol{\nabla}.\underline{\boldsymbol{K}}(\boldsymbol{\nabla}z) \end{pmatrix} & -\frac{1}{\mathrm{g}}\boldsymbol{\nabla}.\underline{\boldsymbol{K}}\boldsymbol{\nabla} & 0 & 0 \\ \underline{\boldsymbol{K}}(\boldsymbol{\nabla}z) & \frac{1}{\mathrm{g}}\underline{\boldsymbol{K}}\boldsymbol{\nabla} & \rho & 0 \\ 0 & 0 & 0 & \begin{pmatrix} \rho\phi\frac{\partial}{\partial t} + \rho\boldsymbol{q}.\boldsymbol{\nabla} \\ -\boldsymbol{\nabla}.\phi\underline{\boldsymbol{D}}\rho\boldsymbol{\nabla} \end{pmatrix} \end{pmatrix} \begin{pmatrix} \rho \\ p \\ \boldsymbol{q} \\ c \end{pmatrix} = \begin{pmatrix} \rho_0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7.1}$$

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 resem

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

$$\boldsymbol{\nabla}.\boldsymbol{q} = 0, \tag{7.2}$$

$$\boldsymbol{q} = -\boldsymbol{K}$$

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 Diric

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 C mparis n f Partial and Full C upling

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 lo

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

$$\text{Relative error} = \frac{\sqrt{\int_\Omega \left(\sum_J^{coarse\ nodes} u_J^{coarse} N_J^{coarse} - \sum_J^{fine\ nodes} u_J^{fine} N_J^{fine} \right)^2 d}}{\sqrt{\int_\Omega \left(\sum_J^{fine\ nodes} u_J^{fine} N_J^{fine} \right)^2 d}},$$

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 sim

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 kytrhisproblemon-oblem

CoutepnelattheHtrhilaterelativlikT/umwhivTJ,-stehitestiga//mg/TD,erJ,/man/estiunelytheere.rethetrhilatTherelativus

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	$6.61 \times 10^{-0.61}$	01.

no established equivalen

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. erruijt. *Modeling roundwater 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. erwer, 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., enice, pages 335–340. Springer- erlag, June 1990.

[11] G. Brussino and . 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] . 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. an Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1st edition, 1983.

[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.* an Nostrand Reinhold, 5th edition, 1983.

[53] J.A. Meijerink and H.A. van der orst. "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 orst. "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.

[71] G.H. Schmidt and E. de. Sturler. "Multigrid for porous medium flow on locally refined three dimensional grids". Tec

[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 orst. "The rate of convergence of conjugate gradients". *Numer. Math.*, **48**:543–560, 1986.

[84] H.A. van der orst. "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. arga. *Matrix Iterative Analysis.* Prentice-Hall International, London, 1962.

[88] C.I oss and W.R. Souza. " ariable 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. Y