14 Arnoldi Iteration and GMRES

14.1 Arnoldi Iteration

The classical iterative solvers we have discussed up to this point were of the form

$$\boldsymbol{x}^{(k)} = G\boldsymbol{x}^{(k-1)} + \boldsymbol{c}$$

with constant G and $\boldsymbol{c}$. Such methods are also known as *stationary methods*. We will now study a different class of iterative solvers *based on optimization*.

All methods will require a "black box" implementation of a matrix-vector product. In most library implementations of such solvers the user can therefore provide a custom function which computes the matrix-vector product as efficiently as possible for the specific system matrix at hand.

One of the main ingredients in all of the following methods are *Krylov subspaces*.

Given $A \in \mathbb{C}^{m\times m}$ and $\boldsymbol{b} \in \mathbb{C}^m$ one generates

$$\{\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, A^3\boldsymbol{b}, \ldots\},$$

which is referred to as a *Krylov sequence*. Clearly, (fast) matrix-vector products play a crucial role in generating this sequence since each subsequent vector in the sequence is obtain from the previous one by multiplication by A.

With the Krylov sequence at hand one defines

$$\mathcal{K}_n = \text{span}\{\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, \ldots, A^{n-1}\boldsymbol{b}\}$$

as the n-th order *Krylov subspace*.

The Arnoldi iteration method to be derived will be applicable to both linear systems and eigenvalue problems, and therefore we are interested in re-examining similarity transformations of the form

$$A = QHQ^*,$$

where H is an upper Hessenberg matrix.

In our earlier work we used Householder reflectors to transform A to upper Hessenberg form. This had its advantages since the resulting algorithm is a stable one. When studying the QR factorization we also looked at the modified Gram-Schmidt algorithm. That algorithm was less stable. However, it has the advantage that one get one column of the unitary matrix Q one column at a time, i.e., the modified Gram-Schmidt algorithm can be stopped at any time and yields a partial set of orthonormal column vectors. On the other hand, with Householder reflectors we always have to perform the entire QR factorization before we get (all) orthonormal vectors.

Thus, Arnoldi iteration can be seen as the use of the modified Gram-Schmidt algorithm in the context of Hessenberg reduction.

14.2 Derivation of Arnoldi Iteration

We start with the similarity transformation $A = QHQ^*$ with $m \times m$ matrices A, Q, and H. Clearly, this is equivalent to

$$AQ = QH.$$

Now we take $n < m$, so that the eigenvalue equations above can be written as

$$A[\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n, \boldsymbol{q}_{n+1}, \ldots, \boldsymbol{q}_m] = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n, \boldsymbol{q}_{n+1}, \ldots, \boldsymbol{q}_m] \times$$
$$\times \begin{bmatrix} h_{11} & h_{12} & & \ldots & & h_{1n} & \ldots & & h_{1m} \\ h_{21} & h_{22} & & \ldots & & h_{2n} & \ldots & & h_{2m} \\ 0 & h_{32} & h_{33} & & & h_{3n} & \ldots & & h_{3m} \\ & 0 & h_{43} & \ddots & & & & & \\ & & 0 & \ddots & h_{n-1,n-2} & \vdots & & & \vdots \\ \vdots & & & \ddots & h_{n,n-1} & h_{nn} & & & \\ & & & & 0 & h_{n+1,n} & & & \\ & & & & & 0 & \ddots & \ddots & \\ 0 & & & \ldots & & & 0 & h_{m,m-1} & h_{mm} \end{bmatrix}.$$

Next, we consider only part of this system. Namely, we let

$$\begin{aligned} Q_n &= [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n] \\ Q_{n+1} &= [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n, \boldsymbol{q}_{n+1}] \\ \widetilde{H}_n &= \begin{bmatrix} h_{11} & h_{12} & & \ldots & & h_{1n} \\ h_{21} & h_{22} & & \ldots & & h_{2n} \\ 0 & h_{32} & h_{33} & & & h_{3n} \\ & 0 & h_{43} & \ddots & & \\ & & 0 & \ddots & h_{n-1,n-2} & \vdots \\ \vdots & & & \ddots & h_{n,n-1} & h_{nn} \\ & & & & 0 & h_{n+1,n} \end{bmatrix} \end{aligned}$$

and then take

$$AQ_n = Q_{n+1}\widetilde{H}_n.$$

Note that here $A \in \mathbb{C}^{m\times m}$, $Q_n \in \mathbb{C}^{m\times n}$, $Q_{n+1} \in \mathbb{C}^{m\times n+1}$, and $\widetilde{H}_n \in \mathbb{C}^{n+1\times n}$ so that both sides of the equation result in an $m \times n$ matrix.

If we compare the n-th columns on both sides, then we get

$$A\boldsymbol{q}_n = h_{1n}\boldsymbol{q}_1 + h_{2n}\boldsymbol{q}_2 + \ldots + h_{nn}\boldsymbol{q}_n + h_{n+1,n}\boldsymbol{q}_{n+1} \tag{40}$$

which constitutes an $(n+1)$ term recursion for the vector $\boldsymbol{q}_{n+1}$. Equation (40) can be re-written as

$$\boldsymbol{q}_{n+1} = \frac{A\boldsymbol{q}_n - \sum_{i=1}^{n} h_{i,n}\boldsymbol{q}_i}{h_{n+1,n}}. \tag{41}$$

The recursive computation of the columns of the unitary matrix Q in this manner is known as *Arnoldi iteration.*

Example The first step of Arnoldi iteration proceeds as follows. We start with the matrix A and an arbitrary normalized vector $\boldsymbol{q}_1$. Then, according to (41),

$$\boldsymbol{q}_2 = \frac{A\boldsymbol{q}_1 - h_{11}\boldsymbol{q}_1}{h_{21}}.$$

Note that this step involves the matrix-vector product $A\boldsymbol{q}_1$ (which has to be computed efficiently with a problem specific subroutine).

Since we want $\boldsymbol{q}_1^*\boldsymbol{q}_2 = 0$ in order to have orthogonality of the columns of Q we get

$$0 = \boldsymbol{q}_1^* A\boldsymbol{q}_1 - h_{11}\boldsymbol{q}_1^*\boldsymbol{q}_1.$$

Taking advantage of the normalization of $\boldsymbol{q}_1$ this is equivalent to

$$h_{11} = \boldsymbol{q}_1^* A\boldsymbol{q}_1$$

– a Rayleigh quotient.

Finally, we let $\boldsymbol{v} = A\boldsymbol{q}_1 - h_{11}\boldsymbol{q}_1$, compute $h_{21} = \|\boldsymbol{v}\|$, and normalize

$$\boldsymbol{q}_2 = \frac{\boldsymbol{v}}{h_{21}}.$$

If we compare (15) with the formula

$$\boldsymbol{q}_n = \frac{\boldsymbol{a}_n - \sum_{i=1}^{n-1} r_{i,n}\boldsymbol{q}_i}{r_{nn}}$$

that we used earlier for the Gram-Schmidt method (cf. (17)), then it is clear that an algorithm for Arnoldi iteration will be similar to that for the modified Gram-Schmidt algorithm:

Algorithm (Arnoldi Iteration)

Let $\boldsymbol{b}$ be an arbitrary initial vector

$\boldsymbol{q}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$

for $n = 1, 2, 3, \ldots$

$\boldsymbol{v} = A\boldsymbol{q}_n$

for $j = 1 : n$

$h_{jn} = \boldsymbol{q}_j^*\boldsymbol{v}$

$\boldsymbol{v} = \boldsymbol{v} - h_{jn}\boldsymbol{q}_j$

end

$h_{n+1,n} = \|\boldsymbol{v}\|_2$

$\boldsymbol{q}_{n+1} = \boldsymbol{v}/h_{n+1,n}$

end

Remark The most expensive operation in the algorithm is the matrix-vector product $A\boldsymbol{q}_n$. The rest of the operations are on the order of $\mathcal{O}(mn)$ (so they get a little more expensive in each iteration). Therefore, in addition to the basic Arnoldi algorithm we need an efficient implementation of the matrix-vector product this should be taylored to the problem. Moreover, the algorithm above treats the matrix-vector product as a "black box" and the algorithm does not need to know or store the matrix A. The only quantity of interest is the product $A\boldsymbol{q}_n$, i.e., the action of A on $\boldsymbol{q}_n$.

14.3 Arnoldi Iteration as Projection onto Krylov Subspaces

An alternative derivation of Arnoldi iteration starts with the *Krylov matrix*

$$K_n = [\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, \ldots, A^{n-1}\boldsymbol{b}].$$

Then

$$AK_n = [A\boldsymbol{b}, A^2\boldsymbol{b}, A^3\boldsymbol{b} \ldots, A^n\boldsymbol{b}]. \tag{42}$$

Since the first $n-1$ columns of the matrix on the right-hand side are the last $n-1$ columns of K_n we can also write

$$AK_n = K_n[\boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n, -\boldsymbol{c}],$$

where

$$\boldsymbol{c} = -K_n^{-1}A^n\boldsymbol{b}$$

and we assume that K_n is invertible. Equivalently,

$$AK_n = K_nC_n$$

with the upper Hessenberg matrix

$$C_n = [\boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n, -\boldsymbol{c}].$$

This A and C_n are similar via $K_n^{-1}AK_n = C_n$. The problem with this formulation is that K_n is usually ill-conditioned (since all of its columns converge to the dominant eigenvector of A, cf. our earlier discussion of simultaneous power iteration).

Remark As a side remark we mention that the matrix C_n above is known as a *companion matrix*. The matrix has characteristic polynomial $p(z) = z^n + \sum_{i=1}^{n} c_i z^{i-1}$, where the c_i are the components of $\boldsymbol{c}$. In other words, the eigenvalues of C_n are the roots of p. This goes also in the other direction. Given a monic polynomial p, we can form its companion matrix C_n and then know that the roots of p are the same as the eigenvalues of C_n.

Returning to the derivation of Arnoldi iteration, we still need to show how the above Krylov subspace formulation is related to the earlier one based on the Gram-Schmidt method.

Denote the QR factorization of the Krylov matrix K_n by

$$K_n = Q_nR_n.$$

Then

$$\begin{aligned} & K_n^{-1}AK_n = C_n \\ \iff \quad & R_n^{-1}Q_n^*AQ_nR_n = C_n \\ \iff \quad & Q_n^*AQ_n = \underbrace{R_nC_nR_n^{-1}}_{=H_n}. \end{aligned}$$

It needs to be pointed out that this approach is computationally not a good one. It is both too expensive and unstable. On the one hand, finding $\boldsymbol{c}$ involves solution of the (ill-conditioned) linear system $K_n\boldsymbol{c} = A^n\boldsymbol{b}$. On the other hand, we would also be required to provide the inverse of R_n.

However, we can get some theoretical insight from this approach. The formula

$$Q_n^* A Q_n = H_n$$

can be interpreted as an *orthogonal projection of A onto K_n* with the column of Q_n as basis, i.e., Arnoldi iteration is nothing but orthogonal projection onto Krylov subspaces.

Remark If A is Hermitian, then everything above simplifies (e.g., Hessenberg matrices turn into tridiagonal), and we get what is know in the literature separately as *Lanczos iteration*.

14.4 GMRES

The method of *generalized minimum residuals* (or GMRES) was suggested in 1986 by Saad and Schultz.

While application of the classical iterative solvers was limited to either diagonally dominant or positive definite matrices, the GMRES method can be used for linear systems $A\boldsymbol{x} = \boldsymbol{b}$ with *arbitrary (nonsingular) square matrices* A. The essential ingredient in this general iterative solver is Arnoldi iteration.

The main idea of the GMRES method is to solve a *least squares problem* at each step of the iteration. More precisely, at step n we approximate the exact solution $\boldsymbol{x}^* = A^{-1}\boldsymbol{b}$ by a vector $\boldsymbol{x}_n \in \mathcal{K}_n$ (the n-th order Krylov subspace) such that the residual

$$\|\boldsymbol{r}_n\|_2 = \|A\boldsymbol{x}_n - \boldsymbol{b}\|_2$$

is minimized. We now describe how to solve this least squares problem.

We start with the Krylov matrix

$$K_n = \big[\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, \ldots, A^{n-1}\boldsymbol{b}\big] \in \mathbb{C}^{m\times n}.$$

Thus the column space of AK_n is $A\mathcal{K}_n$.

Now the desired vector $\boldsymbol{x}_n \in \mathcal{K}_n$ can be written as

$$\boldsymbol{x}_n = K_n\boldsymbol{c}$$

for some appropriate vector $\boldsymbol{c} \in \mathbb{C}^n$. Therefore the residual minimization becomes

$$\|\boldsymbol{r}_n\|_2 = \|A\boldsymbol{x}_n - \boldsymbol{b}\|_2 = \|AK_n\boldsymbol{c} - \boldsymbol{b}\|_2 \to \min.$$

The obvious way to find the least squares solution to this problem would be to compute the QR factorization of the matrix AK_n. However, this is both unstable and too expensive.

Instead, we look for an orthonormal basis for the Krylov subspace $\mathcal{K}_n$. We will denote this by $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n\}$, the columns of the matrix Q_n used in Arnoldi iteration. With this new basis the approximate solution $\boldsymbol{x}_n \in \mathcal{K}_n$ can be written as

$$\boldsymbol{x}_n = Q_n\boldsymbol{y}$$

for some appropriate vector $\boldsymbol{y} \in \mathbb{C}^n$. The residual minimization is then

$$\|A\boldsymbol{x}_n - \boldsymbol{b}\|_2 = \|AQ_n\boldsymbol{y} - \boldsymbol{b}\|_2 \to \min. \tag{43}$$

It is now time to recall the principle behind the Arnoldi iteration. That algorithm is based on the partial similarity transform

$$AQ_n = Q_{n+1}\widetilde{H}_n.$$

Thus (43) turns into

$$\|Q_{n+1}\widetilde{H}_n\boldsymbol{y} - \boldsymbol{b}\|_2 \to \min.$$

Next we take advantage of the fact that multiplication by a unitary matrix does not change the 2-norm. Thus, we arrive at

$$\begin{aligned} & \|Q_{n+1}^*Q_{n+1}\widetilde{H}_n\boldsymbol{y} - Q_{n+1}^*\boldsymbol{b}\|_2 \to \min \\ \iff \qquad & \|\widetilde{H}_n\boldsymbol{y} - Q_{n+1}^*\boldsymbol{b}\|_2 \to \min \end{aligned}$$

Note that the system matrix AQ_n in (43) is an $m \times n$ matrix, while the new matrix $\widetilde{H}_n$ is an $(n+1) \times n$ matrix which is smaller, and therefore will permit a more efficient solution.

The final simplification we can make is for the vector $Q_{n+1}^*\boldsymbol{b}$. In detail, this vector is given by

$$Q_{n+1}^*\boldsymbol{b} = \begin{bmatrix} \boldsymbol{q}_1^*\boldsymbol{b} \\ \boldsymbol{q}_2^*\boldsymbol{b} \\ \vdots \\ \boldsymbol{q}_{n+1}^*\boldsymbol{b} \end{bmatrix}.$$

Recall that the Krylov subspaces are given by

$$\begin{aligned} \mathcal{K}_1 &= \operatorname{span}\{\boldsymbol{b}\}, \\ \mathcal{K}_2 &= \operatorname{span}\{\boldsymbol{b}, A\boldsymbol{b}\}, \\ &\vdots \quad , \end{aligned}$$

and that the columns $\boldsymbol{q}_j$ of Q_n form an orthonormal basis for $\mathcal{K}_n$. Thus

$$\boldsymbol{q}_1 = \frac{\boldsymbol{b}}{\|\boldsymbol{b}\|}$$

and $\boldsymbol{q}_j^*\boldsymbol{b} = 0$ for any $j > 1$. Therefore we actually have

$$Q_{n+1}^*\boldsymbol{b} = \|\boldsymbol{b}\|\boldsymbol{e}_1.$$

Combining all of this work we arrive at the final least squares formulation

$$\begin{gathered} \left\| \widetilde{H}_n\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1 \right\|_2 \to \min \\ \text{with} \quad \boldsymbol{x}_n = Q_n\boldsymbol{y}. \end{gathered}$$

This leads to

Algorithm (GMRES)

Let $\boldsymbol{q}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|$

for $n = 1, 2, 3, \ldots$

Perform step n of Arnoldi iteration, i.e., compute new entries for $\widetilde{H}_n$ and Q_n.

Find $\boldsymbol{y}$ that minimizes $\left\|\widetilde{H}_n\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\right\|_2$ (e.g., with QR factorization)

Set $\boldsymbol{x}_n = Q_n\boldsymbol{y}$

end

The computational cost for the GMRES algorithm depends on the cost of the method used for the least squares problem and that of the Arnoldi iteration. Since the least squares system matrix is of size $(n+1) \times n$ it can be done in $\mathcal{O}(n^2)$ floating point operations. If an updating QR factorization (which we did not discuss) is used, then even $\mathcal{O}(n)$ is possible. Arnoldi iteration takes $\mathcal{O}(mn)$ operations plus the cost for matrix-vector multiplications. These can usually be accomplished at somewhere between $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$ flops.

14.4.1 Convergence of GMRES

In *exact* arithmetic (which we of course do not have in a numerical computing environment) GMRES iteration will always converge in at most m steps (since $\mathcal{K}_m = \mathbb{C}^m = \text{range}(A)$). Moreover, convergence is monotonic since $\|\boldsymbol{r}_{n+1}\| \leq \|\boldsymbol{r}_n\|$. This latter fact is true since $\boldsymbol{r}_n$ is minimized over $\mathcal{K}_n$, and we have $\mathcal{K}_{n+1} \supset \mathcal{K}_n$. Thus, minimization over a larger subspace will allow us to achieve a smaller residual norm.

For practical computations the only case of interest is when the algorithm converges (to within a specified tolerance) in n iterations with $n \ll m$. Usually the *relative residual* is used to control the convergence, i.e., we check whether

$$\frac{\|\boldsymbol{r}_n\|}{\|\boldsymbol{b}\|} < \text{tol},$$

where $\text{tol} = 10^{-6}$ or 10^{-8}. The MATLAB code `GMRESDemo.m` illustrates this.

The test problems are given by 200×200 systems with random matrices and random right-hand side vectors $\boldsymbol{b}$. The different test matrices are a matrix A whose entries are independent samples from the real normal distribution of mean 2 and standard deviation $0.5/\sqrt{200}$. Its eigenvalues are clustered in a disk in the complex plane of radius $1/2$ centered at $z = 2$. The other test matrices have different eigenvalue distributions. They are obtained as

- $B = A + D$, where the entries of the diagonal matrix D are the complex numbers

$$d_k = (-2 + 2\sin\theta_k) + i\cos\theta_k, \qquad \theta_k = \frac{k}{m-1},\ 0 \leq k \leq m-1.$$

- C, a random matrix whose eigenvalues are loosely clustered in the unit disk.

- $D = C^T C$, a random symmetric positive definite matrix (with real positive eigenvalues).
- $E = \frac{1}{2}C$, another random matrix whose eigenvalues are more densely clustered than those of C.

The rate of convergence of the GMRES method depends on the distribution of the eigenvalues of A in the complex plane. For fast convergence the eigenvalues need to be *clustered away from the origin*. Note that the eigenvalue distribution is much more important than the condition number of A, which is the main criterion for rapid convergence of the conjugate gradient method (see next section).

If the GMRES method is applied to a matrix with a "good" eigenvalue distribution (or is appropriately *preconditioned*) then it may solve even a system with an unstructured dense matrix faster than standard LU factorization.