Solutions to the problems

Solutions to Chapter 1

Solution 1.1.

(a) The second definition is not correct. Notice, e.g., that according to that definition the pseudospectra would be a closed set. The definition is corrected by replacing $\leq$ with $<$, namely

$$\sigma_\varepsilon := \left\{ z \in \mathbb{C} \text{ such that } z \in \sigma(A+E) \text{ with } E \in \mathbb{C}^{n\times n} \text{ and } \|E\| < \varepsilon \right\}.$$

(b) Let us denote by $\sigma_\varepsilon^{(1)}(A)$ and $\sigma_\varepsilon^{(2)}(A)$ the pseudospectra according to the first and second definition. Observe that $\sigma_\varepsilon^{(1)}(A) \cap \sigma_\varepsilon^{(2)}(A) \neq \emptyset$ since if $z \in \sigma(A)$ then one can trivially verity that $z \in \sigma_\varepsilon^{(1)}(A) \cap \sigma_\varepsilon^{(2)}(A)$. We therefore assume that $z \notin \sigma(A)$. We start showing that $\sigma_\varepsilon^{(1)}(A) \subseteq \sigma_\varepsilon^{(2)}(A)$. Let $z \in \sigma_\varepsilon^{(1)}(A) \setminus \sigma(A)$, by definition of matrix norm, there exist $v, u \in \mathbb{C}^n$ such that $\|v\| = \|u\| = 1$ and $(z-A)^{-1}v = \alpha^{-1}u$. This implies that $zu - Au = \alpha v$. We can construct $E = \alpha v w^H$, with $w^H u = 1$, such that $Eu = \alpha v$ and $||E|| = \alpha < \varepsilon$ [1]. Hence, it exists $E \in C^{n\times n}$ such that $\|E\| < \varepsilon$ and $zu - Au = Eu$, which implies $z \in \sigma(A+E), ||E|| < \varepsilon$, and therefore we conclude that $z \in \sigma_\varepsilon^{(2)}(A)$. We now show that $\sigma_\varepsilon^{(2)}(A) \subseteq \sigma_\varepsilon^{(1)}(A)$. Let us assume $z \in \sigma_\varepsilon^{(2)}(A) \setminus \sigma(A)$. By definition there exists $E \in \mathbb{C}^{n\times n}$ such that $\|E\| < \varepsilon$ and $z \in \sigma(A+E)$. Therefore there exists $v \in \mathbb{C}^n$ such that $\|v\| = 1$ and $(z-A)v = Ev$ and than $v = (z-A)^{-1}Ev$. By using the Hölder's inequality $1 = \|v\| \leq \|(z-A)^{-1}\|\|E\|$ that directly implies $\|(z-A)^{-1}\| > \|E\|^{-1} > \varepsilon^{-1}$, and therefore we conclude that $z \in \sigma_\varepsilon^{(1)}(A)$.

Solution 1.2. By the second definition in Problem 1.1

$$\sigma_\varepsilon(A) := \left\{ z \in \mathbb{C} \text{ such that } z \in \sigma(A+E) \text{ with } E \in \mathbb{C}^{n\times n} \text{ and } \|E\| < \varepsilon \right\}$$

Let us denote by $\tilde{\sigma}_\epsilon(A)$ the definition of pseudospectra given in this problem. We show that $\sigma_\varepsilon(A) \subseteq \tilde{\sigma}_\epsilon(A)$. Let $z \in \sigma_\varepsilon(A)$, then there exists $E \in \mathbb{C}^{N\times N}$ with $\|E\| < \epsilon$ such that $z \in \sigma(A+E)$. Let v be the corresponding eigenvector such that $\|v\| = 1$. We have $(A+E)v = zv$ that can be written as $(zI - A)v = Ev$. Hence $\|(zI-A)v\| = \|Ev\| \leq \|E\|\|v\| = \|E\| < \varepsilon$. Thus for $z \in \sigma_\epsilon(A)$ we have that $\|(zI-A)v\| < \epsilon$ for some $0 \neq v \in \mathbb{C}^N$ with $\|v\| = 1$, namely $z \in \tilde{\sigma}_\epsilon(A)$. The other inclusion $\tilde{\sigma}_\epsilon(A) \subseteq \sigma_\varepsilon(A)$ is trivial.

Solution 1.3. Observe that the matrix K is the stiffness matrix and not the "eliminated stiffness matrix", therefore it may be singular or ill-conditioned.

[1] If $||\cdot||$ is the 2-norm, then $w = u^H$. For other arbitrary norms, the existence of w is equivalent to the existence of a corresponding linear functional L on $\mathbb{C}^n$ such that $L(u) = 1$ and $\|L\| = 1$, which is guaranteed by the Hahn–Banach theorem.

(a) The 50 largest and 50 smallest eigenvalues can be computed in MATLAB with `eigs`. The smallest eigenvalues have absolute value proportional to machine precision, i.e., the matrix K is numerically singular. The largest eigenvalues are contained in the interval $[0.0027, 0.0522]$ and are not clustered. Therefore, Corollary 1.4 gives a very slow convergence rate since there are eigenvalues extremely close to zero and the largest eigenvalue is 0.0522. By directly using Theorem 1.2, since the eigenvalues in $[0.0027, 0.0522]$ are not clustered, we expect the convergence to happen, at least after 51 iterations. This lower bound on the number of steps is obtained by considering, in Theorem 1.2, the polynomial that is has one root in zero and the other roots correspond to the 50 largest eigenvalues of K.

(b) The linear system $(K - I)x = b$ can efficiently be solved with GMRES. The matrix $(K - I)$ has the same eigenvalues as K shifted with -1. Thus, given the eigenvalue spread above, we can conclude that the eigenvalues of $(K - I)$ will be spread very close to -1. With the disc reasoning in Corollary 1.4 we then get that $\frac{\|r_m\|}{\|r_0\|} \leq \left(\frac{\rho}{c}\right)^m$ with $\rho \leq 0.06$ and $c = 1$. Thus in at most 7 iterations the error is below 10^{-8}. In Figure 1.4 we plot the convergence of a 100 iterations with GMRES for K, $(K - I)$, and $(K - 0.03I)$. The last one also gives good convergence, since the eigenvalues are still clustered, but not too much around zero.

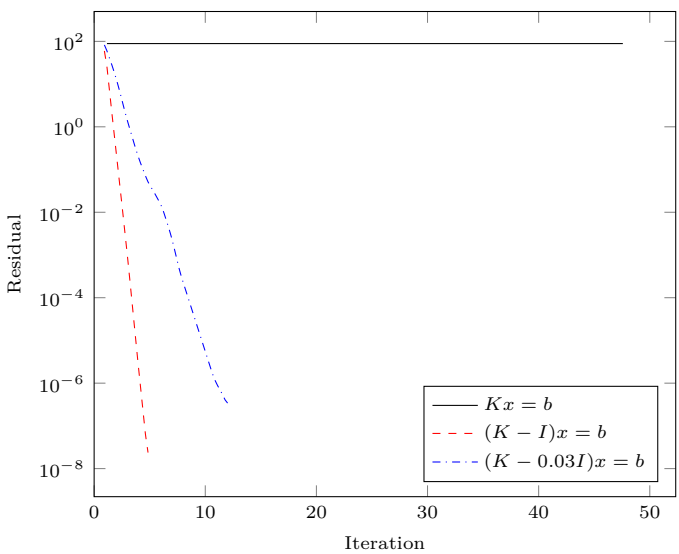

Figure 1.4: GMRES applied to shifted systems.

We see the poor convergence for K. But for the shifted linear system we have good convergence (although there exists some shifts, like for example $(K - 0.01I)$ that gives worse, but perhaps acceptable, convergence).

Solution 1.4.

(a) We start by showing that $c\sigma_\epsilon(A) \subseteq \sigma_{|c|\varepsilon}(cA)$. By definition $c\sigma_\epsilon(A) = \{z = cx \mid x \in \sigma_\epsilon(A)\}$. Therefore, let $z \in c\sigma_\epsilon(A)$ we can express $z = cx$ with x such that $\|(xI - A)^{-1}\| > \epsilon^{-1}$. This implies that z fulfills $\|(\frac{z}{c} - A)^{-1}\| > \epsilon^{-1}$, which can be further simplified to $\|(z - cA)^{-1}\| > (|c|\epsilon)^{-1}$. Thus we conclude that $z \in \sigma_{|c|\epsilon}(cA)$. We now show that $\sigma_{|c|\varepsilon}(cA) \subseteq c\sigma_\epsilon(A)$. Let $z \in \sigma_{|c|\varepsilon}(cA)$ we have $\|(z - |c|A)^{-1}\| > (|c|\epsilon)^{-1}$ that can be rewritten as $\|(\frac{z}{c} - A)^{-1}\| > \epsilon^{-1}$ and then we conclude $z \in c\sigma_\epsilon(A)$.

(b) Theorem 2.2 in [258] states that if the norm is the 2-norm, then the equality is true if and only if A is normal. The equality obviously fails for the simplest type of Jordan blocks of size 2x2. Consider

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

which has a double eigenvalue in 1. Perturb it with

$$E = \begin{pmatrix} 0 & 0 \\ \varepsilon & 0 \end{pmatrix}.$$

Then we can see that $\sigma(A + E) = \{1 \pm \sqrt{\varepsilon}\}$, which for an $\varepsilon < 1$ is outside of $\sigma(A) + D_\varepsilon$ because in this case $\sqrt{\varepsilon} > \varepsilon$. Therefore we conclude that $\sigma(A + E) \not\subseteq \sigma(A) + D_\varepsilon$. We now prove the inclusion $\sigma(A) + D_\varepsilon \subseteq \sigma_\varepsilon(A)$. We consider the following equivalent definition of pseudo-spectra, see Problem 1.1,

$$\sigma_\varepsilon(A) := \left\{z \in \mathbb{C} \text{ such that } z \in \sigma(A + E) \text{ with } E \in \mathbb{C}^{n\times n} \text{ and } \|E\| \le \varepsilon\right\}.$$

Let consider $z \in \sigma(A) + D_\varepsilon$. We can express $z = \lambda + \delta$ where $\lambda \in \sigma(A)$ and $|\delta| < \varepsilon$. Observe that, for the specific choice $E = \delta I$, we have that $\|E\| < \varepsilon$ and $\lambda + \delta \in \sigma(A + E) \subseteq \sigma_\varepsilon(A)$.

c) This is not true. Consider the 2-norm, and A and B normal. Then by the above cited theorem $\sigma_\varepsilon(A) = \sigma(A) + D_\varepsilon$ and $\sigma_\varepsilon(B) = \sigma(B) + D_\varepsilon$. Thus $\sigma_\varepsilon(A) + \sigma_\varepsilon(B) = \sigma(A) + D_\varepsilon + \sigma(B) + D_\varepsilon = \sigma(A + B) + 2D_\varepsilon = \sigma(A + B) + D_{2\varepsilon}$ which is not equal to $\sigma_\varepsilon(A + B) = \sigma(A + B) + D_\varepsilon$.

In other words, one could say that when constructing $\sigma_\varepsilon(A)$ and $\sigma_\varepsilon(B)$ separately, one can do a maximal and constructive perturbation in the eigenvalue corresponding to the same eigenvector for both A and B and thus get a larger total perturbation in the sum, than one could do on the matrix $A + B$.

Solution 1.5. This problem contains a very important message: the pseudospectra contain information about the eigenvectors. Two matrices may have the same eigenvalues but different pseudospectra and in general, the sensitivity of the eigenvalues is described in function of the eigenvectors, see [140, Ch. 7.2]. Therefore, it is not enough to know the spectrum of a matrix in order to fully describe the behavior of GMRES.

(a) In Figure 1.5 is displayed the pseudospectra of A for $n = 12$ whereas in Figure 1.6 the pseudospectra of S.

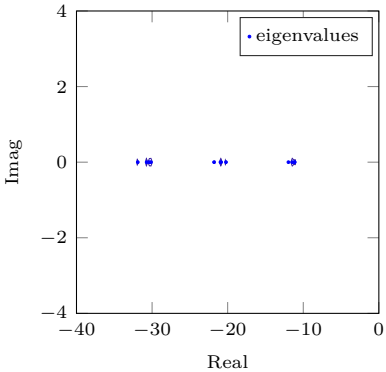

Figure 1.5: $\sigma_\varepsilon(A)$ for $n = 12$.

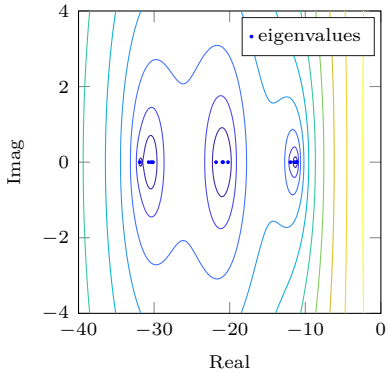

Figure 1.6: $\sigma_\varepsilon(S)$ for $n = 12$.

In Figure 1.7 is displayed the pseudospectra of A for $n = 21$ whereas in Figure 1.8 the pseudospectra of S.

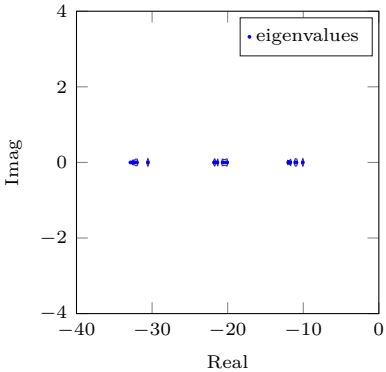

Figure 1.7: $\sigma_\varepsilon(A)$ for $n = 21$.

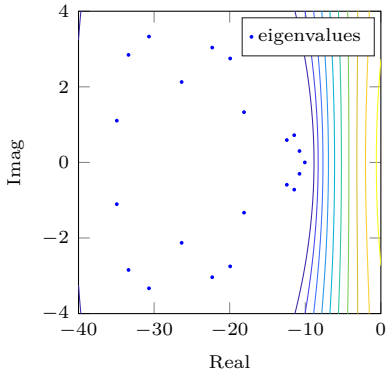

Figure 1.8: $\sigma_\varepsilon(S)$ for $n = 21$.

Notice that the eigenvalues of A and S are numerically different (even if by construction they are mathematically the same). This is due to the fact that the eigenvalues of S are too sensitive to small changes (ill-conditioned) and MATLAB is not able to compute them in a stable way.

(b) Even if the matrix S is now constructed in a more stable way, all the reasoning of the point (a) is still valid. We now consider $n = 66$.

In Figure 1.9 is displayed the pseudospectra of A generated as

```
pscont(A, 4, 500, [-80,0,-50,50],-10:1:1);)
```

whereas in Figure 1.10 the pseudospectra of S generated as

```
pscont(S, 4, 500, [-80,0,-50,50],-10:1:1);).
```

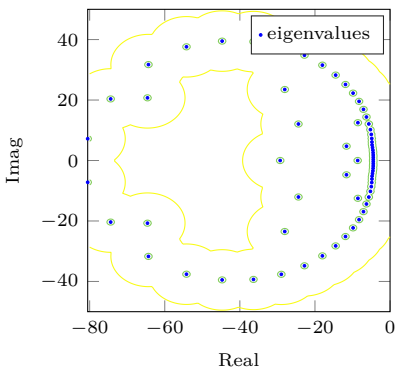

Figure 1.9: $\sigma_\varepsilon(A)$ for $n = 66$.

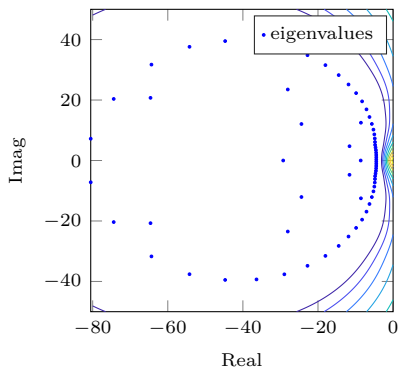

Figure 1.10: $\sigma_\varepsilon(S)$ for $n = 66$.

In Figure 1.11 we have the convergence of GMRES for A and in Figure 1.12 for S.

Clearly, for the matrix S, GMRES does not converge even if A and S have the same eigenvalues. This is explained by the reasoning in (a). The convergence of GMRES does not depend only on the eigenvalues but also on their sensitivity, which is characterized by the eigenvectors). By using the theory presented in the chapter, we can use the pseudospectra bound for the convergence of GMRES we have

$$\frac{\|r_m\|}{\|r_0\|} \leq \frac{L_\varepsilon}{2\pi\varepsilon} \min_{p \in \mathcal{P}_0^m} \max_{\Gamma_\varepsilon} |p(z)|$$

We can clearly see that for the matrix S the constant L_ε is very large, therefore the transient phase will be very long.

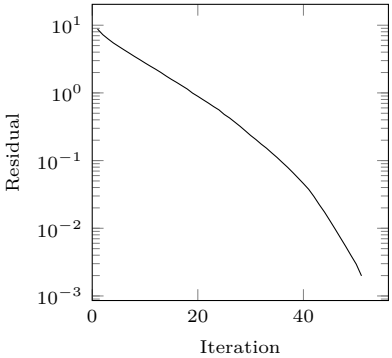

Figure 1.11: GMRES applied to A.

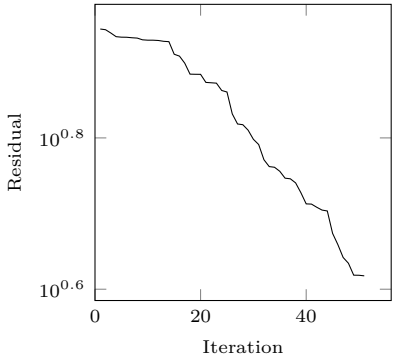

Figure 1.12: GMRES applied to S.

A short answer to this second part can be given by using the standard theory for the convergence of GMRES. More precisely we have

$$\frac{\|r_m\|}{\|r_0\|} \leq \kappa(V_A) \min_{p \in \mathcal{P}_0^m} \max_{\lambda \in \sigma(A)} |p(z)|$$

Where V_A is the matrix containing as columns the eigenvectors of A. We now compute the condition numbers $\kappa(V_A) = 1$ and $\kappa(V_S) = 1.7572e + 16$

Conclusion: for GMRES we have that the asymptotic convergence is characterized by the spectrum whereas the transient phase is depends on the sensitivity of the eigenvalue (that is related to the eigenvectors) and it is characterized by the pseudospectra.

Solution 1.6.

(a) Let $A = Z\Lambda Z^{-1}$ be an eigendecomposition. Since b is as a linear combination of exactly m eigenvectors, we can write $b = Zw$ where

$$w_i = \begin{cases} w_i \neq 0, & 1 \leq i \leq m \\ 0, & m+1 \leq i \leq n \end{cases}$$

Note that we have assumed, without loss of generality, that the m eigenvectors we refer to occur as the first m columns of Z.

Hence, by using the RHS-bound in Theorem 1.5 we have,

$$\frac{||r_m||}{||r_0||} \leq ||Z|| \min_{p \in P_m^0} \left(\sum_{i=1}^{m} |w_i|^2 |p(\lambda_i)|^2 \right)^{1/2}.$$

From the above inequality we can make the upper bound zero by choosing p to be a polynomial with at most m distinct roots/eigenvalues $\lambda_i, \quad i = 1, \ldots, m$, which we are allowed to do since

$$p \in \mathcal{P}_m^0.$$

Hence, we have $0 \leq ||r_m||/||r_0|| \leq 0 \implies r_m = 0$. This means that GMRES will converge after a maximum of $m = 10$ iterations.

(b) Easy extension of the argument in (a).

Solution 1.7.

(a) We are given that A is diagonalizable as $A = Z\Lambda Z^{-1}$ and $Zw = b/||b||$. Hence, substituting for A and b, we get,

$$p(A)b = p(Z\Lambda Z^{-1})Zw||b|| = Zp(\Lambda)w||b|| \tag{1.21}$$

Letting $W = ||b|| \operatorname{diag}(w) \implies w||b|| = We$, where e is a vector of all ones. This leads to

$$Zp(\Lambda)w||b|| = Zp(\Lambda)We = ZWp(\Lambda)e \tag{1.22}$$

Since multiplication of diagonal matrices is commutative. The combination of (1.21) and (1.22) completes the proof.

(b) By replacing, in the minimization property of GMRES, $p(A)b$ with the equivalent expression computed in (a), we have,

$$||r_m|| \leq \min_{p \in \mathcal{P}_m^0} ||ZWp(\Lambda)e|| \leq ||Z|| \min_{p \in \mathcal{P}_m^0} ||Wp(\Lambda)e||$$

Note that $Wp(\Lambda)e$ is a vector whose i-th element is $\|b\|w_i p(\lambda_i)$, where w_i is the coefficient of the i-th eigenvector when $b/||b||$ is written as a linear combination of the eigenvectors, i.e., columns of Z.

(c) The derivation in the paper is different in the following way:

Instead of assuming $ZW = b/||b||$, the assumption is $ZW = r_0/||r_0||$, where r_0 is the initial residual. (Note that here $r_0 = b$ if $x_0 = 0$.)

Solution 1.8.

(a) The fast (red) curve relates to the structured right-hand side vector

```
b=zeros(n,1); b(50:60)=1;
```

and the slow (black curve) relates to the random RHS.

The matrix A is a diagonalizable matrix (it is diagonal and hence trivially diagonalizable). The eigenvectors are the corresponding unit vectors e_i. By considering Theorem 2.2 in [256], the GMRES bound can be written as $\frac{\|r_m\|}{\|r_0\|} \leq \min_{q\in\Pi_m q(0)=1} \left(\sum_{i=1}^n |w_i|^2|q(\lambda_i)|^2\right)^{1/2}$, where Π_m is the set of polynomials of degree max m, and w_i is the the projection coefficient of b (the RHS) onto the i-th basis vector in the eigenvector basis. For this problem, that means that w_i is simply the i-th component in the b, i.e., $w_i = b_i$. With that in mind, we can, for the structured b, rewrite the bound as $\frac{\|r_m\|}{\|r_0\|} \leq \min_{q\in\Pi_m q(0)=1} \left(\sum_{i=50}^{60} |q(\lambda_i)|^2\right)^{1/2}$. Thus full convergence is guaranteed in only 11 iterations. However, with a disc-reasoning based on this bound we can say that $\frac{\|r_m\|}{\|r_0\|} \leq \left(\frac{\rho}{c}\right)^m$, where we can now take $\rho = 5$, and $c = 55$, since the only eigenvalues involved in the last sum are $50, 51, \ldots, 60$. Thus an error of 10^{-6} is reached in at most 6 iterations, which is exactly what is observed in the figure.

This type of reasoning is not possible for a random RHS since in general we will have $w_i \neq 0$ for all i. Some eigenvalues can be "down prioritized", but that is still not giving the same acceleration and simple bounds to compute. A simplified bound could be argued for by using 10 iterations to "remove" the eigenvalues 1-10. Then a disc-argument on the rest would give something like $\rho = 40$, and $c = 50$, which would give that after $10 + 40 = 50$ iterations the error would be bounded by $1.33 \cdot 10^{-4}$. This is simplified behavior is not exactly what is observed, but it goes somewhat along with the observed convergence.

(b) If the RHS is changed (to `b=zeros(n,1); b(90:100)=1;`) the convergence will be faster. One can do a similar argument as the one in a), both in terms of only regarding some eigenvalues, and also the disc-reasoning. However, this time the disc would be $\rho = 5$, and $c = 95$ and thus providing an error of 10^{-6} in at most 5 iterations. Compare the convergence rates $5/95 \approx 0.053$ and $5/55 \approx 0.091$.

Solution 1.9. By construction of the algorithm we have that $z_j = M_j v_j$ (cf Problem 1.10). Therefore, if $M_j = M$ we can write $Z_m = MV_m$. By replacing this expression in the Arnoldi-like factorization, we get

$$AZ_m = V_{m+1}\underline{H}_m$$
$$AMV_m = V_{m+1}\underline{H}_m$$

Observe that this is an Arnoldi factorization for the preconditioned matrix AM.

Solution 1.10.

(a) Modifying the given code in this problem we come up with the solution

```
n=200; e = ones(n,1);
A = spdiags([e 2*e e], -1:1, n, n);

b=rand(n,1); norm_b=norm(b);
V(:,1)=b/norm_b;
m=30;TOL = -inf;
M=@(j,b) BiCGSTABL(A,b,TOL,j);

for j=1:m
 Z(:,j)=M(j,V(:,j));
 V(:,j+1)=A*Z(:,j);
 h=V(:,1:j)'*V(:,j+1);
 V(:,j+1)=V(:,j+1)-V(:,1:j)*h;
 g=V(:,1:j)'*V(:,j+1);
 V(:,j+1)=V(:,j+1)-V(:,1:j)*g;
 H(1:j,j)=h+g;
 H(j+1,j)=norm(V(:,j+1));
 V(:,j+1)=V(:,j+1)/H(j+1,j);
 e1=eye(j+1); e1=e1(:,1);
 y=H\(norm_b*e1);
 xx=Z(:,1:j)*y;
 err(j)=norm(A*xx-b);
end
semilogy(err);
```

The convergence is illustrated in Figure 1.13 in the curve (a).

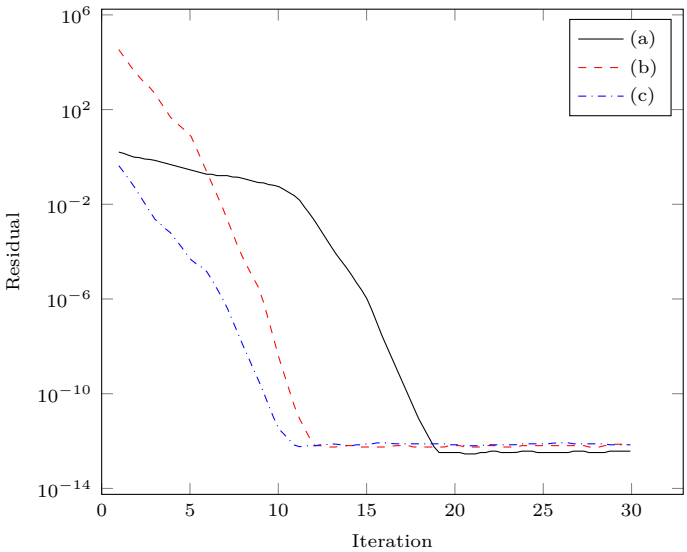

Figure 1.13: Converge of Flexible GMRES with BiCGSTABL as preconditioner.

(b) The change in the script is minor. The only difference is that the preconditioner is now defined by
`M=@(j,b) BiCGSTABL(A,b,TOL,m-j+1);`
and the convergence changes as illustrated in Figure 1.13 in the curve (b).

(c) Once again the change to the code is minor, and the preconditioner is defined by

```
M=@(j,b) BiCGSTABL(A,b,TOL,m);
```

and the convergence changes as illustrated in Figure 1.13 in the curve (c). We notice that the change from (b) is only minor, convergence is like one or two iterations faster.

(d) A comparison with the two plots in (a) and (b) indicate that the convergence is improved by starting out with a better preconditioner, rather than using a good preconditioner in the end. The intuition is that a good preconditioner will help capture the "good" subspace to search for the solution, and if that is done in a better way early in the process, then a "better" subspace will be built.

Further tests with with the FEM-matrix

```
nn = 50;
A = gallery('wathen',nn,nn);
n = size(A,1);
```

and the random matrix

```
n = 200;
rng(123456789)
A = gallery('uniformdata',n,n) - 4.5*eye(n,n);
```

yeild similar results.

Solution 1.11. The pseudocode for PCG is given in Algorithm 3. We now consider $M = LL^T$ where L is the Cholesky factor. We explicitate M in the expression for z_j:

$$z_{j+1} = M^{-1}r_{j+1} = (LL^T)^{-1}r_{j+1} = L^{-T}L^{-1}r_{j+1} = L^{-T}\tilde{r}_{j+1}$$

and from this we make explicit the inner products of β_j:

$$\beta_j = \frac{(r_{j+1}, z_{j+1})}{(r_j, z_j)} = \frac{(r_{j+1}, L^{-T}L^{-1}r_{j+1})}{(r_j, L^{-T}L^{-1}r_j)} = \frac{(L^{-1}r_{j+1}, L^{-1}r_{j+1})}{(L^{-1}r_j, L^{-1}r_j)} = \frac{(\tilde{r}_{j+1}, \tilde{r}_{j+1})}{(\tilde{r}_j, \tilde{r}_j)}$$

and of α_j: we want $(Ap_j, p_j) = (\tilde{A}\tilde{p}_j, \tilde{p}_j)$, where $\tilde{A} = L^{-1}AL^{-T}$.

$$(Ap_j, p_j) = (AL^{-T}L^Tp_j, L^{-T}L^Tp_j) = (L^{-1}AL^{-T}L^Tp_j, L^Tp_j) = (\tilde{A}\tilde{p}_j, \tilde{p}_j)$$

Now we can rewrite the steps of CG:

- $\alpha_j = \frac{(\tilde{r}_j, \tilde{r}_j)}{(\tilde{A}\tilde{p}_j, \tilde{p}_j)}$,
- By multiplying from the left by L^T,
  $L^T x_{j+1} = L^T x_j + \alpha_j L^T p_j \quad \Rightarrow \quad \tilde{x}_{j+1} = \tilde{x}_j + \alpha_j \tilde{p}_j$.
- By multiplying from the left by L^{-1},
  $r_{j+1} = r_j - \alpha_j A p_j \quad \Rightarrow \quad \tilde{r}_{j+1} = \tilde{r}_j - \alpha_j L^{-1} A L^{-T} L^T p_j = \tilde{r}_j - \alpha_j \tilde{A}\tilde{p}_j$
- By multiplying from the left by L^T,
  $p_{j+1} = z_{j+1} + \beta_j p_j \quad \Rightarrow \quad \tilde{p}_{j+1} = L^T L^{-T} \tilde{r}_{j+1} + \beta_j \tilde{p}_j = \tilde{r}_{j+1} + \beta_j \tilde{p}_j$

This is the CG corresponding to the system $\tilde{A}\tilde{x} = L^{-1}b$, where $\tilde{x} = L^T x$. In fact the starting conditions is
$\tilde{r}_0 = L^{-1} r_0 = L^{-1} b - L^{-1} A x_0 = L^{-1} b - L^{-1} A L^{-T} L^T x_0 = L^{-1} b - \tilde{A}\tilde{x}$

Solution 1.12. Let us assume (λ, x) is an eigenpair of $M^{-1}A$. Then,

$$M^{-1}Ax = \lambda x \Leftrightarrow AM^{-1}(Mx) = \lambda(Mx)$$
$$\Leftrightarrow Ax = \lambda LUx \Leftrightarrow L^{-1}AU^{-1}(Ux) = \lambda(Ux)$$

Hence, $M^{-1}A$, AM^{-1} and $L^{-1}AU^{-1}$ have identical eigenvalues with eigenvectors x, Mx and Ux respectively.

Solution 1.13. We now point how to modify a FeNICS script for solving a PDE in a way that the matrix and RHS is exported in a MATLAB format. Three script examples in FeNICS, that can be used to reverse-engineering the whole process and generalize to other PDEs, are available.

It is needed to add in the Python script the following lines (in the beginning):

- `# to avoid reordering/permutations of the matrices`
  `parameters['reorder_dofs_serial'] = False`
- `# to save the matrices`
  `from scipy.io import savemat`
- `parameters['linear_algebra_backend'] = "Eigen"`

After this, you can use FeNICS to define the PDE. Then you have to construct the stiffness matrix, convert it to CSR data, extract the sparse array format and save it in a format that MATLAB can read. In compact, this is done with the following lines of code that you have to add at the end of the script

```
A, b = assemble_system(a, L, bc);
rows,cols,vals = as_backend_type(A).data()  Get CSR data!
A = as_backend_type(A).sparray()!
b = as_backend_type(b).get_local()!
savemat('Poisson.mat', {'A': A,'b':b})!
```

We now propose few example to illustrated how to create such matrices. The files `poisson_membrane.py`, `poisson_membrane_2.py` and `elesticity.py` are scripts in Python that generate a .mat file that you can load with MATLAB. The .mat file contains matrix and RHS of the linear systems derived by a FEM discretization of the associated PDE.

- Example 1: a standard Poisson problem `poisson_membrane.py`
- Example 2: a Poisson problem with a more challenging domain (square with an hole in the middle) `poisson_membrane_2.py`
- Example 3: a 2d elasticity problem `elesticity.py`

If you do not have installed Fenics and you just want to see/test the matrices, just load the files `Poisson.mat, Poisson_2.mat, Elasticity.mat`

Solution 1.14. Here is the procedure to export to MAT-files. Start the LiveLink framework of COMSOL (for communication with MATLAB-process) by running:

```
$ module add comsol # Necessary at KTH
$ comsol server
Username: Myname
Password: Not important
Repeat password: Not important
```

Username and password can be chosen as you like.

Start MATLAB and run

```
>> addpath('/usr/local/comsol55/multiphysics/mli');
>> mphstart
>> mphlaunch;
>> model=mphopen('blabla.mph')
>> MM=mphmatrix(model,'sol1','out',{'K'});
>> size(MM.K)
>> K=MM.K;
>> save('/tmp/myfile.mat','K')
```

Observe: you may need to change `addpath` with the path where COMSOL was installed. The matrix `capacitor_tunable_piezo_domain_K.mat` (K-matrix) of the MEMS device described here (`https://www.comsol.com/model/tunable-mems-capacitor-123`) can generated following the procedure above. A slight variation applied to the heat conductor problem gave: `disk_stack_heat_sink_52a.mat`. We now consider the linear system with the "eliminated stiffness matrix" from `satellite.mph` and a random right-hand side. We test the iterative solvers GMRES and CG (observe that K is SPD).

```
model = mphopen ('satellite.mph');
MM=mphmatrix(model,'sol1','out',{'Kc'});
% Kc=eliminated Stiffness Matrix
K=MM.Kc;

n=length(K); b=rand(n,1);
```

```
[X,FLAG,RELRES,ITER,RESVEC] = gmres(K,b,[],-Inf,200);
[X,FLAG,RELRES,ITER,RESVEC2] = cgs(K,b,-Inf,200);

semilogy(RESVEC); hold on
semilogy(RESVEC2)
```

The converge diagrams are illustrated in Figure 1.14.

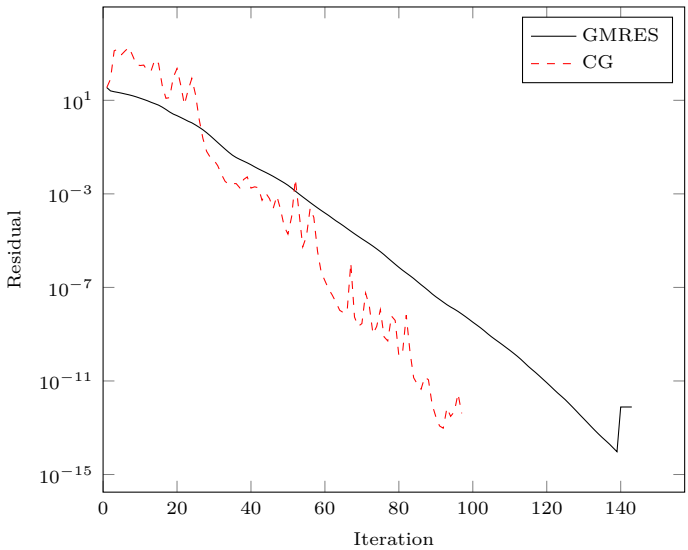

Figure 1.14: Convergence of GMRES and CG applied to the eliminated stiffness matrix from `satellite.mph`.

Solution 1.15.

(a) By using (1.2) and that $r_m \in \mathcal{K}_m(A,b)$ we get

$$\|r_m\| = \min_{x\in\mathcal{K}_m(A,b)} \|Ax-b\| = \min_{p\in\mathcal{P}_m^0} \|p(A)r_0\| \le \min_{p\in\mathcal{P}_m^0} \|p(A)\|\|r_0\|$$

(b) By combining the previous result with the Cauchy integral formula we get

$$\frac{\|r_m\|}{\|r_0\|} \min_{p\in\mathcal{P}_m^0} \|p(A)\| \le \min_{p\in\mathcal{P}_m^0} \frac{1}{2\pi} \left\| \oint_{\Gamma_\varepsilon} p(z)(zI-A)^{-1}dz \right\|.$$

We now use the well known inequality $|\oint_\Gamma f(z)| \le L \max_{z\in\Gamma}$ where L is the length of Γ and we get

$$\frac{\|r_m\|}{\|r_0\|} \le \frac{1}{2\pi} \min_{p\in\mathcal{P}_m^0} \max_{z\in\Gamma_\varepsilon} \|p(z)\| \max_{\Gamma_\varepsilon} \|(zI-A)^{-1}\|$$

We conclude by using the definition of pseudospectra, see Problem 1.1, that gives $\max_{\Gamma_\varepsilon} \|(zI - A)^{-1}\| < 1/\varepsilon$.

Solution 1.16. By using (1.2) we have

$$\|r_m\| = \min_{p \in \mathcal{P}_m^0} \|p(A) r_0\|.$$

From the above expression, by dividing by r_0, and with several algebraic manipulations we get

$$\begin{aligned}
\frac{\|r_m\|}{\|r_0\|} &\leq \min_{p \in \mathcal{P}_m^0} \|p(A) w\| = \min_{p \in \mathcal{P}_m^0} \|K K^{-1} p(A) K K^{-1} w\| \\
&\leq \|K\| \|K^{-1} w\| \min_{p \in \mathcal{P}_m^0} \|K^{-1} p(Z \Lambda Z^{-1}) K\| \\
&\leq \|K\| \|K^{-1} w\| \min_{p \in \mathcal{P}_m^0} \|(Z^{-1} K)^{-1} p(\Lambda) (Z^{-1} K)\| \\
&\leq \|K\| \|K^{-1} w\| \operatorname{cond}(K^{-1} Z) \min_{p \in \mathcal{P}_m^0} \|p(\Lambda)\| \\
&\leq \|K\| \|K^{-1} w\| \operatorname{cond}(K^{-1} Z) \min_{p \in \mathcal{P}_m^0} \max_{\lambda \in \sigma(A)} |p(\lambda)|.
\end{aligned}$$

Solution 1.17. Let $e_m \in \mathbb{C}^n$ be the m-th vector of the canonical basis, the orthogonal basis of the Krylov space and the matrix H_m fulfill the following relation, often referred to as "Arnoldi factorization"

$$A Q_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^H$$

Let (θ, z) an eigenpair of H_m then we have

$$\begin{aligned}
A Q_m z &= Q_m H_m z + h_{m+1,m} q_{m+1} e_m^H z \\
A Q_m z &= \theta Q_m z + h_{m+1,m} q_{m+1} e_m^H z.
\end{aligned}$$

By using that the orthogonality of Q_m, i.e., $Q_m^H Q_m = I$, we get

$$\begin{aligned}
A Q_m z &= \theta Q_m z + h_{m+1,m} v_{m+1} e_m^H Q_m^H Q_m z \\
(A - h_{m+1,m} q_{m+1} e_m^H V_m^H) Q_m z &= \theta Q_m z
\end{aligned}$$

The last equation tells us that θ is an eigenvalue, with eigenvector $Q_m z$, of a perturbation of the matrix A. More precisely is an eigenvalue of $A + E$ with $E = h_{m+1,m} q_{m+1} e_m^H Q_m^H$. By using that Q_m is orthogonal we get $\|E\| = \|h_{m+1,m} q_{m+1} e_m^T Q_m^H\| = h_{m+1,m}$, i.e., $\theta \in \sigma_{H_m}(A)$.

Solution 1.18. We consider the following equivalent definition of pseudospectra [258, pag. 16] which is easly deducible from the first definition in Problem 1.1

$$\sigma_\varepsilon(A) = \{z \in \mathbb{C} \ \text{ s.t. } \ \exists\, v \in \mathbb{C},\ \|v\| = 1,\ \|(zI - A)v\| < \varepsilon\}.$$

Let $z \in \sigma_\varepsilon(A)$, we want to prove that this vector can be written as $z = y + x$ where $y \in W(A)$ and $\|x\| < \varepsilon$, i.e., $x \in D_\varepsilon$. From the definition, there exists a vector v with unitary norm such that $\|(zI - A)v\| < \varepsilon$. Let us define $w := v^H(zI - A)v$, then

$$w = zv^H v - v^H Av = z\|v\|_2^2 - v^H Av = z - v^H Av$$

hence $z = w + v^H Av = w + u$ where $\|w\|_2 \leq \|v^H\|_2 \|(zI - A)v\|_2 < \varepsilon$, i.e., $w \in D_\varepsilon$, and $u \in W(A)$. This directly implies $\sigma_\varepsilon(A) \subseteq W(A) + D_\varepsilon$.

Solution 1.19. The condition $P \approx A^{-1}$ is a sufficient condition for P to be a good preconditioner, but it is not necessary. There are good preconditioners that do not fulfill this property. Let us assume $P \not\approx A^{-1}$. The matrix P is a good preconditioner if $P^{-1}A$ have eigenvalues inside a disc, with a small radius, that does not intersect the origin. Relate this to the Corollary 1.4. Or more in general, P is a good preconditioner if $P^{-1}A$ have eigenvalues clustered around few points. Relate this to Theorem 1.2.

Solution 1.20. The preconditioned matrix is $\tilde{A} = P^{-1}A = P^{-1}MP$ and it holds

$$\sum_{j=0}^{m} \tilde{A}^j = \sum_{j=0}^{m} P^{-1}M^j P = P^{-1}\left(\sum_{j=0}^{m} M^j\right) P = I,$$

namely $q(\tilde{A}) = 0$ with $q(\lambda) = 1 - \sum_{j=1}^{m} \lambda^j$. The thesis follows by (1.2), i.e., due to the minimization property of GMRES, the residual can be expressed as

$$\|r_m\| = \min_{p \in \mathcal{P}_m^0} \|p(\tilde{A})r_0\| \leq \|q(\tilde{A})r_0\| = 0.$$

Solutions to Chapter 2

Solution 2.1. Let consider the following split of the matrix A given by $A = M - N$. It is possible to construct the iterative methods by observing that the linear system $Ax = b$ can written as $Mx - Nx = b$. More precisely, the idea of the iterative methods discussed in this chapter consists of the using the previous equation for generating a sequence of vectors as $Mx^{k+1} = Nx^k + b$.

(a)
- The Jacobi iteration uses the splitting $A = D - C$, where D is the diagonal and C is the rest of the matrix. Then $M_J = D$, and $N_J = C$.
- The Gauss–Seidel iteration uses the splitting $A = D - L - U$, where D is the diagonal and L, U are the lower and upper strictly triangular parts of the matrix. Then $M_{GS} = D - L$, and $N_{GS} = U$.
- The Successive Over-Relaxation method uses the splitting $A = D - L - U$, for the scaled problem $\omega Ax = \omega b$. The splitting used is $\omega M_{SOR} = D - \omega L$, and $\omega N_{SOR} = (1 - \omega)D + \omega U$.

(b) The convergence of these methods can be studied by seeing these as fixed point iterations. The generic method is of the form (if M is invertible)

$$x^{k+1} = M^{-1}Nx^k + M^{-1}b \tag{2.23}$$

which is a fixed point iteration to which the fixed point is a solution to the original linear system according that the discussion above. Let us denote by $x^\star$ the fixed point, the error can be computed by iteratively using (2.23) as follows

$$\begin{aligned} x^\star - x^{k+1} &= M^{-1}Nx^\star + M^{-1}b - M^{-1}Nx^k - M^{-1}b \\ &= (M^{-1}N)(x^\star - x^k) \\ &= \dots \\ &= (M^{-1}N)^k(x^\star - x^0) \end{aligned}$$

This tells us that the error goes to zero if and only if $\rho(M^{-1}N) < 1$, i.e., $(M^{-1}N)^k \to 0$ for $k \to \infty$.

For Jacobi iteration, this condition is fulfilled if, e.g., the matrix is diagonally dominant. For Gauss–Seidel, this condition is fulfilled if, e.g., the matrix is symmetric and positive definite. The Successive Over-Relaxation is a modification of Gauss–Seidel, created to achieve faster convergence, so the conditions on the convergence are the same as Gauss–Seidel.

(c) We prove the claim by induction. By assumption we have $x_{-1} = 0$, then $x_0 = M^{-1}Nx_{-1} + M^{-1}b = M^{-1}b$ and $y_0 = z_0 = M^{-1}b$, hence $y_0 = x_0$, i.e., the base of the induction is verified. Let us assume that $x_k = y_k$ and we perform the induction step.

$$\begin{aligned} x_{k+1} &= M^{-1}Nx_k + M^{-1}b = M^{-1}Ny_k + M^{-1}b \\ &= M^{-1}N\left(\sum_{i=0}^{k} z_i\right) + M^{-1}b \\ &= M^{-1}b + \sum_{i=0}^{k} M^{-1}Nz_i = z_0 + \sum_{i=0}^{k} z_{i+1} = \sum_{i=0}^{k+1} z_i = y_{k+1} \end{aligned}$$

Solution 2.2.

(a) We consider $A = M - N$, where M is the diagonal part of A and $-N$ has the same entries of A except for zeros in the diagonal. Then

$$\left[M^{-1}N\right]_{i,j} = \begin{cases} a_{i,j}/a_{i,i} & i \neq j \\ 0 & i = j \end{cases}$$

and therefore, by using the diagonally dominance property of A, we have for $i = 1, \ldots, n$

$$\sum_{j=1}^{n} |[M^{-1}N]_{i,j}| = \sum_{\substack{j=1 \\ j\neq i}}^{n} \left| \frac{a_{i,j}}{a_{i,i}} \right| = \frac{1}{|a_{i,i}|} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}| < 1$$

Given the above reasoning, and by the Gershgorin localization theory, see [159, Corollary 6.1.3], we conclude from the that $M^{-1}N$ has all its eigenvalues located in a circle centered at 0 and with radius less than 1. This in turn implies that $\rho(M^{-1}N) < 1$, which implies that the Jacobi iteration converges for diagonally dominant matrices.

(c) Consider

$$\underbrace{\begin{pmatrix} 1 & 3 \\ 0.1 & 1 \end{pmatrix}}_{=:A} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{:=M} - \underbrace{\begin{pmatrix} 0 & -3 \\ -0.1 & 0 \end{pmatrix}}_{:=N}$$

then A is not diagonally dominant, but $\rho(M^{-1}N) < 1$, i.e., Jacobi converges.

Solution 2.3.

(a) A regular splitting of the matrix A is given by a pair of matrices M and N such that M is invertible, M^{-1} and N have nonnegative elements and $A = M - N$.

(b) An M-matrix can be defined in many equivalent ways. One of such ways is the following. The matrix A is an M-matrix is invertible and A^{-1} has non-negative elements.

(c) Theorem 4.4 in [237] tells us that if A is an M-matrix then a regular splitting $A = M - N$ is such that $\rho(M^{-1}N) < 1$, in particular the iterative method $Mx_{k+1} = Nx_k + b$ is convergent.

Solution 2.4. The following is the required MATLAB code

```
n=10000; e=ones(n,1);
A=spdiags([e e -5*e e e], -2:2, n, n);
b=sin(pi*(1:n)');

D=diag(diag(A)); E=-tril(A,-1); F=-triu(A,1);
xs = A\b; x0=rand(n,1);

% Jacobi
k = 1; x = x0;
eJ(k) = norm(xs-x);
while(eJ(k)>1e-5)
 k = k+1;
```

```
 x = D\((E+F)*x+b);
 eJ(k)=norm(xs-x);
end

% Gauss-Seidel
k=1; x=x0;
eGS(k) = norm(xs-x);
while(eGS(k)>1e-5)
 k = k+1;
 x = (D-E)\(F*x+b);
 eGS(k) = norm(xs-x);
end

% hybrid
k=1; x=x0;
err(k) = norm(xs-x);
while(err(k)>1e-5)
 k = k+1;
 if rem(k,2)==0
     % Jacobi
     x=D\((E+F)*x+b);
 else
     % Gauss-Seidel
     x=(D-E)\(F*x+b);
 end
 err(k) = norm(xs-x);
end

semilogy(eJ); hold on
semilogy(eGS)
semilogy(err)
legend("J","GS","J+SG");
```

The convergence of these methods is illustrated in Figure 2.4.

Solution 2.5.

(a, b) If the iterations converge to a vector v, this vector fulfills $v = -Bv+a$, namely is the solution to the linear system $(I + B)v = a$. We can therefore test the convergence by measuring the residual at the i-th iteration as $\|v_i + Bv_i - a\|$. The iterations converge to the solution for the first 8 iterations, then the residual grows again. See Figure 2.5

(c) Theorem 11.2.1 in [140] states that the iterates converges for any starting vector v_0 if and only if $\rho(-B) < 1$. The eigenvalues of B are $1/10, 1/2$ and 10 therefore we do not expect converge for a generic starting vector. Let us denote by v the fixed point, i.e., $v = -Bv + a$, then it is possible to show by induction that $v_{k+1} - v = -B\,(v_k - v)$. By using this relation is then clear that, if v and v_0 are linear combination of the eigenvectors associated with the eigenvalues with module smaller then one, the iterations converge.

Solution 2.6. The convergence of Jacobi and Gauss–Seidel for the given problem and starting vector $x_0 = [1, \ldots, 1]$ is illustrated in Figure 2.6. If we use as

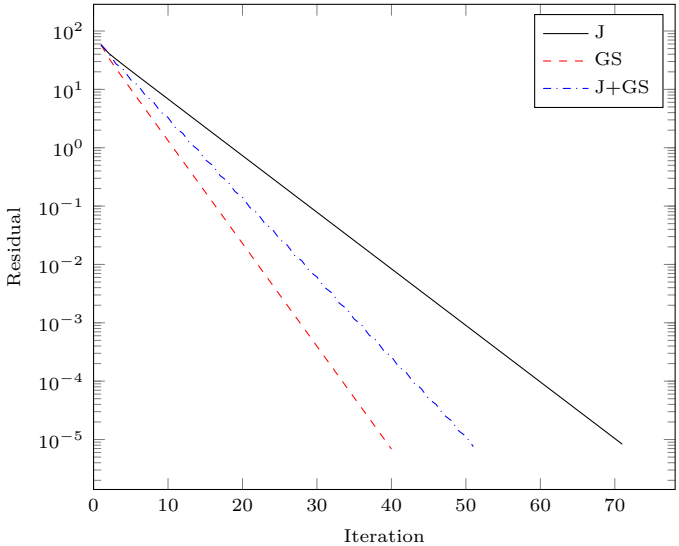

Figure 2.4: Convergence for Jacobi, Gauss–Seidel and hybrid approach in Problem 2.4.

starting vector $x_0 = b$ both method converge in the first iteration. Indeed $x = b$ is a good approximation to the solution to the linear system, more precisely $\|Ab - b\| \approx 10^{-13}$.

Solution 2.7.

(a) The fixed point iteration (2.8) with the splitting $A = M_1 - N_1$ corresponds to the Jacobi method.

(b) The matrix A is diagonally dominant and therefore Jacobi converges. See Problem 2.2.

(c) The convergence of the two splittings is illustrated in Figure 2.7. Observe that the second splitting has a faster convergence in terms of iteration count, however each iteration is computationally more expensive with respect to the first splitting. Indeed, in the second splitting M_2 is tridiagonal and a linear system with this matrix has to be solved at each iteration.

(d) It is possible to verify that $A = M_2 - N_2$ is a regular splitting and therefore fixed point iteration (2.8) converges.

Solution 2.8. The forward SOR half-step is $x_{k+1/2} = (D - \omega E)^{-1}(\omega F + (1 - \omega)D)x_k + \omega(D - \omega E)^{-1}b$ and the following backward half-step is $x_{k+1} = (D -$

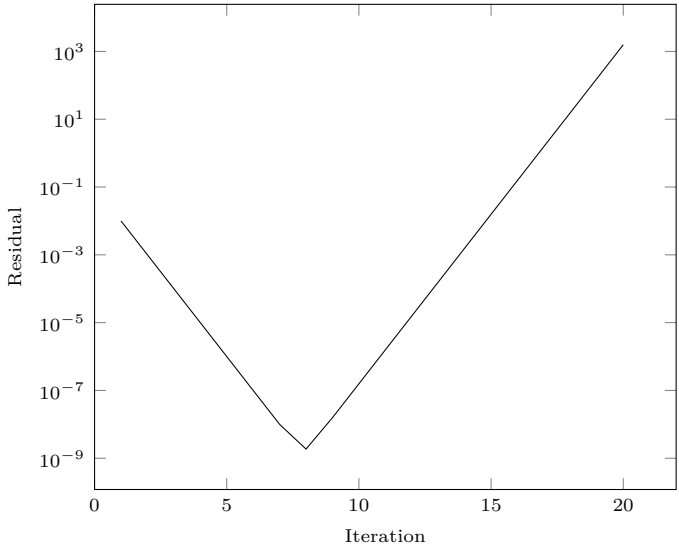

Figure 2.5: Convergence iteration in Problem 2.5

$\omega F)^{-1}(\omega E+(1-\omega)D)x_{k+1/2}+\omega(D-\omega F)^{-1}b$. By combining these two equations we get

$$x_{k+1} = (D-\omega F)^{-1}(\omega E + (1-\omega)D)x_{k+1/2} + \omega(D-\omega F)^{-1}b.$$

Reordering the terms we can be expressed as

$$\begin{aligned} x_{k+1} =& (D-\omega F)^{-1}(\omega E + (1-\omega)D)(D-\omega E)^{-1} \\ & (\omega F + (1-\omega)D)x_k + \omega(D-\omega F)^{-1} \\ & \left[(\omega E + (1-\omega)D)(D-\omega E)^{-1} + I\right] b. \end{aligned}$$

which is equivalent to $x_{k+1} = G_\omega x_k + f_\omega$ where

$$\begin{aligned} G_\omega &= (D-\omega F)^{-1}(\omega E + (1-\omega)D)(D-\omega E)^{-1}(\omega F + (1-\omega)D), \\ f_\omega &= \omega(D-\omega F)^{-1}\left[(\omega E + (1-\omega)D)(D-\omega E)^{-1} + I\right] b. \end{aligned}$$

By using that $x_{k+1} = M^{-1}Nx_k + M^{-1}b$ we get $M^{-1}b = f_\omega$ and therefore $M^{-1} = \omega(D-\omega F)^{-1}\left[(\omega E + (1-\omega)D)(D-\omega E)^{-1} + I\right]$. By further manipulating this expression we can reduce it to $M^{-1} = \omega(2-\omega)(D-\omega F)^{-1}D(D-\omega E)^{-1}$ and therefore $M = \frac{1}{\omega(2-\omega)}(D-\omega F)D^{-1}(D-\omega E)$.

Solution 2.9. (a) A central difference scheme for the final PDE results in the

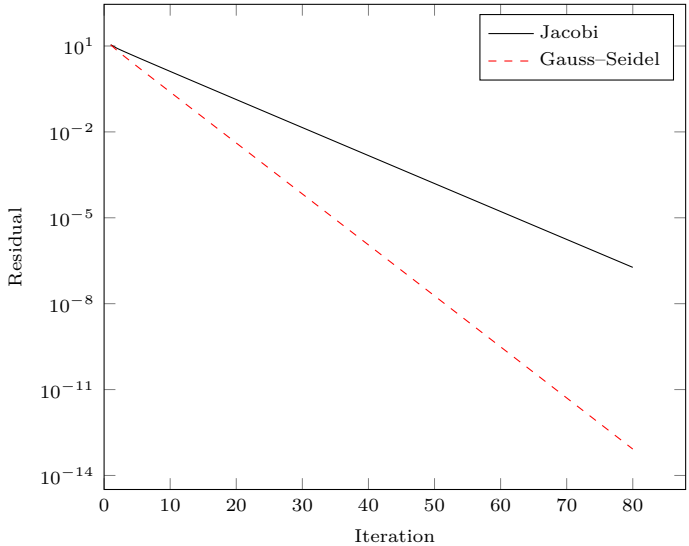

Figure 2.6: Convergence iteration in Problem 2.6 for the starting vector $x_0 = [1, \ldots, 1]$.

following finite difference equations

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + k(x_i, y_i)u_{i,j} = 1$$

for $i, j = 1, \ldots, n$. If we assume row major ordering, then the first term corresponds to a diagonally repeating tridiagonal matrix structure $D \in \mathbb{R}^{n\times n}$, corresponding to the 1-D stencil $[1/h^2, -2/h^2, 1/h^2]$. The second term corresponds to a sum of a diagonal matrix and two off-diagonal matrices, which can be seen as a repeating stretched D. Hence, in matrix form, the sum of the first two terms is a pentadiagonal matrix which can be expressed with the Kronecker products as $M = D \otimes I_n + I_n \otimes D$ and can be computed in MATLAB as `M=kron(D,I)+kron(I,D)`. The third term is just a diagonal matrix obtained by vectorizing the discretization of the function $k(x, y)$, namely $N = \mathrm{vec}(K)$ where $K = [k(x_i, y_j)]_{i,j=1}^n$.

(b)
```
n=100; e = ones(n,1);
xx=linspace(-1,1,n); yy=linspace(-1,1,n); h=xx(2)-xx(1);
D = spdiags([e -2*e e]/(h^2),[-1 0 1],n,n);
I = speye(n);
M = kron(D,I)+kron(I,D);
kk=1e5; k=@(x,y) kk*(x.^2+y.^2<1/10);
f=@(x,y) ones(size(x));
```

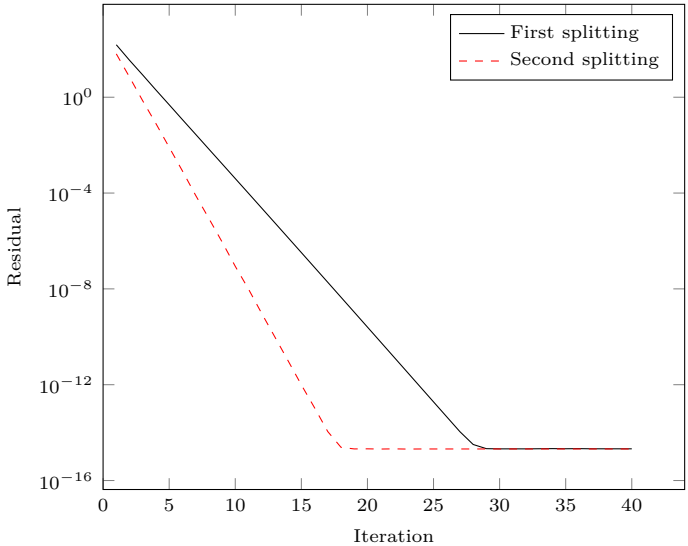

Figure 2.7: Convergence iteration in Problem 2.7 with random starting guess.

```
[XX,YY] = meshgrid(xx,yy); KK=k(XX,YY);
N = -spdiags(KK(:), 0, n^2, n^2);
F= f(XX,YY);
u=(M-N)\F(:); uu=reshape(u,n,n);
imagesc(xx,yy,uu); colorbar
```

In Figure 2.8 is illustrated the numerical solution to the PDE for $n = 100$ and $k = 10^5$.

(c) With the setting of (b) the iterative method $Mx_{k+1} = Nx_k + b$ does not converge since $\rho(M^{-1}N) > 1$. However, if the discretization paramenter n is large enough and the penalty value $\tilde{k}$ is not too large, the splitting is convergent.

(d) It is a trivial change of the previous part.

Solution 2.10. The following script is the implementation of the required iterative methods. Observe that the first splitting corresponds to the Jacobi method.

```
n=1000; e = ones(n,1);
A = spdiags([e 5*e e], -1:1, n, n); A = A^2;
b=rand(n,1);

x0=rand(n,1); mm=100;

% splitting 1
```

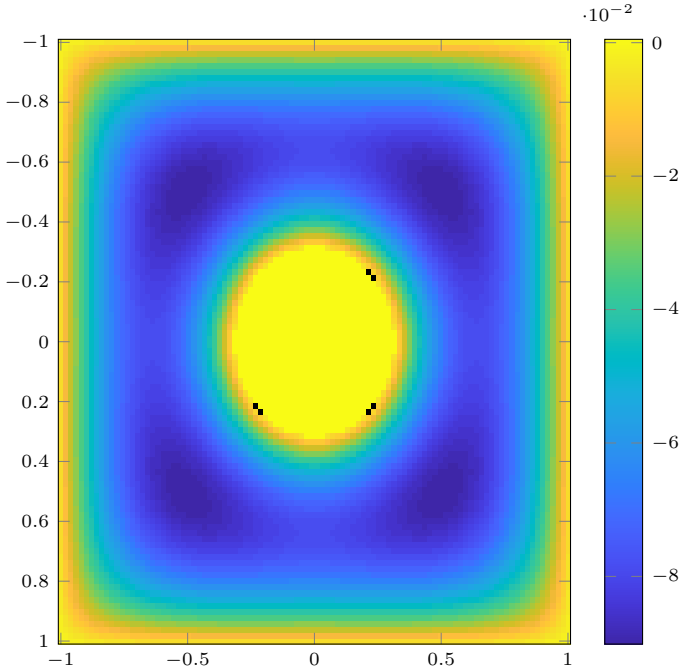

Figure 2.8: Numerical solution to the PDE in problem 2.9.

```
M=diag(diag(A)); N=M-A;
norm(M\N,1)
x = x0;
err1(1)=norm(A*x-b);
for j=1:mm
 x = M\(N*x+b);
 err1(j+1)=norm(A*x-b);
end

% splitting 2
M=diag(diag(A))+diag(diag(A,-1),-1)+diag(diag(A,1),1); N=M-A;
norm(M\N,1)
x = x0;
err2(1)=norm(A*x-b);
for j=1:mm
 x = M\(N*x+b);
 err2(j+1)=norm(A*x-b);
end

semilogy(err1); hold on
semilogy(err2)
```

The convergence of the two iterative methods is illustrated in Figure 2.9. The matrix A is pentadiagonal, we expect better performance from the second splitting since the matrix M_2 better approximate the matrix A. This is also confirmed by the fact that $\|M_1^{-1}N_1 \approx 0.8162$ whereas $\|M_2^{-1}N_2 \approx 0.2857$.

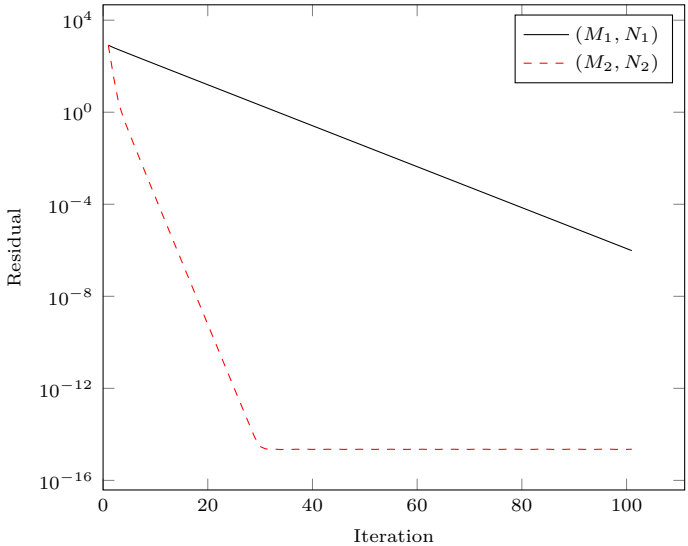

Figure 2.9: Convergence the two iterative methods in Problem 2.10.

Solution 2.11.

(a)
```
function [ S ] = sparsity_pattern( AA )
 S = AA>0;
end
```

(b)
```
function [L, U] = sparse_lu(A)
n = size(A, 1); L = eye(n);
for k = 1 : n
 AA=A(k:n,k:n); % select the proper submatrix
 [~,idx]=sort(sum(sparsity_pattern(AA)),2, 'ascend');
 AA=AA(idx,idx);
 A(k:n,k:n)=AA;
 L(k + 1 : n, k) = A(k + 1 : n, k) / A(k, k);
 for l = k + 1 : n
 A(l, :) = A(l, :) - L(l, k) * A(k, :);
 end
end
U = A;
end
```

(c) The matrix A has the sparsity pattern in Figure 2.10. The sparsity pattern of L factor and U factor for the sparse LU approach is shown in Figure 2.11 and Figure 2.12. The sparsity pattern of L factor and U factor for the classic LU approach is shown in Figure 2.13 and Figure 2.14. By using the naive sparse LU-factorization we are roughly able to reduce the number of non-zero elements, with respect to the standard LU factorization, by half.

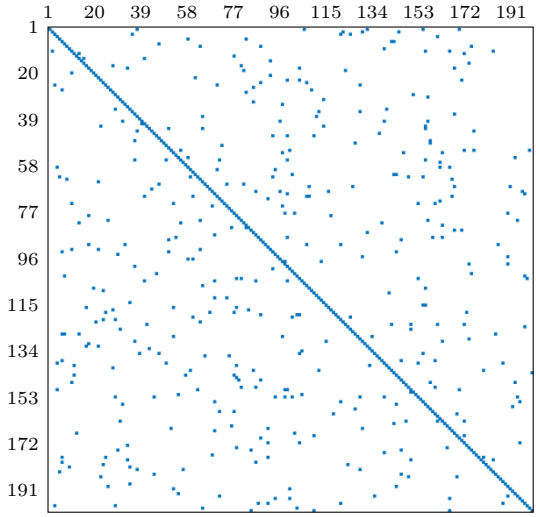

Figure 2.10: Random matrix.

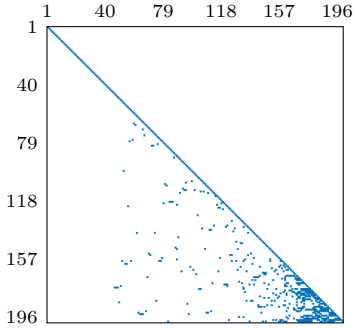

Figure 2.11: Sparse L factor.

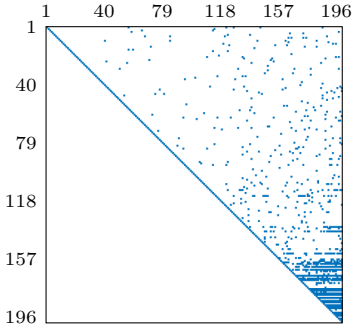

Figure 2.12: Sparse U factor.

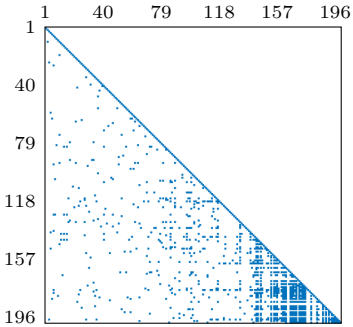

Figure 2.13: L factor.

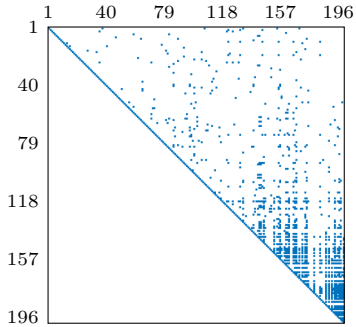

Figure 2.14: U factor.

(d) The matrix A is an arrow matrix, i.e., has the sparsity pattern in Figure 2.15. The sparsity pattern of L factor and U factor for the sparse LU approach is shown in Figure 2.16 and Figure 2.17. The sparsity pattern of L factor and U factor for the classic LU approach is shown in Figure 2.18 and Figure 2.19. By using the naive sparse LU-factorization we are roughly able to reduce the number of non-zero elements, with respect to the standard LU factorization, by half.

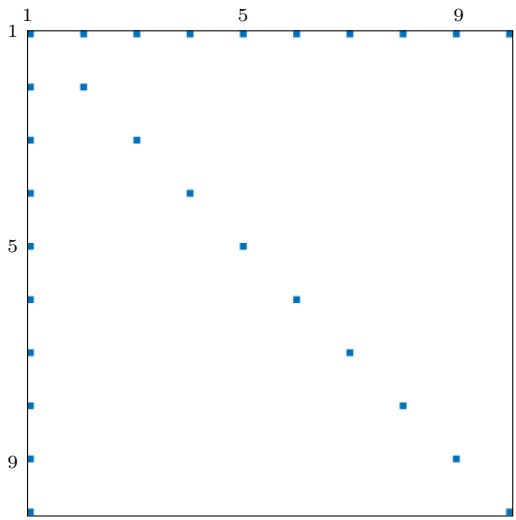

Figure 2.15: Random arrow matrix.

Solution 2.12. (a) Fill-in elements are zero elements of A that become nonzero

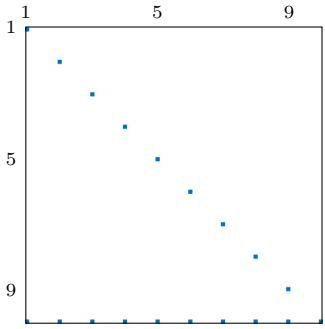

Figure 2.16: Sparse L factor.

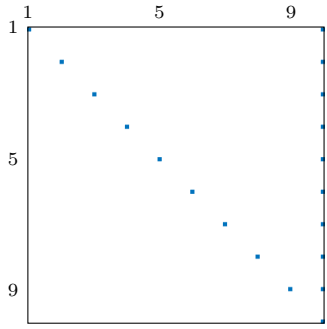

Figure 2.17: Sparse U factor.

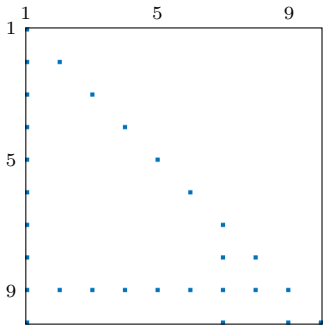

Figure 2.18: L factor.

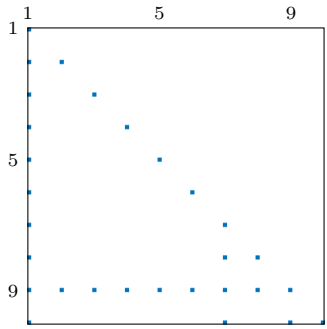

Figure 2.19: U factor.

during a factorization. The level of fill of element (i, j) is defined as

$$lev_{ij} = \begin{cases} 0 & \text{if } a_{ij} \neq 0 \text{ or } i = j, \\ \infty & \text{otherwise.} \end{cases}$$

A level of fill value of 1 indicates that the fill-in was generated by an original element. Whereas a level of 2 means that it was generated by a fill-in caused by an original element and so on. The rationale is that the level of fill should indicate the size of the element and one of the ideas is that fill-in resulting from other fill-in is less important.

To relate to graphs, if an element (i, j) has $lev_{ij} = p > 0$ then this means that an edge has been added to the original graph. The level of fill is also related to graphs by various results in the literature, e.g. Theorem 10.7, in [237]. After completing Gaussian elimination $lev_{ij} = p > 0$ if and only if there exist a fill path of length $p + 1$ between node i and j.

(c) The sparsity pattern for the matrix A and the LU factors (which is the same)

and the sparsity pattern for the and the LU factors computed with LU(0) are shown in Figure 2.20.

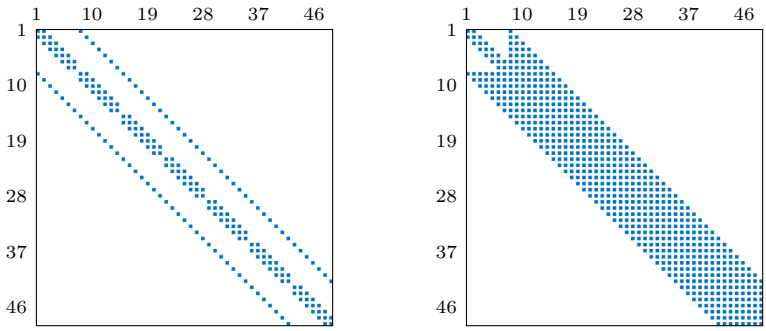

Figure 2.20: Sparsity pattern of A and $L+U$ (is the same) obtained with IL(0) on the left. Sparsity pattern of and $L+U$ computed with classic LU on the right.

(d) A naive function that computes the level of fill is given in the following MATALB code

```
function [ LEV ] = LOFnaive( A )
N = size(A,2);
%Initialize LEV matrix
LEV = inf*ones(N,N);
for i=1:N
  for j=1:N
    if (A(i,j) ~= 0 || i == j)
      LEV(i,j) = 0;
    end
 end
end
% Compute level of fill
for(i=2:N)
  for(k=1:i-1)
    for(j=k:N)
      LEV(i,j) = min(LEV(i,j), LEV(i,k) + LEV(k,j) + 1);
    end
  end
end
end
```

In Figure 2.21 is displayed the colormap of the output of the MATLAB function applied to the given matrix. The color white corresponds to a level of fill value of ∞. The color code is not optimal but it gives some perspective the levels and of which elements that would be removed if one would for instance use ILU(p).

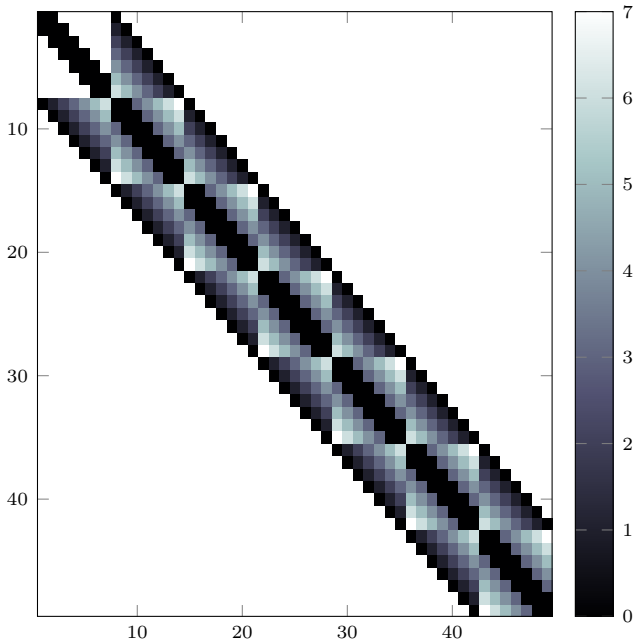

Figure 2.21: Colormap of the matrix obtained as output of LOFnaive.

Solution 2.13. Since M is a block diagonal matrix with the following structure

$$M = \begin{pmatrix} B & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{pmatrix}$$

the graph of M consists of three disjoint parts. An assumption for Algorithm 3.2 in [237] to work, is that the graph is connected. Otherwise one will end up in a situation where Snew=$\emptyset$, but seen $\neq n$, which is what will cause the infinite loop. The algorithm moves forward by considering neighbors of visited nodes, but in the case of a disconnected graph it is not possible to visit the whole graph by only going to neighbors.

On the other hand, this will only happen when the graph is disconnected, and that is equivalent with the matrix being block-diagonal. Thus one could solve the different (disconnected) linear systems separately (in this case 3), and hence one would be interested in doing the Cuthill-McKee ordering on each diagonal block separately and do the LU-factorization on each block separately.

Conclusion: From a software perspective the pseudo-code in Algorithm 3.2 in [237] is not very nice since it can be stuck in an infinite loop. However, the cases when this happens can be regarded as “lucky” cases.

Solution 2.14. The proof is by induction and for $k = 1$ we have by definition of the adjacency matrix that $h_{i,j}$ is non-zero if and only if there is a path of length 1 from node i to node j.

For the inductive step we assume the proposition is true for some value of $k \geq 1$. For ease of notation we let $B = H^{k+1}$ and $C = H^k$. The goal is now to prove that the claim of the proposition is true also for B. Specifically we have $B = CH$, and for element (i, j) we have

$$b_{ij} = \sum_{\ell=1}^{n} c_{i\ell} h_{\ell j}.$$

The elements of H and C are non-negative, and hence $b_{ij} \neq 0$ if and only if there exists (at least) one ℓ such that $c_{i\ell} \neq 0$ and $h_{\ell j} \neq 0$. However, from the induction hypothesis this is true if and only if there exists a path of length k from node i to node ℓ and a path of length 1 from node ℓ to node j. However, in such case there exists (at least) one path of length $k + 1$ from node i to node j.

Solution 2.15. We start by observing that any permutation matrix P can be written as the product of transpositions. A transposition is essentially a very simple permutation, interchanging only two indices. Moreover, a transposition interchanging the indices i and j and can be expressed as an identity matrix with rows i and j interchanged, i.e., the matrix T representing the transposition is all zero except $t_{kk} = 1$ for $k = 1, 2, \ldots, n$ and $k \neq i$, $k \neq j$, as well as $t_{i,j} = 1$, and $t_{j,i} = 1$.

From this the proof follows by induction. If we can show that It is true for a general transposition, that is, the graph of A and TAT^T are the same except for relabelling of the nodes, then it will hold true for a general permutation matrix P since $P = T_1 T_2 \ldots T_{m-1} T_m$, and thus $P^T = T_m^T T_{m-1}^T \ldots T_2^T T_1^T$ for some transpositions T_ℓ, $\ell = 1, \ldots, m$ and some m.

Hence we compare the graphs of A and $\tilde{A} = TAT^T$ for a general transposition T. The multiplication with T will interchange row i and j, and the multiplication with T^T will interchange column i and j. Then we can observe that $a_{k\ell} = \tilde{a}_{k\ell}$ if k and ℓ are not equal to i or j. Thus one can observe that any edge in the graph of A not involving node i or node j is the same in the graph of $\tilde{A}$. Moreover, $a_{i\ell} = \tilde{a}_{j\ell}$ and $a_{\ell i} = \tilde{a}_{\ell j}$ for ℓ not equal to i or j, as well as $a_{ii} = \tilde{a}_{jj}$, and similarly with i and j changed. Thus one can observe that if an edge goes from node i in the graph of A, there is a corresponding edge going from node j in the graph of $\tilde{A}$, again similarly with i and j changed. Thus the graph of A and $\tilde{A}$ are the same, but the labelling of node i and j is interchanged. This concludes the proof.

Solution 2.16. The Cuthill–McKee algorithm can be seen to create level sets: the ordered nodes are iteratively subdivided in their distance relative to the first node. Hence from construction, each level set, except the first and the last, is a graph separator. Let S_ℓ be the ℓ-th level set; then the two graphs formed by the

nodes of

$$V = S_1 \cup S_2 \cup \cdots \cup S_{\ell-1} \quad \text{and} \quad W = S_{\ell+1} \cup \cdots \cup S_m,$$

together with the respective edges, are disconnected components since S_ℓ is a graph separator and thus all the edges go through S_ℓ. Then if we, with a slight abuse of notation, call the adjacency matrix for the partial graph V for V, and correspondingly for W, we see that the adjacency matrix of the complete reordered graph have the structure

$$\begin{pmatrix} V & * & 0 \\ * & S_\ell & * \\ 0 & * & W \end{pmatrix}$$

where $*$ represents the edges between elements of S_ℓ and the two graphs V and W. This is a block-tridiagonal structure and since this is true for all level sets $\ell = 1, \ldots, m$, it is clear that the matrix reordered through Cuthill–McKee is block tridiagonal.

Solution 2.17.

(a) The script solves the Helmoltz equation on a non-regular grid, with wavenumber $\kappa = \sqrt{6}$ and homogeneous Dirichlet boundary condition. The function space consits of standard Lagrangian elements of order 2.

(b,c) The Cuthill–McKee ordering is implemented in the function `symrcm`. If Figure 2.22 and Figure 2.23 is shown the sparsity pattern of A before and after the Cuthill–McKee ordering. In Figure 2.24 and Figure 2.25 is shown the sparsity pattern of the factors L, U computed with the LU-method applied to A without Cuthill–McKee ordering. In Figure 2.26 and Figure 2.27 is shown the sparsity pattern of the factors L, U computed with the LU-method applied to A with Cuthill–McKee ordering.

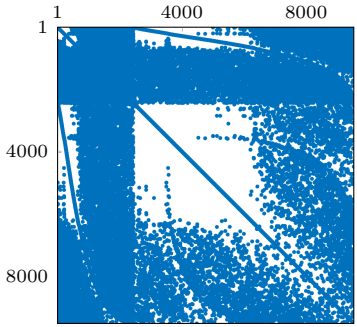

Figure 2.22: Before `symrcm`.

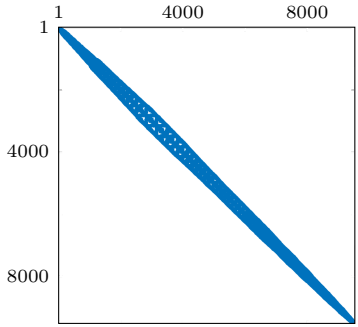

Figure 2.23: After `symrcm`.

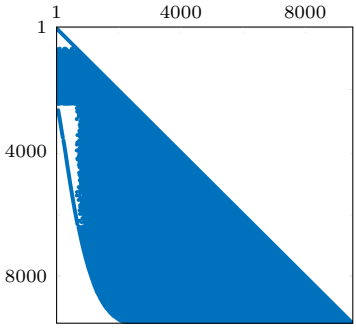

Figure 2.24: L factor.

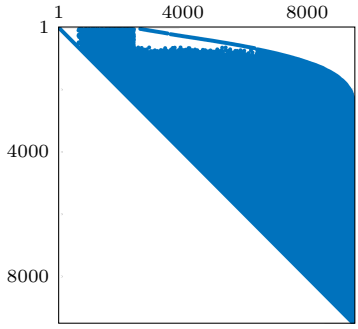

Figure 2.25: U factor.

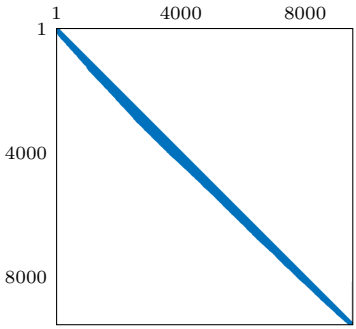

Figure 2.26: Sparse U factor.

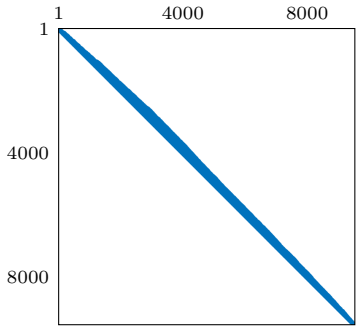

Figure 2.27: Sparse U factor.

(d) In Figure 2.28 is illustrated the convergence of GMRES with and without ILU preconditioner.

Solution 2.18. We start performing the Gauss elimination. We denote by $L^{(i)}$ and the $U^{(i)}$ the current matrix factorization. In particular in the beginning $L^{(0)} = I$ (identity) and $U^{(0)} = A$. In each iteration we have $A = L^{(i)}U^{(i)}$ and at the end of the Gaussian elimination $L^{(5)}$ will be lower triangular and $U^{(5)}$ upped triangular.

In the 0-th iteration we have

$$L^{(0)} = \begin{pmatrix} \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{pmatrix} \qquad U^{(0)} = \begin{pmatrix} \times & & \times & \times & & \times \\ & \times & & \times & & \\ \times & & \times & & \times & \\ \times & \times & & \times & & \times \\ & & \times & & \times & \times \\ \times & & & \times & \times & \times \end{pmatrix}$$

In the 1-th iteration we use the first row of $U^{(0)}$ to cancel the other nonzero

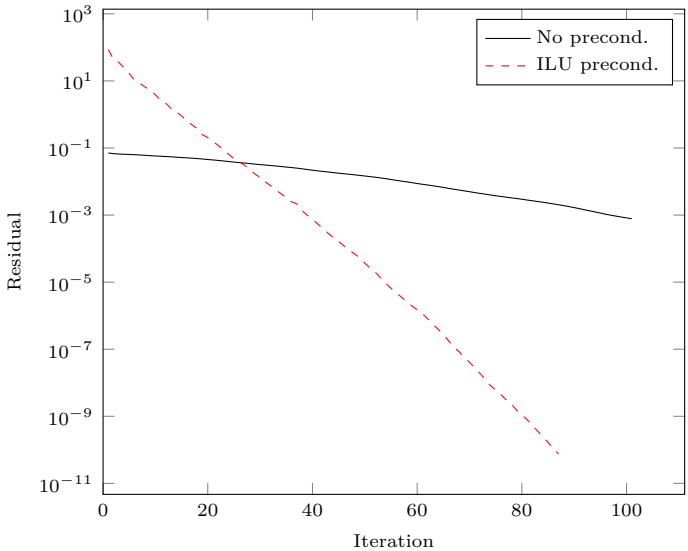

Figure 2.28: Convergence GMRES in Problem 2.17.

elements in the first column (standard Gaussian elimination) and we store in $L^{(1)}$ the coefficients for doing such operation and in $U^{(1)}$ the result of this cancellation. Obviously setting to zero and element in the first column may potentially generate fill-in in the other columns.

$$L^{(1)} = \begin{pmatrix} \times & & & & & \\ & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ & & & & \times & \\ \times & & & & & \times \end{pmatrix} \qquad U^{(1)} = \begin{pmatrix} \times & & \times & \times & & \times \\ & \times & & \times & & \\ & & \times & \times & \times & \times \\ & \times & \times & \times & & \times \\ & & \times & & \times & \times \\ & & \times & \times & \times & \times \end{pmatrix}$$

We continue with the same reasoning (with the second row we set to zero the elements in the second column)

$$L^{(2)} = \begin{pmatrix} \times & & & & & \\ & \times & & & & \\ \times & & \times & & & \\ \times & \times & & \times & & \\ & & & & \times & \\ \times & & & & & \times \end{pmatrix} \qquad U^{(2)} = \begin{pmatrix} \times & & \times & \times & & \times \\ & \times & & \times & & \\ & & \times & \times & \times & \times \\ & & \times & \times & & \times \\ & & \times & & \times & \times \\ & & \times & \times & \times & \times \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} \times & & & & & \\ & \times & & & & \\ \times & & \times & & & \\ \times & \times & \times & \times & & \\ & & \times & & \times & \\ \times & & \times & & & \times \end{pmatrix} \qquad U^{(3)} = \begin{pmatrix} \times & & \times & \times & & \times \\ & \times & & \times & & \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} \times & & & & & \\ & \times & & & & \\ \times & & \times & & & \\ \times & \times & \times & \times & & \\ & & \times & \times & \times & \\ \times & & \times & \times & & \times \end{pmatrix} \qquad U^{(4)} = \begin{pmatrix} \times & & \times & \times & & \times \\ & \times & & \times & & \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & \times & \times \end{pmatrix}$$

$$L^{(5)} = \begin{pmatrix} \times & & & & & \\ & \times & & & & \\ \times & & \times & & & \\ \times & \times & \times & \times & & \\ & & \times & \times & \times & \\ \times & & \times & \times & \times & \times \end{pmatrix} \qquad U^{(5)} = \begin{pmatrix} \times & & \times & \times & & \times \\ & \times & & \times & & \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{pmatrix}$$

Solution 2.19. The Incomplete LU factorization (ILU) is an approximation to the LU factorization, which is obtained using the Gaussian Elimination algorithm (GE) in a way that, instead of obtaining exactly L and U as in GE, one derives L and U so that they satisfy a certain sparsity pattern $\mathcal{P}$, so as to decreasing the storage and computational cost of using L and U.

Let $\mathcal{P}$ be a zero-pattern

$$\mathcal{P} \subset \{(i,j) \ \mid \ i \neq j, \ \ 1 \leq i,j \leq n\}$$

and let L_1 be the matrix which applies the first step of GE, $A_1 = L_1 A$. Then, we only consider the matrix obtained by keeping the elements of A_1 with zero-pattern P, which means that the rest of the elements is subtracted and kept in the matrix R_1, so we get $\tilde{A}_1 = A_1 + R_1$. We repeat the process: by apply one step of GE to this matrix and then only keep the zero-pattern we get $A_2 = L_2 \tilde{A}_1$ and $\tilde{A}_2 = A_2 + R_2$. Just to make a point, let's proceed with another step: $A_3 = L_3 \tilde{A}_2$ and $\tilde{A}_3 = A_3 + R_3$, and then

$$\begin{aligned} \tilde{A}_3 &= L_3 L_2 L_1 A + L_3 L_2 R_1 + L_3 R_2 + R_3 \\ &= L_3 L_2 L_1 A + L_3 L_2 L_1 R_1 + L_3 L_2 L_1 R_2 + L_3 L_2 L_1 R_3 \end{aligned}$$

is obtained by noticing that the matrix L_1 will correspond to the identity matrix in the first row, thereby not affecting the first columns of the matrix it's multiplied

to. In general we have

$$L_k = I - \frac{1}{a_{kk}^{(k)}} \begin{pmatrix} \mathbf{0}_k \\ A(k+1:n,k) \end{pmatrix} e_k^T$$

Consequently, $L_{n-1} \cdots L_{k+1} R_k = L_{n-1} \cdots L_1 R_k$. We define $L := (L_3 L_2 L_1)^{-1}$ and $R := R_1 + R_2 + R_3$, and we get $\tilde{A}_3 = L^{-1}A + L^{-1}R$. In general, after $n-1$ steps, we get, analogously as before,

$$\begin{aligned} U = \tilde{A}_{n-1} &= L_{n-1} L_{n-2} \cdots L_1 A + L_{n-1} L_{n-2} \cdots L_1 (R_{n-1} + \cdots + R_1) \\ &= L^{-1}A + L^{-1}R \end{aligned}$$

Then $LU = A + R$ which gives $A = LU - R$.

In conclusion R represents the sum of all the elements subtracted to the matrix after each GE step so that the zero-pattern P is satisfied.

ILU(0) is the ILU performed where the zero-pattern is the original sparsity pattern of A. Hence, the matrices L and U will both have at most the same sparsity pattern as A, but be lower and upper triangular respectively. OBS: this does not guarantee that the pattern of LU will be the same as the pattern of A! Rwill consequently be the matrix which will subtract the elements of LU which do not belong to the sparsity pattern of A.

ILU(1) is as if the ILU was performed with $\mathcal{P}$ zero-pattern of the LU from the ILU(0) run, which we call now LU. So now the new matrices L and U will have at most the sparsity pattern of LU, and R, again, will be the matrix which will subtract the elements of LU which do not belong to the sparsity pattern of A.

Solution 2.20. Where as ILU simply works by discarding the corresponding elements in the zero pattern, where as the modified ILU tries to compensate for the discarded elements. More precisely, what is called the MILU is, after completing the k-iteration, modifying the diagonal element with the sum of the dropped elements.

The arrangements of the algorithm is based on the IKJ-version of Gaussian elimination, where the outer i-loop works on each row, the middle k-loop works on the lower triangular part given an i (thus essentially doing the elimination from "left to right"), and then for each i and k the j-loop works on modifying the elements $k+1, k+2, \ldots, n$ in accordance with the elimination.

The result of the MILU-compensation operation is that the sum of each row is kept constant. We still have the factorization $A = LU - R$, for a nonzero R, but additionally we have that $Ae = LUe$, where e is a vector of ones. Note that this does not affect the zero-pattern used, but only the actual values filled in. However, the zero-pattern for R becomes different. One reason for doing this is that the new operator $A = LU$, when stemming from a PDE, will behave correctly for constant functions.

As a numerical comparison for the given matrix (with $n = 3$) the norm of $Ae - LUe$ for ILU(0) is 0.8474, and for MILU is $8.9509e-16$.

Solutions to Chapter 3

Solution 3.1.

(a) The equation (3.21) can be solved analytically. More precisely, by computing the roots of the characteristic equation $\lambda^2 + \kappa^2 = 0$, we get that the solutions can be written as $u(x) = Ae^{i\kappa x} + Be^{-i\kappa x}$, and for κ real it can be further simplified as $u(x) = A\sin(\kappa x) + B\cos(\kappa x)$. The constants A and B are computed by imposing the boundary conditions, which results in the following linear system

$$\begin{aligned} A + B &= u_0 \\ A\sin(\kappa) + B\cos(\kappa) &= u_1 \end{aligned}$$

(b) We fix the boundary conditions as $u_0 = 1$ and $u_1 = 0$. The solution to (3.21) is plotted in Figure 3.3 for $\kappa = 10$ and Figure 3.4 for $\kappa = 100$. Clearly the oscillations increase proportionally to κ.

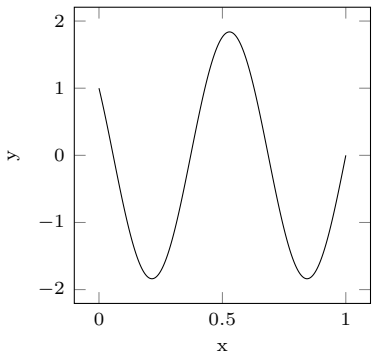

Figure 3.3: $\kappa = 10$.

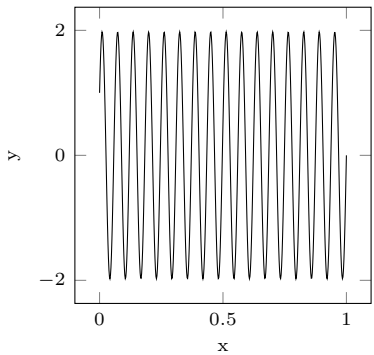

Figure 3.4: $\kappa = 100$.

The following MATLAB script illustrates how to compute the numerical solution with FD.

```
n=100;  % number of discretization points
kk=30;  % wavenumber

xx=linspace(0,1,n); h=xx(2)-xx(1);
e=ones(n,1); I=speye(n);
D = spdiags([e -2*e e], -1:1, n, n);
A = D./(h^2)+kk^2*I;
A(1,:)=0;        A(1,1)=1;
A(end,:)=0;      A(end,end)=1;
b=zeros(n,1);    b(1)=1; b(end)=0;

% numerical solution
```

```
u=A\b;
plot(xx,u)

% analytic solution
cc=[0 1; sin(kk) cos(kk)]\[1;0];
uu=@(x) cc(1)*sin(kk*x)+cc(2)*cos(kk*x);
hold on
plot(xx,uu(xx))
```

If n is not large enough, namely the exact solution oscillates between two discretization points, then the numerical solution does not even estimate the exact solution[2]. This concept is illustrated in Figure 3.5 and Figure 3.6, where for $\kappa = 40$, and for a reasonable number of discretization points $n = 50$, the numerical solution does not even estimate the exact solution.

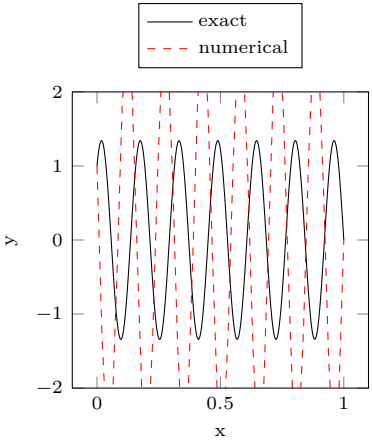

Figure 3.5: $\kappa = 40$ and $n = 50$.

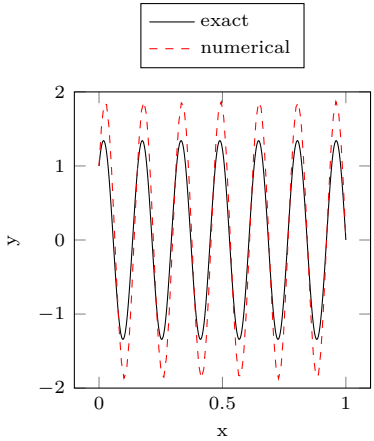

Figure 3.6: $\kappa = 40$ and $n = 100$.

(c) If κ is complex the solution is damping. The larger is the imaginary part of κ the faster is the damping. See Figures 3.7, 3.8, 3.9, 3.10.

Solution 3.2.

(a) See Problem 3.1.

(b) By Problem 3.1 we have that the general solution is

$$u(x) = Ae^{i\kappa(1+\varepsilon i)x} + Be^{-i\kappa(1+\varepsilon i)x}.$$

Where A and B are determined from the boundary conditions. The homogeneous Dirichlet boundary condition at infinity $\lim_{x\to\infty} u(x) = 0$ implies

[2]In signal processing this phenomena is called *aliasing*

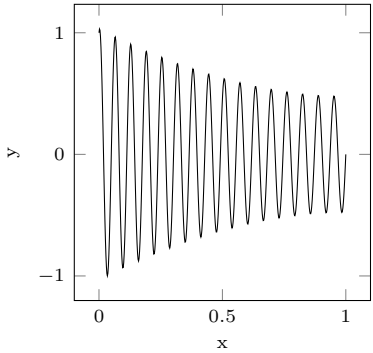

Figure 3.7: Real part of sol. to (3.21) for $\kappa = 100 + i$

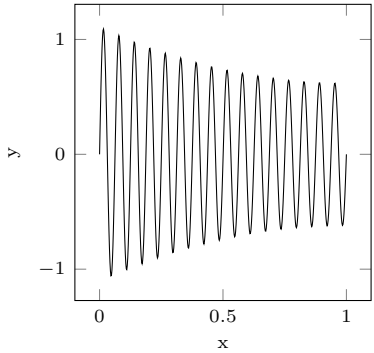

Figure 3.8: Imaginary part of sol. to (3.21) for $\kappa = 100 + i$

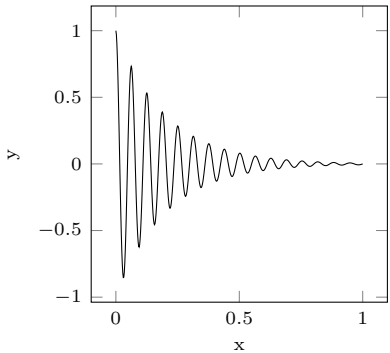

Figure 3.9: Real part of sol. to (3.21) for $\kappa = 100 + 5i$

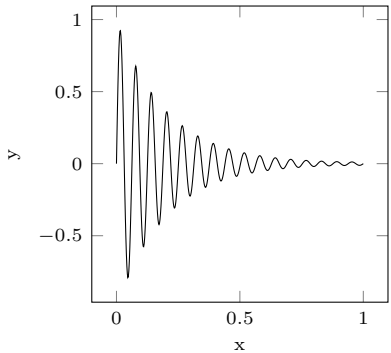

Figure 3.10: Imaginary part of sol. to (3.21) for $\kappa = 100 + 5i$

$A = 0$, and therefore solution can be written as

$$u(x) = B \underbrace{e^{i\kappa x}}_{f(x)} \underbrace{e^{-\varepsilon\kappa x}}_{g_\varepsilon(x)},$$

where, by using the Euler's formula, we can clearly see that $f(x)$ is periodic $f(x) = e^{i\kappa x} = \cos(\kappa x) + i\sin(\kappa x)$.

(c) The function $g_\varepsilon(x) = e^{-\varepsilon\kappa x}$ goes to zero exponentially for $x \to \infty$ and, the larger is ε the faster is the decay.

Solution 3.3.

(a) In Figure 3.11 is illustrated the convergence of GMRES for different values of κ^2. The number of steps to reach a desired tolerance is not very sensitive

with respect to the value of κ^2, i.e. this preconditioner performs well independently on the value of κ^2.

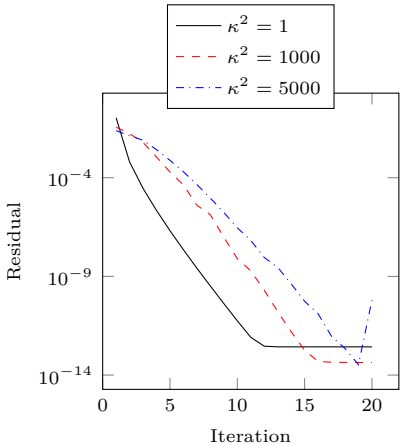

Figure 3.11: convergence of GMRES for different values of κ^2.

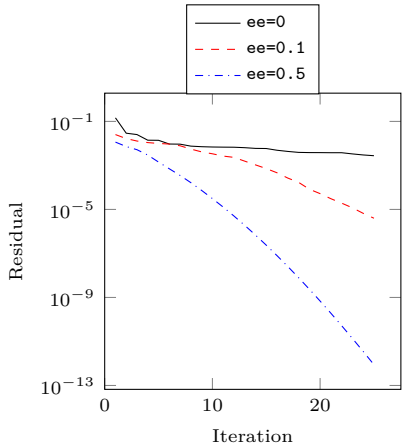

Figure 3.12: convergence of GMRES for different values of `ee`.

(b) The larger is κ^2 the slower is the convergence. Theoretically, larger κ^2 means higher frequency of the oscillations, even for the shifted problem. Oscillations make the problem harder to solve for the two grids method, because the accuracy of the approximation in the solution on the coarser grid suffers from the smaller number of points used to capture the oscillatory behavior.

(c) We fix $\kappa^2 = 100000$ so that the solutions has oscillations with high frequencies. In Figure 3.12 we report the convergence plots of GMRES for different values of `ee`. As expected the preconditioner works better if the solution is damping fast, i.e. for larger values of `ee`.

Solution 3.4. The following is the MATLAB script which can be used to reproduce the results presented in the following points.

```
ee=0.5;              % value of epsilon
kk=500*(1+ee*1i); % value of k^2
n=50; xx=linspace(0,1,n); h=xx(2)-xx(1);
e=ones(n,1);
D=spdiags([e -2*e e], -1:1, n, n)/h^2;
D(1,:)=0; D(1,1)=1; D(end,:)=0; D(end,end)=1;
I=speye(n); A=kron(D,I)+kron(I,D)+kk*kron(I,I);
b=ones(n,n); b(:,1)=0; b(1,:)=0; b(:,end)=0; b(end,:)=0; b=b(:);
u=A\b; u=reshape(u,n,n);
s=surf(xx,xx,abs(u));
```

(a) The larger is κ^2 the higher is the frequency of the oscillations. See Figure 3.13 for $\kappa^2 = 10$ and Figure 3.14 for $\kappa^2 = 100$.

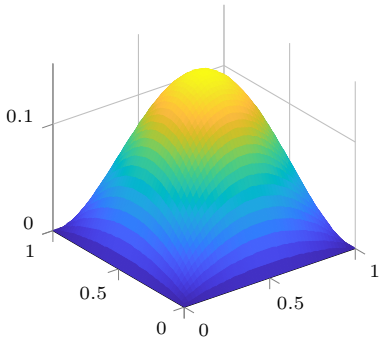

Figure 3.13: $\kappa^2 = 10$

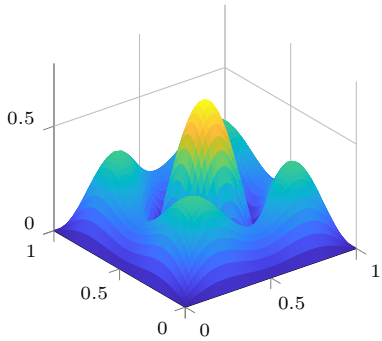

Figure 3.14: $\kappa^2 = 100$

(b) We fix $\kappa^2 = 500$. If $\varepsilon > 0$ the solution is damping in the middle of the domain. More precisely the larger is ε the faster is the damping. See Figure 3.15 for $\varepsilon = 0$ and Figure 3.16 for $\varepsilon = 0.5$.

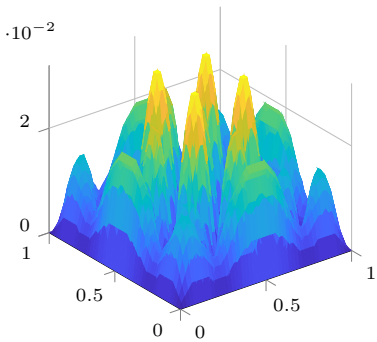

Figure 3.15: $\kappa^2 = 500, \varepsilon = 0$

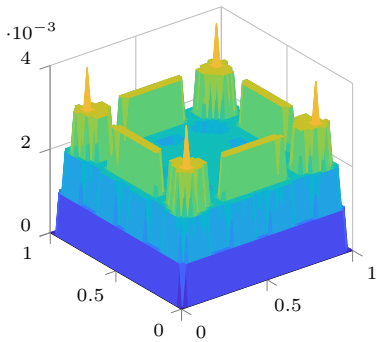

Figure 3.16: $\kappa^2 = 500, \varepsilon = 0.5$

(c) As for the mono-dimensional case, the oscillations have higher frequency when κ grows and $\varepsilon > 0$ introduces a damping. The larger is ε the faster is the damping.

Solution 3.5. The output matrix produced by the script is available online.

(a) The script produces a FEM discretization of the Helmholtz equation

$$\begin{aligned} \Delta u(x,y) + \kappa^2 u(x,y) &= f(x,y) && (x,y) \in [0,1]^2 \\ u(x,y) &= 1 && (x,y) \in \partial[0,1]^2 \end{aligned}$$

where $f(x,y) = -6$ and $\kappa = 20$.

(b) The following is the required MATLAB script

```
load Helmholtz.mat
u=A\b.'; n=sqrt(length(u)); u=reshape(u,n,n);
xx=linspace(0,1,n); surf(xx,xx,u)
```

and it produces the Figure 3.17.

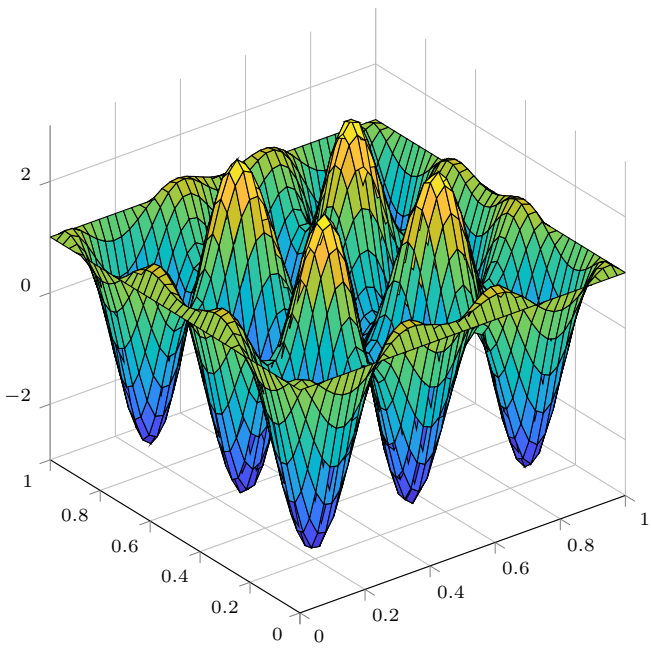

Figure 3.17: Solution Helmholtz equation from FEniCS.

Solution 3.6.

(a) The easiest weak formulation (as there can be many) is the generalization of the weak formulation used for Poisson equation $\Delta u = -f$ with zero Dirichlet boundary condition: $a(u,v) = \int_\Omega \nabla u \cdot \nabla v$ and $F(v) = \int_\Omega f v$. Assuming zero Dirichlet boundary conditions, from $\Delta u + k^2 u = -f$ by multiplying by a test function v and integrating over the domain Ω, after

applying Green's formulas to integrate by parts, we get the bilinear form $a_D(u,v) = \int_\Omega \left(\nabla u \cdot \nabla v - k^2 uv\right)$ and the functional $F(v) = \int_\Omega fv$.

(b) A bilinear form $a : H \times H \to \mathbb{R}$ is said coercive if it exists a constant $C' > 0$ such that $a(v,v) \geq C'\|v\|_H^2$ for all $v \in H$. This, crudely said, means that the the functional which describes the problem grows more than quadratically when v goes to infinity (in the sense of the H-norm), and this assures of the existence and uniqueness of the solution to the problem, which is the minimizer of said functional. Of course, more hypotheses are needed on a and F. The bilinear form $a_D(u,v)$ is not coercive, and actually is not even sign-definite. To show this, let u_j be the j-th eigenfunction to the negative Laplacian $-\Delta u_j = \lambda_j u_j$, if $k^2 = \lambda_j$, then $a_D(u_j, u_j) = 0$.

(c) A Galerkin method approximates the solution u to a continuous problem with a solution v^* belonging to a certain finite-dimensional space V_h. Let $v \in V_h$ be the optimal solution in this space, hence $\|u - v\|_H = \inf_{v_h \in V_h} \|u - v_h\|_H$. We say that v^* is a quasi-optimal solution if it exists a constant C such that $\|u - v^*\|_H \leq C\|u - v\|_H$. This means that the solution is optimal up to a scaling constant. Of course, if the constant is very big, the estimate is not very illuminating. When this kinds of estimates are true, then the notation $a \lesssim b \iff a \leq Cb$ is used.

(d) When the complex shift is introduced in the equation, the corresponding bilinear form becomes

$$a_\varepsilon(u,v) = \int_\Omega \left(\nabla u \cdot \nabla v - k^2(1+\varepsilon i)uv\right)$$

which is now coercive. By using [134, Lemma 3.1] we have that the following norm is defined on $H^1(\Omega)$

$$\|v\|_{1,k,\Omega}^2 := \|\nabla v\|_{L^2(\Omega)}^2 + k^2\|v\|_{L^2(\Omega)}^2$$

which is equivalent to the usual $H^1(\Omega)$ norm when $k > k_0$ for some $k_0 > 0$. Then, if $0 < \varepsilon \lesssim k^2$, it exists a constant α independent of k, η, and ε, such that

$$|a_\varepsilon(v,v)| \geq \alpha \frac{\varepsilon}{k^2}\|v\|_{1,k,\Omega}^2 \qquad \text{for all } k > 0 \text{ and } v \in H^1(\Omega).$$

The bilinear form is also continuous: in the same hypotheses, given $k_0 > 0$, there exists C_c independent of k, η, and ε such that $|a_\varepsilon(v,v)| \leq C_c\|u\|_{1,k,\Omega}\|v\|_{1,k,\Omega}$ for all $k \geq k_0$ and $u, v \in H^1(\Omega)$. When the domain Ω is smooth or a convex polygon, it is possible to show (by using the coercivity) that the Galerkin approximation to the shifted Helmholtz problem is quasi-optimal:

$$\|u - u_N\|_{1,k,\Omega} \leq \frac{2C_c}{\alpha} \inf_{v_N \in V_N} \|u - v_N\|_{1,k,\Omega}$$

but there are more general theorems which prove quasi-optimality when the domain is less smooth, when certain other conditions are met.

Solution 3.7. Let's consider the FD discretization with n points. By using the equations (3.9) and (3.10), we should prove that $\lambda_j/\lambda_j^\varepsilon$ lie in the circle centered in $1/2$ with radius $1/2$. This can be done with standard techniques of complex calculus. One can directly verify this fact by plotting $\lambda_j/\lambda_j^\varepsilon$ for $j = 1, \ldots, n$ without computing the eigenvalues of the discretized problem, but using (3.9) and (3.10).

Solution 3.8. The following MATLAB script does the required comparison.

```
load('Helmholtz.mat'); b = b';
nn = length(b); n = sqrt(nn);
[xx_no,~,~,~,relres_no] = gmres(A,b,[],1e-9,200);
semilogy(relres_no);
e = ones(n,1); DD = spdiags([e -2*e e], -1:1, n, n);
prec = @(x) reshape(lyap(DD, -reshape(x,n,n)),nn,1);
[xx_prec,~,~,~,relres_prec] = gmres(A,b,[],1e-9,200,prec);
hold on; semilogy(relres_prec)
xlabel('Iteration'); ylabel('Residual')
```

The larger is κ the worse is the preconditioner. See Figure 3.18 for $\kappa = 20$ and Figure 3.19 for $\kappa = 30$.

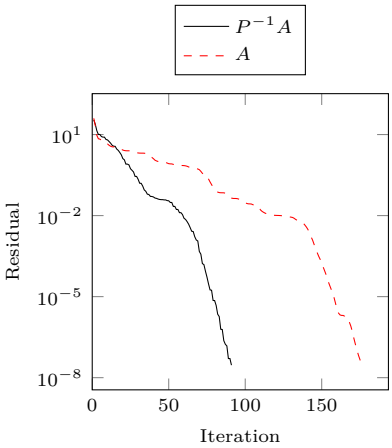

Figure 3.18: $\kappa = 20$

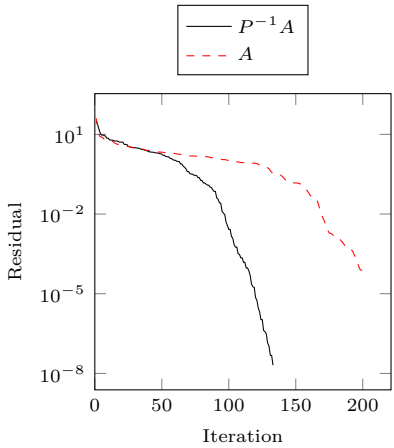

Figure 3.19: $\kappa = 30$

Solution 3.9. The following is the required Python script

```
from __future__ import print_function
from FEniCS import *
import matplotlib.pyplot as plt
```

```
# Create mesh and define function space
N=10; p=3;
mesh = IntervalMesh(N, 0, 1)
V = FunctionSpace(mesh, 'CG',p)

# Define boundary condition
u_D = Expression('1', degree=1)

def boundary(x, on_boundary):
    return on_boundary

bc = DirichletBC(V, u_D, boundary)

# Define variational problem
k = 30;
u = TrialFunction(V)
v = TestFunction(V)
f = Constant(-6.0)
a = -dot(grad(u), grad(v))*dx+k**2*dot(u,v)*dx
L = f*v*dx

# Compute solution
u = Function(V)
solve(a == L, u, bc)

# Plot solution in a finer mesh
# OBS: it is needed to interpolate it
mesh2 = IntervalMesh(100, 0, 1)
V2 = FunctionSpace(mesh2, 'CG',1)
u=interpolate(u,V2)
plot(u); plt.show()
```

Even if the solution oscillates between the nodes, if a large enough p is used, the correct solution can still be computed. In Figure 3.20 we have the solution for $p = 1$ and in Figure 3.21 the solution for $p = 8$. In particular, the solution obtained for $p = 8$ is a good approximation to the exact solution. In conclusion, a good approximation to the solution can be obtained by increasing n (number of nodes), which corresponds to solving a larger linear system, or by increasing p (degree basis functions), which corresponds to solving a denser linear system.

Solution 3.10. Let us denote $\kappa = ig$ with $g \in \mathbb{R}$, the solution is $u(x) = c_1 \exp(gx) + c_2 \exp(-gx)$ where c_1 and c_2 depend on the boundary conditions. Clearly, the function $u(x)$ does not oscillate. Methods based on exploiting the regularity of the solution, e.g. multigrid, are very effective. The Jacobi method does not converge since $\rho(D^{-1}A) > 1$ where A is the discretization matrix and D is its diagonal.

Solution 3.11.

(a) No. There are PDEs that are not symmetric[3] that lead to a symmetric

[3]For symmetric PDE we mean that the PDE does not change if we permute the independent variables, i.e. we replace x with y and y with x.

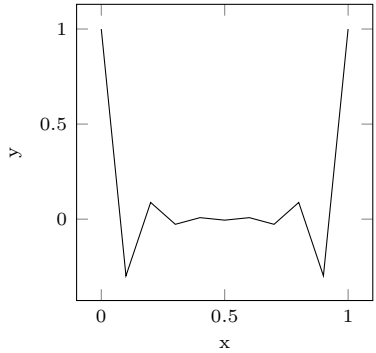

Figure 3.20: $p = 1$

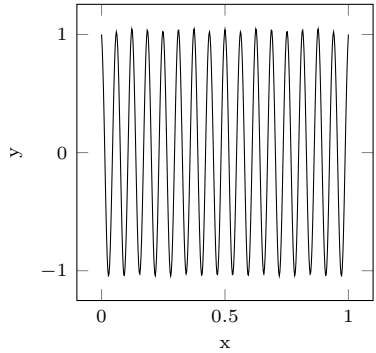

Figure 3.21: $p = 8$

systems and PDEs that are symmetric lead to nonsymmetric systems. For example the following PDE is nonsymmetric but the discretized linear systems is symmetric.

$$\begin{cases} \dfrac{\partial^2}{\partial x^2} u(x,y) + u(x,y) = f(x,y) & (x,y) \in [0,1]^2 \\ u(x,y) = 0 & (x,y) \in \partial[0,1]^2 \end{cases}$$

Instead the following PDE is nonsymmetric but the discretized linear systems is symmetric.

$$\begin{cases} \Delta u(x,y) + \dfrac{\partial}{\partial x} u(x,y) + \dfrac{\partial}{\partial y} u(x,y) + u(x,y) = f(x,y) & (x,y) \in [0,1]^2 \\ u(x,y) = 0 & (x,y) \in \partial[0,1]^2 \end{cases}$$

(b) The LDL^T factorization exists only for symmetric matrices. See also Problem 3.12.

Solution 3.12. Let us consider the decomposition $A = LDM^T$. Since L and M are lower triangular with ones in the diagonal, they are invertible, therefore $M^{-1}AM^{-T} = M^{-1}LD$. Since inverse of a lower triangular matrix is lower triangular and product of two lower triangular matrices is lower triangular, $M^{-1}LD$ is lower triangular. Since $M^{-1}AM^{-T}$ is symmetric (due to symmetry of A), $M^{-1}1LD$ is symmetric as well. Symmetry and lower triangularity together imply that $M^{-1}LD$ is diagonal, which means that $M^{-1}L$ is diagonal since D is nonsingular. Since both M and L are unit lower-triangular, $M^{-1}L = I$ implies that $M = L$, which completes the proof.

Solution 3.13. The solution is also described in Algorithm 2.1 in [122]. The following code is the completition of the script for computing the sweeping factorization

```
nn = size(A,1); n = sqrt(nn);
T = sparse(nn,nn);
idx_transf = @(m) (1:n)+(m-1)*n; %Helper to modify indices
m1 = idx_transf(1);
S = A(m1,m1); Tm = inv(S);
for m = 2:n
 m_idx = idx_transf(m);
 mm1_idx = idx_transf(m-1);
 S = A(m_idx, m_idx) - A(m_idx, mm1_idx) * Tm * A(mm1_idx, m_idx);
 Tm = inv(S);  T(m_idx, m_idx) = Tm;
end
```

Recall that, in our application (discretization of the Helmholtz equation) the matrix A is a block tridiagonal matrix of size n^2 where each block has size n. The complexity of the sweeping factorization, for a matrix with this specific structure, is dominated by the inversion $T_m = S_m^{-1}$, which has complexity $\mathcal{O}(n^3)$. Since this operation is performed n times, the total complexity is $\mathcal{O}(n^4)$.

Solution 3.14. The solution is also described in Algorithm 2.2 in [122]. The following is the completion of the provided MATLAB code

```
nn = size(A,1); n = sqrt(nn);
idx_transf = @(m) (1:n)+(m-1)*n; %Helper to modify indices
u = zeros(n^2,1);
for m = 1:n
  m_idx = idx_transf(m);
  u(m_idx) = f(m_idx);
end
for m = 1:n-1
  m_idx = idx_transf(m);
  mp1_idx = idx_transf(m+1);
  Tm = T(m_idx, m_idx);
  Am1m = A(mp1_idx, m_idx);
  u(mp1_idx) = u(mp1_idx) - Am1m * (Tm*u(m_idx));
end
for m = 1:n
  m_idx = idx_transf(m);
  Tm = T(m_idx, m_idx);
  u(m_idx) = Tm*u(m_idx);
end
for m = (n-1):-1:1
  m_idx = idx_transf(m);
  mp1_idx = idx_transf(m+1);
  Tm = T(m_idx, m_idx);
  Amm1 = A(m_idx, mp1_idx);
  u(m_idx) = u(m_idx) - Tm*(Amm1*u(mp1_idx));
end
```

Recall that, in our application (discretization of the Helmholtz equation) the matrix A is a block tridiagonal matrix of size n^2 where each block has size n. The complexity come from the matrix-vector multiplications which are $\mathcal{O}(n^2)$ and is performed $\mathcal{O}(n)$ times. Therefore, the total complexity is $\mathcal{O}(n^3)$.

Solution 3.15. Sweeping factorization is a block LDL^T factorization of A that eliminates the unknowns layer by layer. For example consider the system $Au = b$ where

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & & \\ A_{2,1} & A_{2,2} & \ddots & \\ & \ddots & \ddots & A_{n-1,n} \\ & & A_{n,n-1} & A_{n,n} \end{pmatrix}$$

where $A_{m,m}$ are tridiagonal and $A_{m,m-1} = A_{m-1,m}^T$. The block LDL^T factorization is then performed block by block. Starting from the rows corresponding to the first block layer

$$A = L_1 \begin{pmatrix} S_1 & & & & \\ & S_2 & A_{2,3} & & \\ & A_{3,2} & A_{3,3} & \ddots & \\ & & \ddots & \ddots & A_{n-1,n} \\ & & & A_{n,n-1} & A_{n,n} \end{pmatrix} L_1^T$$

where $S_1 = A_{1,1}, S_2 = A_{2,2} - A_{2,1}S_1^{-1}A_{1,2}$ and

$$L_1 = \begin{pmatrix} I & 0 & & & \\ A_{2,1}S_1^{-1} & I & 0 & & \\ & 0 & I & \ddots & \\ & & \ddots & \ddots & 0 \\ & & & 0 & I \end{pmatrix}.$$

This process is then repeated such that

$$A = L_1 \dots L_{n-1} \begin{pmatrix} S_1 & & \\ & \ddots & \\ & & S_n \end{pmatrix} L_{n-1}^T \dots L_1^T$$

where $S_m = A_{m,m} - A_{m,m-1}S_{m-1}^{-1}A_{m-1,,m}$ for $m = 2, \dots, n-1$ and L_m is a block bidiagonal matrix whose diagonal blocks are I and the off-diagonal blocks are 0 except the block $(m+1, m)$ that is $A_{m+1,m}S_m^{-1}$.

If a different elimination ordering is used the rank of the off-diagonal blocks of S_m become much higher and therefore the technique does not result in an efficient algorithm.

Solution 3.16. Given a matrix M with (ordered) singular values $\sigma_1, \dots, \sigma_n$, the numerical rank of a M is k if $\sigma_1/\sigma_k < \varepsilon$, where ε is the wanted tolerance, which we

fixed to 10^{-16}. We used the code in Problem 3.13 and generate the matrices such that T_m has size $n^2 = 256$ and it is composed by block matrices of size $n = 16$. In Figure 3.22 we illustrate the numerical rank of each block of T_2 (similar results hold for the other matrices T_m). As we can see the rank decreases moving away from the block diagonal. This effect is amplified for larger matrices.

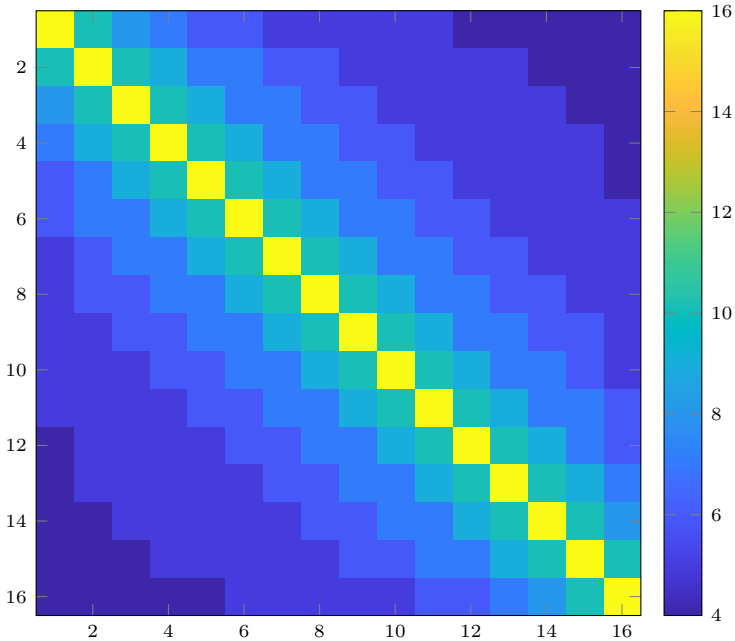

Figure 3.22: The color represent to rank of each block. Yellow is full-rank (16) and blue is low-rank (4).

Solution 3.17.

(a) With a direct computation, following the Schur complement scheme, we get

$$
\begin{aligned}
M &= \begin{pmatrix} A & uv^T \\ uv^T & B \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ uv^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & B - uv^T A^{-1} uv^T \end{pmatrix} \begin{pmatrix} I & 0 \\ uv^T A^{-1} & I \end{pmatrix}^T
\end{aligned}
$$

which has rank-1 structure, and can be reached with $\mathcal{O}(n)$ operations.

(b) Analogously to the previous point

$$
\begin{aligned}
M &= \begin{pmatrix} A & UV^T \\ UV^T & B \end{pmatrix} \\
&= \begin{pmatrix} I & 0 \\ UV^TA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & B - UV^TA^{-1}UV^T \end{pmatrix} \begin{pmatrix} I & 0 \\ UV^TA^{-1} & I \end{pmatrix}^T
\end{aligned}
$$

Observe that $Y := A^{-1}V$ can be calculated in $\mathcal{O}(nk)$ by solving k linear systems with the matrix A. Same for $W := Y^TU$, which is also calculated in $\mathcal{O}(nk)$. Then, the LDL^T factorization is given by

$$
M = \begin{pmatrix} I & 0 \\ UY^T & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & B - UWV^T \end{pmatrix} \begin{pmatrix} I & 0 \\ UY^T & I \end{pmatrix}^T
$$

(c) Let us consider the truncated SVD factorization $\tilde{E} = \tilde{U}\tilde{\Sigma}\tilde{V}^T$ such that $\|E - \tilde{E}\| < 10^{-16}$. The approximated LDL^T factorization is given by

$$
\tilde{M} = \tilde{L}\tilde{D}\tilde{L}^T = N \begin{pmatrix} A & 0 \\ 0 & B - \tilde{U}\tilde{\Sigma}\tilde{V}^TA^{-1}\tilde{U}\tilde{\Sigma}\tilde{V}^T \end{pmatrix} N^T
$$

where

$$
N = \begin{pmatrix} I & 0 \\ \tilde{U}\tilde{\Sigma}\tilde{V}^TA^{-1} & I \end{pmatrix}
$$

which can be computed in $\mathcal{O}(nk)$ where k is the rank of the matrices in the truncated SVD factorization.

Solution 3.18. A matrix $A \in \mathbb{R}^{n\times n}$ has an hierarchical structure if we can find non-intersecting index sets $(J_k)_{k=1}^p$ and $(I_r)_{r=1}^q$ such that $\cup_{k=1}^p J_k = \cup_{r=1}^q I_r = (1, 2, \ldots, n)$ and the submatrices A_{J_k,I_r} have low rank. The idea behind the hierarchical matrices is to extend the benefits of the low rank theory. We know that if a matrix $A \in \mathbb{R}^{n\times n}$ has rank k it can be stored/represented by using $\mathcal{O}(kn)$ data. Clearly if a matrix $A \in \mathbb{R}^{n\times n}$ is not low rank, but can be represented (via submatrices) by using low rank matrices, then we have benefits in terms of memory usage in its representation. Standard matrix-matrix and matrix-vector operations (sum, multiplication, products, linear solves, etc) can efficiently be performed by exploiting this structure. As an example, low rank matrices are hierarchical. The following matrix

$$
A = \begin{pmatrix} 1 & 3 & 2 & 6 \\ -1 & 0 & -2 & 0 \\ 2 & 3 & 4 & 6 \\ -2 & 1 & -4 & 2 \end{pmatrix}
$$

is full rank, but has an hierarchical structure. Select $I_1 = (1,3)$,$I_2 = (2,4)$,$J_1 = (1,3)$,$J_2 = (2,4)$ and verify that the submatrices A_{J_k,I_r} have rank one. Another class of hierarchical matrices are tridiagonal matrices. Quasi-separable matrices are also hierarchical matrices. In certain applications from PDEs we get hierarchical matrices.

Solution 3.19. We denote $S := S_5$. This matrix has a very specific structure, the absolute values of its elements decay exponentially from the diagonal, i.e., there are two constants C and α such that $(T_m)_{i,j} \approx Ce^{-\alpha|i-j|}$. See Figure 3.23. Such matrices appear in many different fields, see e.g., [102]. Because of this, we expect the diagonal preconditioning to be effective. See Figure 3.24.

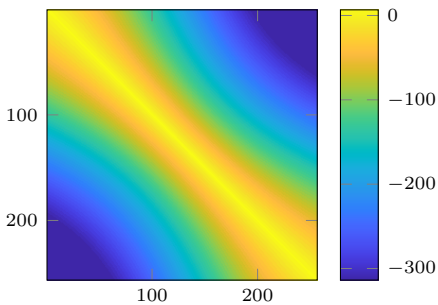

Figure 3.23: `imagesc(lohg10(abs(S)))`.

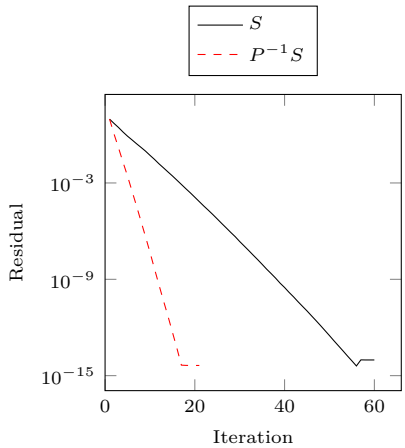

Figure 3.24: Convergence of GMRES w/o diagonal preconditioner.

Solution 3.20. Consider any function that can be decomposed as

$$G(x,y) = H(x,y) + \sum_{i=1}^{k} w_i(x) z_i(y)$$

such that $\{H \neq 0\}$ has small area with respect to $\{G \neq 0\}$.

Solutions to Chapter 4

Solution 4.1.

(a) The following lines contain the missing code