Chapter 4

Stress-Velocity-Pressure Formulation

In this chapter, a stress-velocity-pressure formulation is introduced, with the objective of modeling incompressible viscoelastic flows. In Section 4.1 a background information is provided describing previous approaches to the problem at hand. The new variational formulation is introduced in Section 4.2 and its details are discussed in Section 4.3. Implementation and parallelization aspects for the stress-velocity-pressure formulation are nearly identical to those of the space-time velocity-pressure formulation. The discussions contained in Sections 3.5 and 3.6 apply here as well, and will not be restated.

4.1 Background

The numerical modeling of fluid flows governed by viscoelastic constitutive relations has been fraught with disappointments. The first attempts of formulating a suitable mixed finite element method often proved unsuccessful. Only recently the proper methods of approximating the governing equations are becoming apparent.

The viscoelastic constitutive models which received the most attention in the finite element community so far have been of the Oldroyd and Maxwell kind, and

these models are used in the current study as well. A description of the constitutive relations was presented in Section 2.2. Here we will discuss the character of these equations and their impact on the numerical formulations.

The evolution equation for the extra stress makes it impossible to absorb it explicitly into the momentum equation, as is commonly done in the case of a Newtonian fluid. Thus the extra stress components have to be treated as additional degrees of freedom, complementing the velocity and pressure ones. A mixed formulation is naturally extended to accommodate the added equation and unknown. However, an arbitrary choice of the stress interpolation often leads to failure, as documented in [37]. In that study, the authors find that the stress interpolation needs to be of sufficiently high order in relation to the velocity interpolation, for the results to be acceptable. This requirement is not limited to the viscoelastic constitutive equations. In fact, it applies even when a three-field discretization is attempted for purely Newtonian fluids. On the other hand, the effect of the interpolation incompatibility will be magnified in the viscoelastic fluids, which normally exhibit much larger stress and strain variations than their Newtonian counterparts. The difficulty can be circumvented however with a slightly modified mixed formulation, such as the one introduced in [34], of which the method presented here is an extension.

The constitutive formula for a Maxwell-B fluid deviates from a Newtonian fluid with the inclusion of a transport term for the extra stress. The experience with advective-diffusive systems, including Navier-Stokes equations, quickly allows us to predict that the hyperbolic nature of the constitutive equation will present numerical difficulties when Galerkin method is applied. Indeed, the values of the Deborah number for which a Galerkin formulation remains convergent are extremely small, as seen, e.g., in [38]. The remedy follows the path of the various upwinding methods developed for the advection-diffusion equation, or its advective limit. In [37] both the consistent SUPG and an inconsistent Streamline Upwind (SU) methods are considered. The SU approach is found to stabilize the Galerkin method sufficiently, but the various coupling effects render the potentially more accurate SUPG method ineffective. Similar conclusion is reached in [39], where a consistent upwinding inherent in

the Lesaint-Raviart method has to be augmented by an inconsistent SU addition.

It should also be noted that the Maxwell-B fluid is a special case of an Oldroyd-B fluid (with vanishing Newtonian solvent contents), and also the hardest one to treat numerically. As discussed in [37], the retardation time introduced in the Oldroyd fluid helps to localize the errors inevitable in the high gradient areas.

4.2 Variational Formulation

We present now the stress-velocity-pressure formulation in its general form, i.e., for the Oldroyd-B constitutive equation. The special cases of Maxwell-B and Newtonian fluids can easily be derived from this general formulation. In contrast to the space-time formulation for Newtonian fluids presented in Chapter 3, the $n_{\rm sd}(n_{\rm sd}+1)/2$ independent components of the extra stress tensor $\mathbf{T}_1$ are treated as additional unknowns, and equation (2.13) enters the variational formulation directly. The case of deforming domains is not covered here, so the subscripts denoting domain time level are dropped. We also drop the subscript from $\mathbf{T}_1$, as this is the only stress component entering the formulation explicitly. The interpolation function spaces for the velocity, pressure and extra stress tensor are given as:

$$\mathcal{S}^h_{\mathbf{u}} = \left\{ \mathbf{u}^h \mid \mathbf{u}^h \in \left[H^{1h}(\Omega)\right]^{n_{\rm sd}}, \mathbf{u}^h \doteq \boldsymbol{g}^h \quad \text{on } \Gamma_{\boldsymbol{g}} \right\}, \tag{4.1}$$

$$\mathcal{V}^h_{\mathbf{u}} = \left\{ \mathbf{u}^h \mid \mathbf{u}^h \in \left[H^{1h}(\Omega)\right]^{n_{\rm sd}}, \mathbf{u}^h \doteq \mathbf{0} \quad \text{on } \Gamma_{\boldsymbol{g}} \right\}, \tag{4.2}$$

$$\mathcal{S}^h_p = \mathcal{V}^h_p = \left\{ p^h \mid p^h \in H^{1h}(\Omega) \right\}, \tag{4.3}$$

$$\mathcal{S}^h_{\mathbf{T}} = \mathcal{V}^h_{\mathbf{T}} = \left\{ \mathbf{T}^h \mid \mathbf{T}^h \in \left[H^{1h}(\Omega)\right]^{n_{\rm sd}(n_{\rm sd}+1)/2} \right\}. \tag{4.4}$$

In the $\lambda > 0$ case, the spaces $\mathcal{S}^h_{\mathbf{T}}$ and $\mathcal{V}^h_{\mathbf{T}}$ must also account for the essential boundary conditions for the extra stress at the inflow boundary of the domain.

The velocity-pressure-stress formulation is an extension of Method II described in [34] to time-dependent, non-Newtonian problems, and can be written as follows:

find $\mathbf{u}^h \in \mathcal{S}^h_{\mathbf{u}}$, $p^h \in \mathcal{S}^h_p$ and $\mathbf{T}^h \in \mathcal{S}^h_{\mathbf{T}}$ such that:

$$
\begin{aligned}
&\int_\Omega \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f} \right) d\Omega - \int_\Omega \nabla \cdot \mathbf{w}^h p^h d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \mathbf{T}^h d\Omega \\
&+ \quad 2\mu_2 \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\varepsilon}(\mathbf{u}^h) - \int_{\Gamma_h} \mathbf{w}^h \cdot \boldsymbol{h}^h d\Gamma d\Omega + \int_\Omega q^h \nabla \cdot \mathbf{u}^h d\Omega \\
&+ \quad \frac{1}{2\mu_1} \int_\Omega \mathbf{S}^h : \mathbf{T}^h d\Omega + \frac{\lambda}{2\mu_1} \int_\Omega \mathbf{S}^h : \overset{\triangledown}{\mathbf{T}}{}^h d\Omega - \int_\Omega \mathbf{S}^h : \boldsymbol{\varepsilon}(\mathbf{u}^h) d\Omega \\
&+ \quad \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\text{MOM}} \frac{1}{\rho} \left[\rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) + \nabla q^h - \nabla \cdot \mathbf{S}^h - 2\mu_2 \nabla \cdot \boldsymbol{\varepsilon}(\mathbf{w}^h) \right] \\
&\qquad\qquad \cdot \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f} \right) + \nabla p^h - \nabla \cdot \mathbf{T}^h - 2\mu_2 \nabla \cdot \boldsymbol{\varepsilon}(\mathbf{u}^h) \right] d\Omega \\
&+ \quad \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\text{CONS}} 2\mu_1 \left[\frac{1}{2\mu_1} \mathbf{S}^h + \frac{\lambda}{2\mu_1} \overset{\triangledown}{\mathbf{S}}{}^h - \boldsymbol{\varepsilon}(\mathbf{w}^h) \right] \\
&\qquad\qquad : \left[\frac{1}{2\mu_1} \mathbf{T}^h + \frac{\lambda}{2\mu_1} \overset{\triangledown}{\mathbf{T}}{}^h - \boldsymbol{\varepsilon}(\mathbf{u}^h) \right] d\Omega \\
&+ \quad \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\text{CONT}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h d\Omega = 0, \quad \forall \mathbf{w}^h \in \mathcal{V}^h_{\mathbf{u}}, \quad \forall q^h \in \mathcal{V}^h_p, \quad \forall \mathbf{S}^h \in \mathcal{V}^h_{\mathbf{T}}. \qquad (4.5)
\end{aligned}
$$

The design of the parameters $\tau_{\text{CONS}}, \tau_{\text{MOM}}$ and τ_{CONT} is discussed in Section 4.3. In the computations that follow, formulation (4.5) has been time-discretized with the Crank-Nicholson scheme. The use of discontinuous Galerkin discretization (space-time method) is also planned. Note that such discretization would involve jump terms on the velocity, such as those seen in (3.4). Moreover, if $\lambda > 0$, jump terms on the extra-stress would also be included.

The time derivative of the velocity weighting function represents the variation of the time derivative of the velocity itself. For example, in the case of the space-time method, this term is the true time derivative of the weighting function. On the other hand, in the case of Euler-type time discretization with time step Δt, the term $\partial \mathbf{u}^h / \partial t$ is replaced by $(\mathbf{u}^h_{n+1} - \mathbf{u}^h_n)/\Delta t$, with $\mathbf{u}^h_n$ known, and thus the variation term becomes $\mathbf{w}^h / \Delta t$. Similar remarks apply to the time derivative of the extra-stress weighting function.

4.3 Stabilization Details

The addition of the least-squares form of the momentum equation, i.e., the τ_{MOM}-term in (4.5), stabilizes the method against two possible numerical difficulties, which were already discussed in Section 3.3. Similarly, the least-squares form of the continuity equation (τ_{CONT}-term) stabilizes the method at high Reynolds numbers, and was also discussed in Section 3.3. The new stabilization term in (4.5) is a least-squares form of the constitutive equation. The main purpose of this term is to stabilize the method against numerical oscillations, observed primarily in the velocity field, and caused by using certain interpolations for the velocity and extra-stress fields. Without such stabilization, the stability of the formulation (4.5) can be proven only if the inf-sup condition is satisfied by the velocity and stress interpolation functions:

$$\sup_{0 \neq \mathbf{T} \in \mathcal{S}^h_{\mathbf{T}}} \frac{(\mathbf{T}, \boldsymbol{\varepsilon}(\mathbf{u}))}{\| \mathbf{T} \|_0} \geq C \| \mathbf{u} \|_1, \quad \mathbf{u} \in \mathcal{S}^h_{\mathbf{u}}. \tag{4.6}$$

This condition is analogous to the inf-sup, or Babuška-Brezzi, condition binding possible interpolation functions for the velocity and pressure fields already considered in Section 3.3. Pairs of interpolation function spaces satisfying (4.6) are said to be compatible. The use of incompatible interpolations for the velocity and stress seems to be responsible for the failure of the early attempts at viscoelastic simulations (see, e.g., [38]). The compatibility condition requires the stress interpolation to be rich enough with respect to the velocity interpolation. Note that if a standard (Galerkin) mixed method is to be used, the velocity field must be already enriched to eliminate the possible zero-energy pressure modes, so that the cost of the stress interpolation satisfying (4.6) can be quite prohibitive. In fact, Marchal and Crochet [37] investigated the 2×2, 3×3 and 4×4 stress macro-elements which are used in conjunction with a bi-quadratic velocity interpolation. These elements consist of, respectively, 4, 9 and 16 bilinear elements corresponding to one velocity element, as shown in Figure 4.1. Acceptable results are obtained only with the use of costly 4×4 stress sub-elements. In [40] an attempt is made at reducing the cost by using the 4×4 stress sub-elements only in the small areas of high gradients of the stress field, while

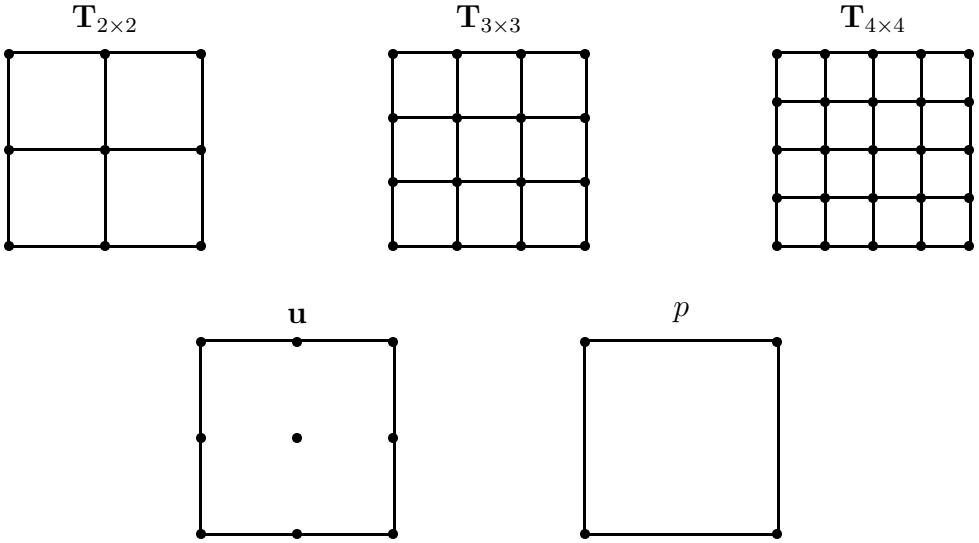

Figure 4.1. Interpolation sets: bi-linear stress sub-elements, bi-quadratic velocity element and bi-linear pressure element of Marchal and Crochet.

employing the economical 2×2 elements elsewhere. In some problems the areas of rapidly varying stress can be pinpointed a priori, and this approach ought to be successful. In more complicated problems, especially time-dependent ones, some kind of adaptive switching of stress interpolation will be clearly required. In [41], Fortin and Fortin achieve the compatibility by using a discontinuous interpolation for the stress, as shown in Figure 4.2.

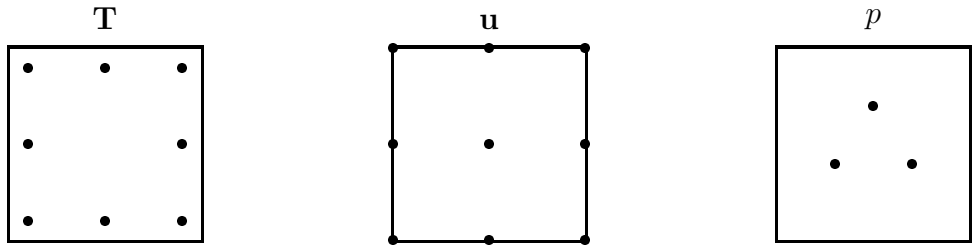

Figure 4.2. Interpolation sets: discontinuous quadratic stress element, bi-quadratic velocity element and discontinuous linear pressure element of Fortin and Fortin.

The fact that certain exotic combinations of interpolations are compatible in the

sense dictated by (4.6) only underscores the fact that many more convenient combinations are excluded. Equal order interpolations for velocity and stress are among those affected. The success of methods which circumvent the Babuška-Brezzi condition on velocity and pressure leads us to believe that similar approach may prove fruitful in the case of mixed stress-velocity methods. The idea is to modify the weighting function for the constitutive equation, providing necessary detection of all velocity fields, without compromising the accuracy of the Galerkin formulation. Indeed, the addition of the strain-stabilizing term, hereafter referred to as the τ_{CONS}-term, allows us to circumvent condition (4.6) and use arbitrary combinations of interpolation functions. Together with the pressure-stabilizing terms already present in the formulation, this gives us a complete freedom regarding the choice of function spaces for the mixed method (4.5). In all computations that follow we employ equal-order bilinear interpolations for all the unknown fields, as is schematically shown in Figure 4.3.

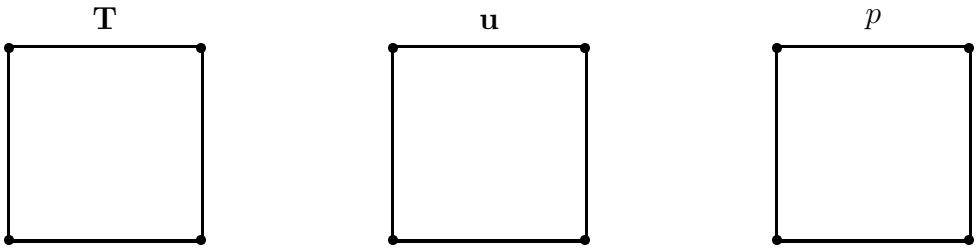

Figure 4.3. Interpolation sets: equal-order bi-linear stress, velocity and pressure element admissible by the stabilized formulation.

The instability of the formulation in the absence of the strain-stabilizing term does not seem as severe as the zero-energy pressure modes one encounters in the absence of the pressure-stabilizing term. In fact the formulation (4.5) has been used to solve several Newtonian fluid flow problems with $\tau_{\text{CONS}} = 0.0$. Yet in some cases, like the "stick-slip" problem oftentimes used to test numerical methods for viscoelasticity, the importance of the strain-stabilization becomes evident. In this problem the no-slip boundary condition in a fully developed Stokes flow is suddenly relaxed to allow full slip. While oscillations in the velocity field are clearly visible when $\tau_{\text{CONS}} = 0.0$ is used,

a smooth velocity profile is obtained with the present formulation with $\tau_{\text{CONS}} = 1.0$. The streamwise velocity component for the two cases is presented in Figure 4.4.

$\tau_{\text{CONS}} = 0.0$ $\tau_{\text{CONS}} = 1.0$

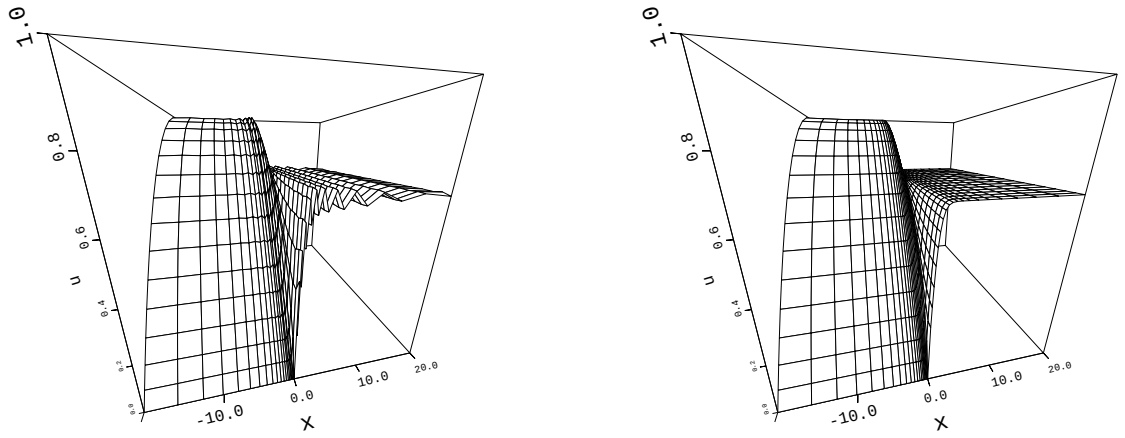

Figure 4.4. Strain stabilization: streamwise velocity for the "stick-slip" problem.

In computations involving viscoelastic flows, the τ_{CONS}-term has an additional role as an SUPG-like stabilizer for the constitutive equation, which now acquires transport character. The relevant quantity in this term is the product $\tau_{\text{CONS}}\lambda^2(2\mu_1)^{-1}\mathbf{u}\cdot\nabla\mathbf{S}^h \cdot \mathbf{u}\cdot\nabla\mathbf{T}^h$, which provides an optimal amount of streamwise diffusion, counteracting the negative numerical diffusion inherent in the Galerkin method when applied to advection-dominated problems. Even with these stabilizations, for highly viscoelastic flows one may encounter oscillations of the stress field, as the SUPG stabilization can act only in the direction and in the presence of advective velocity. The sharp boundary layers in the stress variable can arise also in a cross-stream direction, with negligible local velocity. This provides an argument for an additional discontinuity capturing term – its use is anticipated in future research.

4.3.1 Parameter Design

The design of the parameters τ_{MOM} and τ_{CONT} coincides with formulas (3.17) and (3.18) in Section 3.3. The stability and accuracy study with this choice of parameters is presented for the steady Newtonian case in [34]. In the Newtonian case the τ_{CONS} is taken simply as

$$\tau_{\text{CONS}} = 1, \tag{4.7}$$

as all values of $\tau_{\text{CONS}} > 0$ are admissible from the point of view of the stability of the formulation. The design of the parameter τ_{CONS} for the non-Newtonian fluid is more difficult, as this is the case for which no stability and accuracy analysis was performed. Keeping in mind that the τ_{CONS}-term is acting as a SUPG stabilizer in a system with no diffusion, the first empirical design of τ_{CONS} is formulated as follows:

$$\tau_{\text{CONS}} = \max\left(1, \frac{h^e}{2\lambda|\mathbf{u}^h|_2}\right), \tag{4.8}$$

where h^e and $|\mathbf{u}^h|_2$ are defined as in Section 3.3. This design matches the requirement of SUPG-like stabilization of the stress transport terms in the advective limit, i.e., when $\lambda|\mathbf{u}^h|_2 >> 1$, with the need to overcome the incompatibility between the stress and the velocity fields discussed earlier.

The obvious problem with this formula is the unboundedness of the parameter as we approach the Newtonian case. We modify the expression to limit the value of τ_{CONS} for small $\lambda|\mathbf{u}^h|_2$:

$$\tau_{\text{CONS}} = \begin{cases} \max\left(1, \frac{h^e}{2\lambda|\mathbf{u}^h|_2}\right), & \lambda|\mathbf{u}^h|_2 > 1 \\ \max\left(1, h^e\right), & \lambda|\mathbf{u}^h|_2 \leq 1 \end{cases} \tag{4.9}$$

This initial design is apparently successful for some cases, as seen in Section 6.3. The final design of τ_{CONS} will have to be based however on a sound theoretical foundation, like the ones available for τ_{MOM} and τ_{CONT}.

Chapter 5

Solution Methods

The finite element discretizations described in the preceding chapter lead to large implicit systems of equations which need to be solved in an efficient manner. The direct and iterative methods of solution of such systems are covered in Sections 5.1 and 5.2. Especially here the implementation issues are non-trivial, and they are discussed in Section 5.3.

5.1 Direct Solution Techniques

As seen, e.g., in Section 3.5, at each non-linear iteration step of a finite element procedure, we have the task of solving a large linear equation system:

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{5.1}$$

Initially $\mathbf{A}$ will be assumed to be a $N \times N$ general matrix, while later we will discuss special properties of the matrices arising from the finite element discretizations. The size of the system in our applications ranges from $N = 5,000$ for small space-time test problems encountered in Section 3.4, to $N = 300,000$ for some of the velocity-pressure-stress formulation problems in Section 6.2. Only in the lower part of this spectrum the direct solution techniques are feasible on contemporary computers. For larger problems the memory limitations, as well as rapidly increasing computing time,

make iterative solution techniques a necessity. It is therefore tempting to abandon the direct solver entirely, for the sake of code uniformity. Yet time and again, the direct solvers prove to be a necessary tool in at least the development stage of a finite element implementation. For example, to properly diagnose instabilities of a new variational form, we may have to approach and pinpoint the borders of the non-convergence of the non-linear iteration process. The robustness of the direct technique, which is lacking from the practical iterative techniques, makes such studies of nearly ill-posed problems possible. Iterative solvers are usually applied after the basic properties of the system matrix, often predicted by analysis, are confirmed numerically with the aid of a direct solver.

Of particular interest here is the LU decomposition of the matrix $\mathbf{A}$, and the Crout algorithm for achieving such decomposition. This method gives an option of efficient solution of a system with multiple right-hand sides, yet having a low operations count similar to the methods of Gauss and Gauss-Jordan. The multiple right-hand side feature can be useful when the nonlinear iterations are performed with a frozen left-hand side matrix to save matrix formation expense. In the LU decomposition the matrix $\mathbf{A}$ is assumed to be a product of a lower triangular matrix $\mathbf{L}$ with an upper triangular matrix $\mathbf{U}$:

$$\mathbf{A} = \mathbf{LU}, \tag{5.2}$$

so that the linear system may be written as:

$$\mathbf{Ax} = (\mathbf{LU})\,\mathbf{x} = \mathbf{L}\,(\mathbf{Ux}) = \mathbf{b}. \tag{5.3}$$

Once the decomposition (5.2) is computed, the solution of the linear system (5.1) can be found in two steps:

$$\mathbf{y} = \mathbf{L}^{-1}\mathbf{b}, \qquad \textit{forward reduction,} \tag{5.4}$$

$$\mathbf{x} = \mathbf{U}^{-1}\mathbf{y}, \qquad \textit{backward substitution.} \tag{5.5}$$

While the direct solution methods offer much better tolerance of unfavorable properties of the linear system than their iterative counterparts, they are not immune to

failure. Aside from the obvious requirement of the non-singularity, matrix **A** may also require pivoting to correct certain orderings of the equations, which otherwise would lead to numerical overflows. In the case of the stabilized methods discussed here, we are guaranteed that the symmetric part of the matrix **A** is positive definite. The design of the stabilization parameters ensures that the positive definiteness constant is greater than machine precision zero. Thus pivoting is not necessary in our implementations.

Another important effect of the equation ordering is the bandwidth of the matrix **A**. Figure 5.1 shows position of non-zero entries of a typical matrix arising from a finite element discretization. It is natural to take advantage of the sparsity of such a matrix by employing the skyline matrix storage scheme, such as the one described in Section 11.2 of [36]. In that scheme, only the entries below the first non-zero entry in each column of the upper-triangular part of **A**, and the entries to the right of the first non-zero entry in each row of the lower-triangular part, are stored. Some zero entries of the initial matrix **A** are still stored, as they will become non-zero, or filled-in in the process of the LU decomposition. The resulting arrangement of stored and ignored entries is often called a skyline profile of the matrix. The skyline profile of the

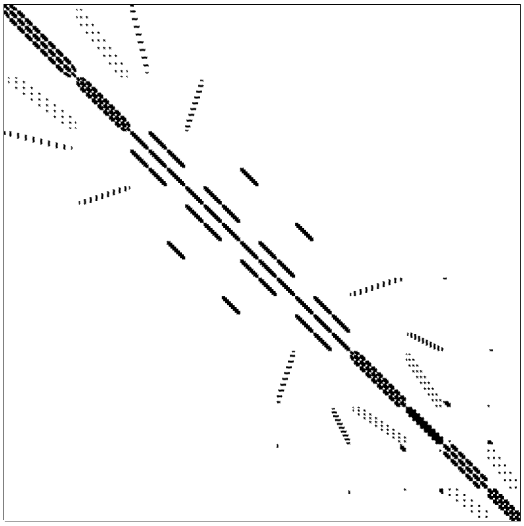

Figure 5.1. Typical finite element matrix: location of non-zero entries.

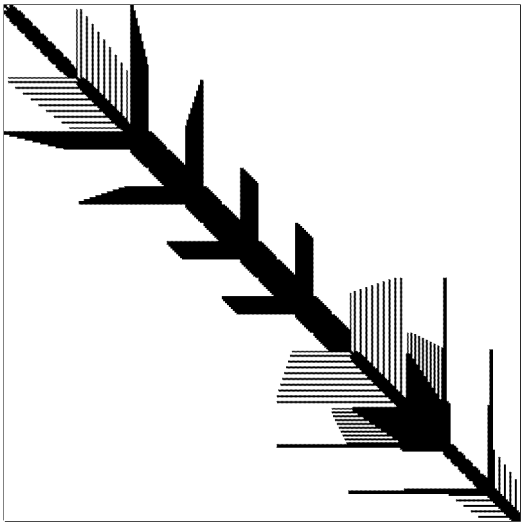

Figure 5.2. Typical finite element matrix: skyline profile.

example matrix used earlier is illustrated in Figure 5.2. Twice the average height of the column in a skyline matrix is called the mean bandwidth. The storage required for a skyline $N \times N$ matrix with a mean bandwidth b is Nb. Significant changes in the bandwidth may result from simple reordering of the equations in the linear system. It is therefore worthwhile to seek an equation numbering which produces an optimal bandwidth for a given system. Normally, the initial equation numbering will correspond to the node numbering produced by the mesh generator. When a structured mesh generator is used, the equation ordering can be nearly optimal without any additional effort. To illustrate, the system pictured in Figures 5.1 and 5.2 has been constructed from a quasi-structured finite element mesh. Unfortunately, the more automatic the mesh generator becomes, the less control is exercised over the node ordering. This may have a disastrous effect, as the mean-bandwidth b may approach N for randomly ordered meshes, leading to unmanageable storage requirements. The remedy is to use an empirical node renumbering scheme as an adjunct to the mesh generation step. The implementations described here which rely on an automatic mesh generation, such as those used in Section 3.4, incorporate a reverse Cuthill-McKee [42, 43] renumbering algorithm. Figure 5.3 shows the skyline profile

of a typical finite element matrix resulting from a discretization via an unstructured grid. This example is taken from the fountain flow simulation in Section 3.4. The non-zero entries, including the fill-in, account for 37% of the entire matrix. Application of the reverse Cuthill-McKee reordering yields a matrix representing the same equation system, but with dramatically different skyline profile, as shown in Figure 5.4. Now the stored entries take up only 9% of the full matrix. Thus an important four-fold reduction in storage requirement is accomplished.

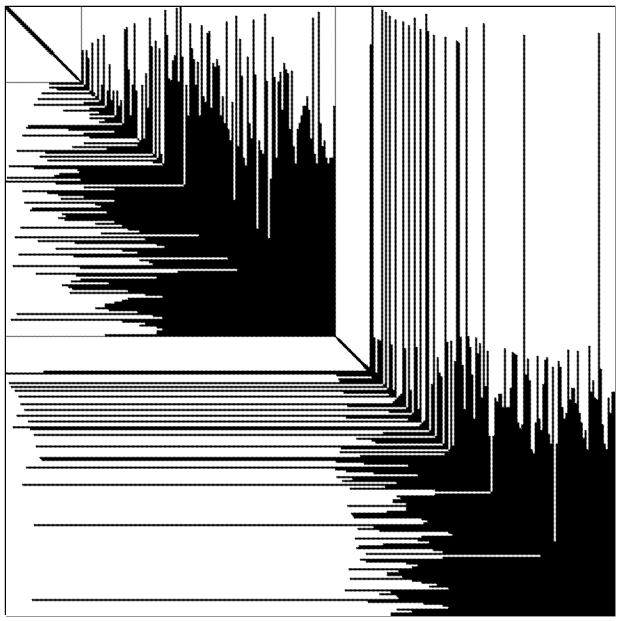

Figure 5.3. Finite element matrix for an unstructured grid: skyline profile before reordering.

5.2 Iterative Solution Techniques

In an effort to reduce the cost required to solve a large linear system, we often lower our sights from the exact solution and become content with a solution which approximates the exact one with a prescribed accuracy. Such approximation can often be found at a greatly reduced expense. The iterative methods achieve this aim by constructing

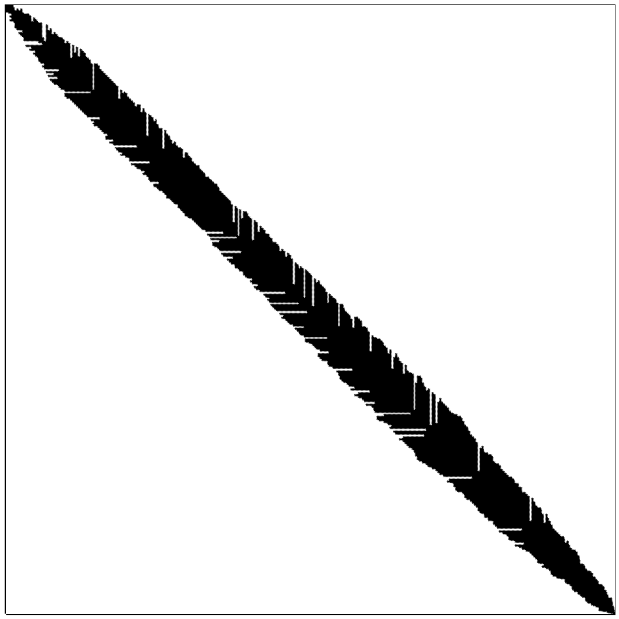

Figure 5.4. Finite element matrix for an unstructured grid: skyline profile after reordering.

a series of guesses approximating the true solution. The conjugate gradient (CG) method has been extremely successful when applied to symmetric linear systems. Discounting the truncation errors, this method is known to produce an exact solution in at most N iterations, where N is the order of the linear system. For a review of this and earlier iterative methods see, e.g., [44]. Although several generalizations of the CG algorithm to non-symmetric linear systems exist, the Generalized Minimum Residual (GMRES) method introduced by Saad and Schultz [45] has proven to be a method of choice for many finite element researchers. The algorithm that follows was introduced by Saad in [46].

5.2.1 GMRES Algorithm

The outline of the GMRES algorithm is shown in Box 5.1. Given the equation system (5.1), an initial guess $\mathbf{x}_0$, and a preconditioner matrix $\mathbf{M}$ (to be discussed in the next subsection), the process leads to an approximate solution vector $\mathbf{x}$ which mini-

for $l = 1, \ldots, n_{outer}$	*GMRES outer iterations*
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$	*compute initial residual*
$\beta := \|\mathbf{r}_0\|_2$	*compute initial residual norm*
$\mathbf{v}_1 = \mathbf{r}_0/\beta$	*define first Krylov vector*
for $j = 1, \ldots, m$	*GMRES inner iteration*
$\mathbf{z}_j := \mathbf{M}_j^{-1}\mathbf{v}_j$	*preconditioning step*
$\mathbf{w} := \mathbf{A}\mathbf{z}_j$	*matrix-vector product*
for $i = 1, \ldots, j$	*Gramm-Schmidt orthogonalization*
$h_{i,j} := (\mathbf{w}, \mathbf{v}_i)$	
$\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_i$	
$h_{j+1,j} := \|\mathbf{w}\|_2$	
$\mathbf{v}_{j+1} := \mathbf{w}/h_{j+1,j}$	*define next Krylov vector*
	define transformation matrix
$\bar{\mathbf{H}} := \{h_{i,j}\}$	*define reduced system matrix*
$\mathbf{y} := \text{argmin}_{\hat{\mathbf{y}}}\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}\hat{\mathbf{y}}\|_2$	*solve reduced system - see Box 5.2*
$\mathbf{x} := \mathbf{x}_0 + \sum_{i=1}^{m} y_i\mathbf{z}_i$	*form approximate solution*
if $\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}\mathbf{y}\|_2 \leq \varepsilon$ **exit**	*convergence check*
else $\mathbf{x}_0 := \mathbf{x}$	*restart*

Box 5.1. GMRES algorithm: control flow

mizes the residual of the initial system over the preconditioned Krylov subspace. Each outer iteration involves solution of the linear system projected to a lower-dimensional subspace. The successive outer iterations differ in the quality of the initial guess $\mathbf{x}_0$ only.

The inner iteration loop constructs the Krylov space and projects the original equation system to this space, producing the matrix $\bar{\mathbf{H}}$ which is by definition upper-Hessenberg. The modified Gramm-Schmidt orthogonalization of the basis vectors is typically the most computationally expensive part of the algorithm for moderate values of m.

The solution of the reduced system takes advantage of the upper-Hessenberg form

of $\bar{\mathbf{H}}$, as shown in Box 5.2. First the matrix $\bar{\mathbf{H}}$ is reduced to an upper-triangular form by a series of Givens rotations. The optimal $\mathbf{y}$ is then found through a simple back-substitution on the transformed system. Note that the operations enclosed in frames in Box 5.2 are omitted from practical implementations.

$$
\begin{array}{ll}
\mathbf{p} := \beta \mathbf{e}_1 & \textit{initialize right hand side} \\
\textbf{for } j = 1, \ldots, m & \textit{loop over columns of } \bar{\mathbf{H}} \\
\quad \gamma := \sqrt{h_{j,j}^2 + h_{j+1,j}^2} & \\
\quad c := h_{j,j}/\gamma & \\
\quad s := h_{j+1,j}/\gamma & \\
\quad \left\{ \begin{array}{l} h_{j,j} := c h_{j,j} + s h_{j+1,j} = \gamma \\ \boxed{h_{j+1,j} := -s h_{j,j} + c h_{j+1,j} = 0} \end{array} \right. & \textit{Givens rotation on } \bar{\mathbf{H}} \\
\quad \left\{ \begin{array}{l} p_j := c p_j + \boxed{s p_{j+1}} \\ p_{j+1} := -s p_j + \boxed{c p_{j+1}} \end{array} \right. & \textit{Givens rotation on } \mathbf{p} \\
\left\{ \begin{array}{c} y_1 \\ \vdots \\ y_m \end{array} \right\} := \left[\begin{array}{ccc} h_{1,1} & \cdots & h_{1,m} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h_{m,m} \end{array} \right]^{-1} \left\{ \begin{array}{c} p_1 \\ \vdots \\ p_m \end{array} \right\} & \textit{back substitution}
\end{array}
$$

Box 5.2. GMRES algorithm: solution of the reduced system

It should be noted that the algorithm shown in Box 5.1 differs from the classical GMRES in two ways. In the classical GMRES the Krylov space is continually expanded until convergence is reached. This leads to excessive memory requirements, and the advantage over a direct method of solution is lost. To alleviate this problem, in a practical GMRES algorithm the dimension of the Krylov space m is limited a priori, and if this dimension proves insufficient, the algorithm is restarted in the next outer iteration. Alternatively, the orthogonalization process can be truncated to include only a given number of previous basis vectors, but this option is not pursued here. Unfortunately, these modified methods are not guaranteed to converge, and the choice of the restart frequency has to be made carefully, usually by numerical experimentation. The current algorithm also differs from other GMRES derivatives

in that it allows variable preconditioning at each inner iteration. This feature enables us to, e.g., freely mix two different but complementary preconditioners, as described in [47].

In a more efficient (but less intuitive) version of the GMRES algorithm, the Givens rotations are computed and applied to the reduced system even as it is being constructed. This modification gives us an option of discontinuing the inner GMRES iterations when proper convergence criterion is met within the inner iteration loop. We use here the fact, that after j rotations applied to the reduced system, the current value of the norm $\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}\mathbf{y}\|_2$ is equal to the absolute value $|p_{j+1}|$, as shown in [45]. The final algorithm adopted for present computations is thus presented in Box 5.3.

For equation systems stemming from the increment form (3.29) of the variational formulation the initial guess $\mathbf{x}_0$ is normally taken as zero.

for $l = 1, \ldots, n_{outer}$ — *GMRES outer iterations*

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ — *compute initial residual*

$\beta := \|\mathbf{r}_0\|_2$ — *compute initial residual norm*

$\mathbf{v}_1 = \mathbf{r}_0/\beta$ — *define first Krylov vector*

$\mathbf{p} := \beta\mathbf{e}_1$ — *initialize right hand side*

for $j = 1, \ldots, m$ — *GMRES inner iteration*

$\mathbf{z}_j := \mathbf{M}_j^{-1}\mathbf{v}_j$ — *preconditioning step*

$\mathbf{w} := \mathbf{A}\mathbf{z}_j$ — *matrix-vector product*

for $i = 1, \ldots, j$ — *Gramm-Schmidt orthogonalization*

$h_{i,j} := (\mathbf{w}, \mathbf{v}_i)$

$\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_i$

$h_{j+1,j} := \|\mathbf{w}\|_2$

$\mathbf{v}_{j+1} := \mathbf{w}/h_{j+1,j}$ — *define next Krylov vector*

for $i = 1, \ldots, j-1$ — *previous Givens rotations on* $\bar{\mathbf{H}}$

$$\begin{cases} h_{i,j} := c_i h_{i,j} + s_i h_{i+1,j} \\ h_{i+1,j} := -s_i h_{i,j} + c_i h_{i+1,j} \end{cases}$$

$\gamma := \sqrt{h_{j,j}^2 + h_{j+1,j}^2}$ — *compute next rotation*

$c_j := h_{j,j}/\gamma; \quad s_j := h_{j+1,j}/\gamma$

$$\begin{cases} h_{j,j} := \gamma \\ h_{j+1,j} := 0 \end{cases}$$ — *Givens rotation on* $\bar{\mathbf{H}}$

$$\begin{cases} p_j := c_j p_j \\ p_{j+1} := -s_j p_j \end{cases}$$ — *Givens rotation on* $\mathbf{p}$

if $|p_{j+1}| \leq \varepsilon$ **exit loop** — *inner loop convergence check*

$$\begin{Bmatrix} y_1 \\ \vdots \\ y_j \end{Bmatrix} := \begin{bmatrix} h_{1,1} & \cdots & h_{1,j} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h_{j,j} \end{bmatrix}^{-1} \begin{Bmatrix} p_1 \\ \vdots \\ p_j \end{Bmatrix}$$ — *back substitution*

$\mathbf{x} := \mathbf{x}_0 + \sum_{i=1}^{j} y_i \mathbf{z}_i$ — *form approximate solution*

if $|p_{j+1}| \leq \varepsilon$ **exit loop** — *outer loop convergence check*

else $\mathbf{x}_0 := \mathbf{x}$ — *restart*

Box 5.3. GMRES algorithm: modified control flow

5.2.2 Preconditioning

In contrast to the direct methods of solution, the iterative schemes can be extremely sensitive to the particular numerical properties of the system matrix $\mathbf{A}$, and GMRES algorithm is no exception. The rate of convergence of Krylov subspace methods is influenced by the condition number of the matrix $\mathbf{A}$, as shown in [48]. The analysis contained therein indicates also that the order of convergence will degrade as eccentricity of the ellipse containing the eigenvalues of the linear system decreases. In practical terms, this measure is smaller for problems dominated by advection effects, as indicated by the Reynolds number.

In most cases a preconditioning of the original system will be needed to achieve reasonable convergence rates. Preconditioning involves a matrix $\mathbf{M}$ resembling in some sense the matrix $\mathbf{A}$, but whose inverse is easier to compute. Assuming constant preconditioner matrix for the duration of the iterative process (as mentioned earlier, variable preconditioning is also possible), the (right) preconditioned system becomes

$$\mathbf{A}\mathbf{M}^{-1}\left(\mathbf{M}\mathbf{x}\right) = \mathbf{b}. \tag{5.6}$$

The closer the matrix $\mathbf{A}\mathbf{M}^{-1}$ is to identity, the better convergence one may expect from the iterative solver. Once the solution $\mathbf{M}\mathbf{x}$ of (5.6) is obtained, the solution of the original system, i.e., the $\mathbf{x}$ vector itself, is easily computed.

Scaling of the linear system is a related concept. Here we construct a system

$$\mathbf{W}^{-1/2}\mathbf{A}\mathbf{W}^{-1/2}\left(\mathbf{W}^{1/2}\mathbf{x}\right) = \mathbf{W}^{-1/2}\mathbf{b}. \tag{5.7}$$

The matrix of the iteratively solved system becomes thus $\mathbf{W}^{-1/2}\mathbf{A}\mathbf{W}^{-1/2}$. In contrast with the one-sided preconditioning, the scaled matrix retains the symmetry characteristics of the original system.

The simplest preconditioning method uses the diagonal part of the system matrix $\mathbf{M} = \operatorname{diag}\mathbf{A}$. Such preconditioning removes the worst effects of the large variations in the magnitude of diagonal entries, which for diagonally dominant matrices is closely related to a high condition number. A diagonal scaling may be constructed for positive definite matrices with $\mathbf{W} = \operatorname{diag}\mathbf{A}$ as well. Diagonal preconditioning

or scaling become less effective as the Reynolds number increases and off-diagonal entries assume importance. The more advanced preconditioning techniques such as the Element-By-Element (EBE) [49], Cluster-Element-By-Element (CEBE) [50, 51] and CEBE/Cluster Companion (CEBE/CC) [47] methods will not be discussed here. The iterative implementations used here use the diagonal scaling exclusively.

5.3 Parallel Implementation

The solution of the linear system (5.1) typically consumes most of the computational resources in an implicit finite element program, and is also the most challenging part from the point of view of parallel implementation. The direct solution techniques are extremely hard to implement efficiently on massively parallel computers, except for the special cases of, e.g., uniformly-banded matrix with very small bandwidth of 3 to 5 elements. Such matrices rarely appear in finite element codes. Efforts are underway to create more general parallel direct solvers employing the MIMD paradigm which by nature is more flexible than the SIMD model.

In contrast, the iterative solution techniques have proven significantly more adaptable to parallel architectures. The examples in Sections 6.1 and 6.2 employ a data parallel implementation of the GMRES algorithm described previously. Building on concepts introduced in Section 3.6, we now discuss aspects of this implementation.

Bulk of the computations in the GMRES algorithm shown in Box 5.3 will involve `EQN` structures only. These include the set of Krylov vectors (original and preconditioned) and the entries of the diagonal preconditioner. The computationally intensive task of Gramm-Schmidt orthogonalization of the Krylov vectors involves only on-processor operations and communication operations of the scan/reduce type. The latter are performed very efficiently on the architectures used here, being in fact one of the fastest inter-processor communication operations. The only interplay between the `ELEM` and `EQN` occurs when the matrix vector product is to be computed. Such product involves a gather of `EQN`-level values into the `ELEM` dataset, followed by a communication-free on-processor matrix-vector product, and finally a scatter back

into the `EQN` data set.

In the current implementation all variables related to the reduced system are stored on the scalar front-end processor, and the factorization and back substitution of that system is also performed in a scalar fashion. For moderate values of the Krylov space dimension $m \leq 50$, it was observed that the solution of the reduced system consumed less than 3% of the time spent on the GMRES iterations, so the use of the comparatively slow front-end processor here is not an impediment.

See also the Ph.D. thesis of Johan [52] for another discussion of data parallel solution techniques and description of a memory-saving matrix-free GMRES implementation.