Chapter 7
Solvers for the Linear Systems of Equations

Remark 7.1. Motivation. The simulation of incompressible flow problems requires the simulation of linear problems of the form

$$\mathcal{A}\underline{x} = \begin{pmatrix} A & D \\ B & C \end{pmatrix} \begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix} = \begin{pmatrix} \underline{f} \\ \underline{g} \end{pmatrix} = \underline{y}, \tag{7.1}$$

with

$$A \in \mathbb{R}^{dN_v \times dN_v},\ D \in \mathbb{R}^{dN_v \times N_p},\ B \in \mathbb{R}^{N_p \times dN_v},\ C \in \mathbb{R}^{N_p \times N_p},$$
$$\underline{u}, \underline{f} \in \mathbb{R}^{dN_v},\ \underline{p}, \underline{g} \in \mathbb{R}^{N_p},$$

such that

$$\mathcal{A} \in \mathbb{R}^{(dN_v+N_p)\times(dN_v+N_p)}, \quad \underline{x}, \underline{y} \in \mathbb{R}^{dN_v+N_p}.$$

For the Navier–Stokes equations and in particular for time-dependent problems, systems of form (7.1) have to be solved over and over again. The efficient solution of these systems is the core of the overall efficient simulation of incompressible flow problems.

If $C = 0$, then (7.1) is a linear saddle point problem. literature, overview paper, list of notations contents of this chapter □

Remark 7.2. Sparse matrices, CSR storage format. All matrix blocks are sparse matrices such that $\mathcal{A}$ is sparse, too. Usually, sparse matrices are stored in special formats. The condensed sparse row (CSR) format is very popular. In this format, the non-zero entries are stored row-wise. It is not necessary to order them with respect to the columns. Let $\mathcal{A} \in \mathbb{R}^{m\times n}$ and let nnz be the number of its non-zero entries. Three arrays are needed to store a sparse matrix in the CSR format:

- an array of length nnz where the entries of $\mathcal{A}$ are stored,
- an array of length nnz with integers which contains the column indices of the corresponding entries of the first array,

- an array of length $(m+1)$ with integers whose i-th entry indicates where the i-th row starts in the other two arrays. The last entry of this array indicates where the $(m+1)$-st row would start.

For incompressible flow problems, a popular approach is to store each block of $\mathcal{A}$ individually in this format. □

Remark 7.3. Some properties of the system matrix. In many situations, the system matrix has a more special form compared with (7.1) or some matrix blocks have special properties. In the case of inf-sup stable finite element spaces and the Galerkin finite element method, there are $C = 0$ and $D = B^T$. The second property depends also on the boundary conditions of the concrete problem. For the Stokes equations, the matrix block A is symmetric. Usually, C is also symmetric, e.g., in the PSPG method. □

7.1 Solvers for the Coupled Problems

Remark 7.4. Sparse direct solvers. Easiest to apply are direct solvers for sparse linear systems of equations, so-called sparse direct solvers. Popular solvers are

- umfpack, Davis (2004b),
- pardiso, Schenk et al. (2008),
- MUMPS, **?**.

These solvers are particularly efficient for two-dimensional problems and small to medium ($\lesssim 500000$ degrees of freedom) problems. However, they tend to become inefficient for three-dimensional problems. This problem arises from the different sparsity structure of the matrices for discretizations of two- and three-dimensional problems. An internal degree of freedom in three dimensions possesses usually more neighbors than in two dimensions, such that there are more matrix entries and the matrices are less sparse. Sparse direct solvers try to compute a factorization of the matrices with little fill-in, which is easier for matrices where the structure is very sparse.

Direct solvers do not take advantage of situations where a good approximation of the solution is already known. This situation appears in particular in time-dependent problems, discretized with not too large time steps, where the solution from the previous discrete time or some extrapolation of solutions from previous times give usually a good approximation of the solution of the current time.

For ill-conditioned problems, it is known from numerical linear algebra that the numerical results of direct solvers might become inaccurate. Then, it is advise to apply a post processing with an iterative method for improving the accuracy. □

Remark 7.5. Iterative solvers, Krylov subspace methods. Let $\underline{r} \in \mathbb{R}^{dN_v+N_p}$, then the space

$$K_k(\underline{r},\mathcal{A}) := \text{span}\left\{\underline{r},\mathcal{A}\underline{r},\ldots,\mathcal{A}^{k-1}\underline{r}\right\}, \quad k \ge 1,$$

is called the Krylov subspace of dimension k which is spanned by $\underline{r}$ and $\mathcal{A}$.

Popular iterative solves for systems of type (7.1) can be found in the family of Krylov subspace methods. Let $\underline{x}^{(0)}$ be some initial approximation of the solution of (7.1), then the residual vector is given by

$$\underline{r}^{(0)} = \underline{y} - \mathcal{A}\underline{x}^{(0)}$$

and the Krylow subspace $K_k\left(\underline{r}^{(0)},\mathcal{A}\right)$ is considered.

- *GMRES, FGMRES.* The method GMRES (generalized minimal residual), proposed in Saad and Schultz (1986), computes the k-th iterate such that the Euclidean norm of the residual vector is minimized in $K_k\left(\underline{r}^{(0)},\mathcal{A}\right)$. The advantage of this method is that there is a control on the norm of the residual, since this norm cannot increase. However, one has to store the whole basis of $K_k\left(\underline{r}^{(0)},\mathcal{A}\right)$. These memory requirements can be afforded usually only for a rather small number k. Also, the cost per iteration (number of floating point operations) increases with each iteration. In practice, one prescribes a maximal number of iterations, often of the order $10,\ldots,100$. If the method did not converge after k steps, it is stopped and restarted with the current iterate. This strategy is called GMRES with restart, GMRES(restart).
  As it will be discussed in Remark **??**, one has to apply usually a preconditioner for accelerating the speed of convergence. If this preconditioner is not a fixed matrix but a numerical method, then one should use the flexible GMRES methods proposed in Saad (1993). The application of a method can be represented by a matrix, which is usually unknown, but this matrix might change from iteration to iteration. The flexible GMRES method can cope with changes in the preconditioner during the iteration. However, one has to store in the flexible GMRES method the bases of two Krylov spaces, which increases the memory requirements further.
- *MINRES.* todo
- *CGS.* The main idea of CGS (conjugate gradient squared), proposed in Sonneveld (1989) consists in computing the k-th iterate such that it is orthogonal to $K_k\left(\underline{r}^{(0)},\mathcal{A}^T\right)$. CGS is a realization of this approach which does not need the knowledge of $\mathcal{A}^T$. This method can be implemented with a so-called short recurrence, i.e., one has to store only the three check last arrays of the basis of the Krylov space. Hence, the memory requirements are much less compared with GMRES. However, the Euclidean norm of the residual vector is not minimized and it can happen that this norm increases hugely during the iteration. In this sense, an irregular convergence behavior is possible.
- *BiCGStab.* The BiCGStab (bi-conjugate gradient stabilized) method relies on the same principle as the CGS method. It is a generalization of

the CGS method that was developed in van der Vorst (1992) to stabilize the CGS method.

To accelerate the convergence of iterative solvers, one has to apply so-called preconditioners, see Section 7.2. □

Remark 7.6. Strategies for solving (7.1) *iteratively.* For the Navier–Stokes equations, system of form (7.1) arise in the Picard or Newton iteration, see Section 5.3. From experience it is known that it is often inefficient to solve (7.1) to high accuracy if an iterative solver is used. For this reason, one prescribes as stopping criteria a small number of maximal iterations or a small reduction factor for the Euclidean norm of the residual vector, e.g., 10 to gain one digit. After having satisfied the stopping criterion, with usually only few iterations, one proceeds with the next Picard or Newton step. □

7.2 Preconditioners for Iterative Solvers

generalities

7.2.1 Preconditioners Treating Velocity and Pressure in a Decoupled Way

Remark 7.7. Motivation. todo □

Remark 7.8. A factorization of the system matrix, Schur complement matrix. Let in (7.1) $D = B^T$ and $C = 0$, then a straightforward calculation shows that

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & -BA^{-1}B^T \end{pmatrix} \begin{pmatrix} I & A^{-1}B^T \\ 0 & I \end{pmatrix}. \tag{7.2}$$

The matrix

$$S = -BA^{-1}B^T \tag{7.3}$$

is called Schur complement matrix or pressure Schur complement matrix. This matrix is generally not explicitly available. Since it contains the factor A^{-1}, it is usually not a sparse matrix. □

Example 7.9. Semi-Implicit Method for Pressure-Linked Equations (SIMPLE). SIMPLE is a classical preconditioner which was introduced in Patankar and Spalding (1972). Its principal approach is as follows:

- An approximation of the pressure is assumed to be known, e.g., from the previous iteration.
- Then, the velocity is computed from the momentum equation, see (7.7). This velocity will usually not satisfy the discrete continuity equation since

the used pressure is only an approximation of the pressure which corresponds to the new velocity.
- Finally, the velocity and the pressure are corrected such that the discrete continuity equation is satisfied, see (7.8) and (7.9).

From the point of view of linear algebra, SIMPLE relies on the factorization

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix} = \begin{pmatrix} A & 0 \\ B & -BA^{-1}B^T \end{pmatrix} \begin{pmatrix} I & A^{-1}B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix} = \begin{pmatrix} \underline{f} \\ \underline{g} \end{pmatrix}, \tag{7.4}$$

where the first and second factor in (7.2) are multiplied. The factorization (7.4) is of the form of a block LU factorization. The SIMPLE method is obtained by the approximation

$$A^{-1} \approx (\operatorname{diag}(A))^{-1} = A_{\text{diag}}^{-1}.$$

The approximation of the Schur complement matrix is denoted by

$$S_{\text{d}} = -BA_{\text{diag}}^{-1}B^T. \tag{7.5}$$

Since B^T represents a discrete gradient operator and B a discrete divergence operator, the matrix S_{d} can be thought of representing a discrete diffusion operator with non-constant diffusion. The non-constant diffusion is given by A_{diag}^{-1}. Thus, one iteration of SIMPLE solves the following system

$$\begin{pmatrix} A & 0 \\ B & S_{\text{d}} \end{pmatrix} \begin{pmatrix} I & A_{\text{diag}}^{-1}B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} \underline{u} \\ \underline{p} \end{pmatrix} = \begin{pmatrix} \underline{f} \\ \underline{g} \end{pmatrix}.$$

In practice, one can compute with one step of the iteration directly the new iterate or one can compute just updates for the old iterate. Here, the version with the updates will be presented. Given the iterates at iteration m, then the ansatz for the next iterates is

$$\begin{pmatrix} \underline{u}^{(m+1)} \\ \underline{p}^{(m+1)} \end{pmatrix} = \begin{pmatrix} \underline{u}^{(m)} \\ \underline{p}^{(m)} \end{pmatrix} + \omega \begin{pmatrix} \underline{\delta u} \\ \underline{\delta p} \end{pmatrix}, \tag{7.6}$$

where $\omega \in (0, 1]$ is some damping factor. Inserting this ansatz in (7.4) gives

$$\omega \begin{pmatrix} A & 0 \\ B & -BA^{-1}B^T \end{pmatrix} \begin{pmatrix} I & A^{-1}B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} \underline{\delta u} \\ \underline{\delta p} \end{pmatrix} = \begin{pmatrix} \underline{f} \\ \underline{g} \end{pmatrix} - \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \underline{u}^{(m)} \\ \underline{p}^{(m)} \end{pmatrix} = \begin{pmatrix} \underline{r}_{\text{u}} \\ \underline{r}_{\text{p}} \end{pmatrix}.$$

Now, SIMPLE proceeds as follows:
- Solve
$$\begin{pmatrix} A & 0 \\ B & S_{\text{d}} \end{pmatrix} \begin{pmatrix} \underline{\delta u}^* \\ \underline{\delta p}^* \end{pmatrix} = \begin{pmatrix} \underline{r}_{\text{u}} \\ \underline{r}_{\text{p}} \end{pmatrix},$$
  i.e., one has to solve the first

$$A\underline{\delta u}^* = \underline{r}_{\mathrm{u}} \tag{7.7}$$

and then

$$S_{\mathrm{d}}\underline{\delta p}^* = \underline{r}_{\mathrm{p}} - B\underline{\delta u}^*. \tag{7.8}$$

- Solve
$$\begin{pmatrix} I & A_{\mathrm{diag}}^{-1}B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} \underline{\delta u} \\ \underline{\delta p} \end{pmatrix} = \begin{pmatrix} \underline{\delta u}^* \\ \underline{\delta p}^* \end{pmatrix},$$
which is nothing else than to set first
$$\underline{\delta p} = \underline{\delta p}^* \tag{7.9}$$
and then to set
$$\underline{\delta u} = \underline{\delta u}^* - A_{\mathrm{diag}}^{-1}B^T\underline{\delta p}.$$
- Compute the new iterate with (7.6).

interpretation as block Gauss–Seidel possible? □

Remark 7.10. Concerning SIMPLE. SIMPLE is easy to implement, which makes it attractive. It relies on the already assembled matrix blocks. Only the approximation S_{d} of the Schur complement matrix given in (7.5) has to be computed. This matrix couples pressure degrees of freedom which are usually not coupled in finite element approximations of the diffusion operator, but S_{d} is still a sparse matrix.

The efficiency of SIMPLE depends on how good A^{-1} is approximated by A_{diag}^{-1}. It is known, e.g., from Elman et al. (2006), check that the efficiency is bad for convection-dominated problems since the diagonal of A does not contain sufficient information about the convective term. In addition, SIMPLE is known to perform poor when the spatial mesh size becomes small. reference □

Example 7.11. Least Squares Commutator (LSC) preconditioner. The inefficient behavior of SIMPLE comes from the fact that the approximation (7.5) of the Schur complement is usually bad for convection-dominated problems. In Elman et al. (2006), a preconditioner is proposed which contains information about the convection, the so-calles Least Squares Commutator (LSC) preconditioner.

Starting point is the multiplication of the second and third factor of (7.2) yielding

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & S \end{pmatrix}.$$

Taking the inverse of the last matrix gives

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & S \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix}.$$

This representation leads to the idea that

$$\begin{pmatrix} A & B^T \\ 0 & S \end{pmatrix}^{-1} \tag{7.10}$$

might be a good right-oriented preconditioner. more details In order to apply an approximation of this preconditioner, one has to approximate the Schur complement matrix S.

The basic idea of the LSC approach consists in the approximation

$$A\left(D_{\text{lsc}}^{-1}B^T\right) \approx B^T A_{\text{pres}}, \tag{7.11}$$

where A_{pres} represents a discretization of a convection-diffusion operator for the pressure. The diagonal matrix D_{lsc} with positive diagonal entries is a scaling matrix. Taking for the moment D_{lsc} to be the identity, then the left-hand side of (7.11) represents the the discretization of a convection-diffusion operator for the velocity applied to a discrete gradient and the right-hand side the discrete gradient applied to a convection-diffusion operator for the pressure. If the equal sign would hold in (7.11), there would be a commutation of a convection-diffusion operator and a gradient operator.

Multiplying (7.11) with $-BA^{-1}$ from left and A_{pres}^{-1} from right gives, compare (7.3),

$$S = -BA^{-1}B^T \approx -BD_{\text{lsc}}^{-1}B^T A_{\text{pres}}^{-1} = S_{\text{lsc}}.$$

Now, A_{pres} is determined by minimizing the commutation error from (7.11) in a weigthed norm

$$\min_{A_{\text{pres}}} \left\| AD_{\text{lsc}}^{-1}B^T - B^T A_{\text{pres}} \right\|_{D_{\text{lsc}}^{-1}}^2, \tag{7.12}$$

where the norm is induced by the inner product

$$\left(\underline{x}, \underline{y}\right)_{D_{\text{lsc}}^{-1}} = \underline{x}^T D_{\text{lsc}}^{-1} \underline{y}.$$

Thus, the normal equation corresponding to the least squares problem (7.12) is given by

$$BD_{\text{lsc}}^{-1}\left(A\left(D_{\text{lsc}}^{-1}B^T\right) - B^T A_{\text{pres}}\right) = 0.$$

It follows that

$$A_{\text{pres}} = \left(BD_{\text{lsc}}^{-1}B^T\right)^{-1}\left(BD_{\text{lsc}}^{-1}AD_{\text{lsc}}^{-1}B^T\right).$$

Inserting this expression into (7.3) gives the approximation for the Schur complement matrix

$$S_{\text{lsc}} = -BD_{\text{lsc}}^{-1}B^T\left(BD_{\text{lsc}}^{-1}AD_{\text{lsc}}^{-1}B^T\right)^{-1}BD_{\text{lsc}}^{-1}B^T. \tag{7.13}$$

In the application of the preconditioner, see (7.10), one has to compute the inverse of S_{lsc}. From the representation (7.13) one can see that the computation of this inverse requires the solution of two discrete diffusion problems with the same matrix $BD_{\mathrm{lsc}}^{-1}B^T$. In Elman et al. (2006) it is proposed to use the diagonal of the velocity mass matrix, see (**??**), as D_{lsc}. check $\square$

AMG, Stokes with penalty method, Uzawa, Elman et al.