1. Exercise 10.1:

   a) Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix. Use (8.1b) to conclude that the approximation x_n obtained in the n-th step (10.4) according to the MINRES criterion (10.3b) is the exact solution x_* of $Ax = b$.

   b) Assume additionally that for some $m \le n$ there holds $\mathcal{K}_m = \mathcal{K}_n$. Show that then already

   $$x_m = x_{m+1} = \cdots = x_*$$

   Hint: Show that $\mathcal{K}_k = \mathcal{K}_m = \mathcal{K}_n$ also for all $k > m$.

2. Exercise 10.2:

   Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix. Assume $r_0 \neq 0$ and let $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ be defined by the Arnoldi algorithm,

   $$\bar{H}_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2m} \\ & h_{32} & h_{33} & \dots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \\ & & & & h_{m+1,m} \end{pmatrix} = \begin{pmatrix} (Av_1,v_1) & (Av_2,v_1) & (Av_3,v_1) & \dots & (Av_m,v_1) \\ (Av_1,v_2) & (Av_2,v_2) & (Av_3,v_2) & \dots & (Av_m,v_2) \\ & (Av_2,v_3) & (Av_3,v_3) & \dots & (Av_m,v_3) \\ & & \ddots & \ddots & \vdots \\ & & & (Av_{m-1},v_m) & (Av_m,v_m) \\ & & & & (Av_m,v_{m+1}) \end{pmatrix}$$

   Consider the subdigonal entries of $\bar{H}_m$, and assume $h_{j+1,j} \neq 0$ for $j = 1 \dots m-1$. Show:

   a) $\mathcal{K}_m = \operatorname{span}\{v_1, \dots, v_m\}$ and $\dim \mathcal{K}_m = m$.

   b) $\bar{H}_m$ has full column rank: $\operatorname{rank} \bar{H}_m = m$

   c) If $h_{m+1,m} = 0$, then $A^m r_0 \in \mathcal{K}_m$ (!). Conclude that $\mathcal{K}_m = \mathcal{K}_{m+1} = \cdots = \mathcal{K}_n$.

3. Vollziehen Sie den Abschnitt über 'Block Arnoldi' aus dem Buch von Y. Saad nach (siehe Anhang), und erläutern Sie das in der Übung.

   Konkret: Weisen Sie nach, dass Algorithmus 6.21 die Block-Verallgemeinerung von Algorithm 9.1 aus dem Skriptum ist, wobei p die Dimension der Blöcke ist. Checken Sie insbesondere die Orthogonalitätseigenschaften der Blöcke V_j und zeigen Sie, dass (6.108) gilt.

   Anmerkung: In Algorithmus 6.21 bedeutet 'V unitary': $V^{\mathrm{T}}V = I_{p\times p}$. Mit '$Q$-$R$ factorization' ist die reduzierte QR-Zerlegung gemeint.

4. **a)** Lösen Sie ein Gleichungssystem $Ax = b$ (b irgendwie gewählt), mit

   $$A = \begin{pmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & 1 & \\ & & & & 1 & 1 \\ & & & & & 1 \end{pmatrix} \in \mathbb{R}^{1000\times 1000}$$

   mittels MATLAB/`gmres`. Plotten Sie den Verlauf des Residuums (2-Norm) über die Iterationsschritte hinweg ($m = 1, 2, \dots, n$) ggf. auf einer logarithmischen Skala. Was beobachten Sie?

b) Wiederholen Sie das Experiment aus **a)** mit

$$b = (1, 0, 0, 0, \ldots)^{\mathrm{T}}$$
$$b = (1, 1, 0, 0, \ldots)^{\mathrm{T}}$$
$$b = (1, 1, 1, 0, \ldots)^{\mathrm{T}}$$

usw.

c) Wie sieht es mit der Definitheit der Matrix A aus **a)** d.h., der Definitheit ihres Realteils aus? Was ändert sich, wenn die Diagonalelemente von A auf 2 gesetzt werden? Definitheit? Testen Sie wieder mit `gmres`, und vergleichen Sie die Abnahme der 2-Norm des Residuums mit der a priori-Abschätzung (10.15).

Anmerkung: Man kann die Definitheitskonstante angeben, indem man die Eigenwerte von $\mathrm{Re}(A)$ exakt angibt (diese ist eine tridiagonale Toeplitz-Matrix, so wie die 1D-Poisson-Matrix). Es genügt hier jedoch, wenn Sie das numerisch machen.

5. Für GMRES benötigt man die (für wachsendes m 'upgedatete') QR-Zerlegung der Matrix $\bar{H}_m$. Wie man dies mittels Givens-Rotationen durchführt, ist auf

   `en.wikipedia.org/wiki/Generalized_minimal_residual_method`

   beschrieben. Checken Sie das, schreiben Sie es auf, und erläutern Sie es.

6. **a)** Sei $\mathrm{Re}\, A > 0$. Zeigen Sie: Das 're-started' GMRES(m) konvergiert für beliebiges $m \geq 1$.

   b) 'Vorkonditionierung' bedeutet, dass man ein Krylov-Verfahren auf das Gleichungssystem $W^{-1}Ax = W^{-1}b$ anwendet (genauere Details später in VO), wobei $W \approx A$, W 'billig invertierbar'.

   Zeigen Sie: Falls A, W SPD sind und falls

   $$aW \leq A \leq bW$$

   gilt, dann ist

   $$\kappa_\sigma(W^{-1}A) \leq \frac{b}{a}$$

   wobei $\kappa_\sigma(M) = \lambda_{max}(M)/\lambda_{min}(M)$ die spektrale Konditionszahl einer Matrix M mit positivem Spektrum bezeichnet.

   Zeigen Sie auch

   $$\kappa_\sigma(W^{-1}A) = \kappa_A(W^{-1}A).$$

An explicit expression for the coefficient $C_m\left(\frac{a}{d}\right) \;/\; C_m\left(\frac{c}{d}\right)$ and an approximation are readily obtained from (6.99-6.100) by taking $\gamma = 0$:

$$\frac{C_m\left(\frac{a}{d}\right)}{C_m\left(\frac{c}{d}\right)} = \frac{\left(\frac{a}{d}+\sqrt{\left(\frac{a}{d}\right)^2-1}\right)^k + \left(\frac{a}{d}+\sqrt{\left(\frac{a}{d}\right)^2-1}\right)^{-k}}{\left(\frac{c}{d}+\sqrt{\left(\frac{c}{d}\right)^2-1}\right)^k + \left(\frac{c}{d}+\sqrt{\left(\frac{c}{d}\right)^2-1}\right)^{-k}} \approx \left(\frac{a+\sqrt{a^2-d^2}}{c+\sqrt{c^2-d^2}}\right)^k .$$

Since the condition number $\kappa_2(X)$ of the matrix of eigenvectors X is typically not known and can be very large, results of the nature of the corollary are of limited practical interest. They can be useful only when it is known that the matrix is nearly normal, in which case, $\kappa_2(X) \approx 1$.

6.12 BLOCK KRYLOV METHODS

In many circumstances, it is desirable to work with a block of vectors instead of a single vector. For example, out-of-core finite-element codes are more efficient when they are programmed to exploit the presence of a block of the matrix A in fast memory, as much as possible. This can be achieved by using block generalizations of Krylov subspace methods, for which A always operates on a group of vectors instead of a single vector. We begin by describing a block version of the Arnoldi algorithm.

ALGORITHM 6.21: Block Arnoldi

1. *Choose a unitary matrix V_1 of dimension $n \times p$.*
2. *For $j = 1, 2, \ldots, m$ Do:*
3. *Compute $H_{ij} = V_i^T A V_j \quad i = 1, 2, \ldots, j$*
4. *Compute $W_j = AV_j - \sum_{i=1}^{j} V_i H_{ij}$*
5. *Compute the Q-R factorization of W_j: $W_j = V_{j+1} H_{j+1,j}$*
6. *EndDo*

The above algorithm is a straightforward block analogue of Algorithm 6.1. By construction, the blocks generated by the algorithm are orthogonal blocks that are also orthogonal to each other. In the following we denote by I_k the $k \times k$ identity matrix and use the following notation:

$$\begin{aligned} U_m &= [V_1, V_2, \ldots, V_m], \\ H_m &= (H_{ij})_{1 \le i,j \le m}, \quad H_{ij} \equiv 0, \quad \text{for} \quad i > j+1, \\ E_m &= \text{matrix of the last } p \text{ columns of } I_n. \end{aligned}$$

Then, the following analogue of the relation (6.4) is easily proved:

$$AU_m = U_m H_m + V_{m+1} H_{m+1,m} E_m^T. \tag{6.108}$$

Here, the matrix H_m is no longer Hessenberg, but band-Hessenberg, meaning that it has p subdiagonals instead of only one. Note that the dimension of the subspace in which the solution is sought is not m but $m.p$.

A second version of the algorithm uses a modified block Gram-Schmidt procedure instead of the simple Gram-Schmidt procedure used above. This leads to a block generalization of Algorithm 6.2, the Modified Gram-Schmidt version of Arnoldi's method.

ALGORITHM 6.22: Block Arnoldi with Block MGS

1. *Choose a unitary matrix* V_1 *of size* $n \times p$
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. *Compute* $W_j := AV_j$
4. *For* $i = 1, 2, \ldots, j$ *do:*
5. $H_{ij} := V_i^T W_j$
6. $W_j := W_j - V_i H_{ij}$
7. *EndDo*
8. *Compute the Q-R decomposition* $W_j = V_{j+1} H_{j+1,j}$
9. *EndDo*

Again, in practice the above algorithm is more viable than its predecessor. Finally, a third version, developed by A. Ruhe [170] for the symmetric case (block Lanczos), yields a variant that is quite similar to the original Arnoldi algorithm. Assume that the initial block of p orthonormal vectors, $v_1, \ldots, v_p$ is available. The first step of the algorithm is to multiply v_1 by A and orthonormalize the resulting vector w against $v_1, \ldots, v_p$. The resulting vector is defined to be v_{p+1}. In the second step it is v_2 that is multiplied by A and orthonormalized against all available v_i's. Thus, the algorithm works similarly to Algorithm 6.2 except for a delay in the vector that is multiplied by A at each step.

ALGORITHM 6.23: Block Arnoldi–Ruhe's variant

1. *Choose* p *initial orthonormal vectors* $\{v_i\}_{i=1,\ldots,p}$.
2. *For* $j = p, p+1, \ldots, m$ *Do:*
3. *Set* $k := j - p + 1$*;*
4. *Compute* $w := Av_k$*;*
5. *For* $i = 1, 2, \ldots, j$ *Do:*
6. $h_{i,k} := (w, v_i)$
7. $w := w - h_{i,k} v_i$
8. *EndDo*
9. *Compute* $h_{j+1,k} := \|w\|_2$ *and* $v_{j+1} := w/h_{j+1,k}$.
10. *EndDo*

Observe that the particular case $p = 1$ coincides with the usual Arnoldi process. Also, the dimension m of the subspace of approximants, is no longer restricted to being a multiple