Iterative Methods for Solving Linear Systems

Chen Greif

UBC/CS, Fall 2013

When it comes to the solution of sparse large linear systems, direct methods (the Gaussian elimination approach) have some serious drawbacks:

1. *Fill-in.* Even if a matrix is narrow-banded, within the band many zeros could become nonzero during the elimination: bad when the band is sparse.

2. *Can't solve inexactly.* Sometimes we do not really need to solve exactly, e.g., when solving the linear system is an iteration within a nonlinear solver.

3. *Can't use a 'good' initial guess.* For example in time-dependent problems, when we already know the solution at a previous time step and know it might be very close to the solution at the current time step.

4. *Can't be used if the linear operator is given implicitly.* Often only the operator is known and it is prohibitively expensive to compute the associated matrix. In this case we need an approach that relies on matrix-vector products.

1 Stationary methods

There is more than one way to derive or motivate stationary schemes. For example, given $Ax = b$, we can rewrite as $x = (I - A)x + b$, which yields the iteration

$$x_{k+1} = (I - A)x_k + b.$$

From this we can generalize: for a given *splitting* $A = M - N$, we have $Mx = Nx + b$, which leads to the fixed point iteration

$$Mx_{k+1} = Nx_k + b.$$

The first equation above is just a special case of the second, more general equation, with $M = I$, and is known as a *Richardson iteration*. (The classical Richardson iteration includes a parameter and will be introduced soon.)

1.1 Basic stationary schemes

What motivates us to use iterative schemes is the possibility that inverting A may be very difficult, to the extent that it may be worthwhile to invert a much 'easier' matrix several times, rather than inverting A directly only once. Suppose, indeed, that $A = M - N$ is a splitting, and that $Mx_0 = b$ is much easier to solve than $Ax = b$. How can we correct x_0 to make it closer to the true solution x? If

$$e_0 = x - x_0$$

is the error, then

$$Ae_0 = b - Ax_0 := r_0.$$

Notice that r_0 is computable whereas e_0 is not, because x is not available. Clearly,

$$x = x_0 + e_0 = x_0 + A^{-1}r_0,$$

but this in itself does not help because it requires solving for A which defeats the purpose. So, instead of solving the new system by inverting A, we use the idea of solving a system with M. We thus set

$$M\tilde{e}_0 = r_0,$$

and then

$$x_1 = x_0 + \tilde{e}_0$$

is our new guess. x_1 is hopefully closer to x than x_0.

At this point we have an *iterative scheme*:

$$x_{k+1} = x_k + M^{-1}(b - Ax_k) \ , \quad k = 0, 1, 2, \ldots$$

The scheme is identical to the scheme presented at the top of this page, with $N = M - A$. It is called *stationary* because we are basically looking for

a stationary vector or a *fixed point* that satisfies $x = g(x)$, where $g(x) = x + M^{-1}(b - Ax)$, and we are doing so by iterating $x_{k+1} = g(x_k)$.

Denote the diagonal, strictly lower triangular, and strictly upper triangular parts of A by D, E, and F respectively, so that $A = D + E + F$. Here are some basic, well-known stationary schemes:

- Jacobi: The scheme corresponds to the choice $M = D$. In componentwise form

  for $i = 1 : n$

  $$x_{k+1}^{(i)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_k^{(j)} \right]$$

  end

- Gauss-Seidel: $M = D + E$, and so we have

  for $i = 1 : n$

  $$x_{k+1}^{(i)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j<i} a_{ij} x_{k+1}^{(j)} - \sum_{j>i} a_{ij} x_k^{(j)} \right]$$

  end

- SOR: $M = \frac{1}{\omega}(D + \omega E)$, and so here we have

  for $i = 1 : n$

  $$x_{k+1}^{(i)} = (1 - \omega) x_k^{(i)} + \frac{\omega}{a_{ii}} \left[b_i - \sum_{j<i} a_{ij} x_{k+1}^{(j)} - \sum_{j>i} a_{ij} x_k^{(j)} \right]$$

  end

In general the matrix M defining the splitting should satisfy two somewhat contradictory requirements:

- $Mz = r$ is much easier to solve, compared to $Ax = b$;
- $M^{-1} \approx A^{-1}$ (in a slightly vague, nonetheless intuitive, sense).

How do we know that a certain choice of M will yield a scheme that converges to the solution? Convergence analysis for stationary schemes is straightforward and fairly complete. Since $Mx = Nx + b$, it is easy to show that $M(x - x_{k+1}) = N(x - x_k)$, and if indeed $e_k = x - x_k$ then we have

$$e_k = T^k e_0,$$

where

$$T = M^{-1}N$$

is called *the iteration matrix*. Since we want $e_k \to 0$ as $k \to \infty$, it is sufficient to require

$$\|e_k\| \leq \|T\|^k \|e_0\|.$$

Theorem. A necessary and sufficient condition for the convergence of a stationary scheme with any initial guess is $\rho(M^{-1}N) < 1$, where ρ is the *spectral radius*:

$$\rho(T) = \max_i |\lambda_i(T)|.$$

Computational work. Asymptotically, $\|e_k\| \approx \rho^k \|e_0\|$ and so the smaller ρ is, the faster we converge. We can roughly estimate how many iterations it takes to reduce the initial error by a factor of 10^m as follows:

$$\frac{\|e_k\|}{\|e_0\|} \approx \rho^k \approx 10^{-m}.$$

Therefore,

$$-\log_{10} \rho \approx \frac{m}{k}.$$

The quantity $-\ln \rho$ (which is just a constant multiple of the expression on the left hand side above) is known as the *asymptotic rate of convergence.*

Some properties of Jacobi, Gauss-Seidel, and SOR. (brief)

1. They converge for any initial guess if the matrix is diagonally dominant.
2. SOR(ω) converges for $0 < \omega < 2$ is A is SPD. The condition $0 < \omega < 2$ is a necessary condition for the scheme to converge, for any given matrix.
3. For the classical finite difference discretized Poisson equation in 2D, we can show that:
   (a) Both jacobi and Gauss-Seidel converge within $O(n^2)$ iterations. Gauss-Seidel is faster than Jacobi by a factor of approximately 2.

(b) SOR with the optimal parameter converges within $O(n)$ iterations (see more on this in the next item).

4. For a certain class of *consistently ordered* matrices, there exists a beautiful theory for SOR, due to Young. The optimal SOR parameter, ω_{opt}, is known analytically in terms of the spectral radius of the Jacobi iteration matrix, T_J:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(T_J)^2}}.$$

The spectral radius of the SOR iteration matrix for this value of ω_{opt} is $\omega_{opt} - 1$!

1.2 The Richardson method and extensions

The Richardson iteration is probably the simplest stationary scheme, but it allows for an elegant and transparent analysis that can nicely motivate the introduction of the modern *Krylov subspace methods*. The latter will keep us busy for a while later on.

In its simplest form the scheme is given by

$$x_{k+1} = x_k + \alpha r_k,$$

where α is a parameter and as before $r_k = b - Ax_k$. We would like to find the optimal parameter, which we will denote by α_{opt}.

The above scheme corresponds to a standard stationary scheme with a splitting $A = M - N$ where $M^{-1} = \alpha I$. Suppose A is symmetric positive definite and let

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0$$

denote the eigenvalues of A. Then

$$T = I - \alpha A$$

and hence

$$\mu_i = 1 - \alpha\lambda_i,$$

where μ_i denote the eigenvalues of T. For the optimal α we must have that the positive and negative eigenvalues on both sides of the spectrum are equal in magnitude:

$$1 - \alpha_{\text{opt}}\lambda_n = -(1 - \alpha_{\text{opt}}\lambda_1),$$

and this yields

$$\alpha_{\text{opt}} = \frac{2}{\lambda_1 + \lambda_n}.$$

It is straightforward from here to see that

$$\rho(T) = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = \frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}.$$

This shows that asymptotically, we will be in trouble if the condition number of the matrix A is large: not only will we expect to have accuracy problems, we may also be facing an extremely slow rate of convergence.

1.3 Elements of Krylov subspaces

For the generic form of stationary schemes we have $x_{k+1} = x_k + M^{-1}r_k$, and so it is straightforward to see that if $\tilde{A} = M^{-1}A$ and $\tilde{b} = M^{-1}b$, any stationary scheme can be considered as a Richardson scheme applied to the system $\tilde{A}x = \tilde{b}$ with $\alpha = 1$. The matrix M is often referred to as a *preconditioner*; an important notion as we shall see soon. Thus, without loss of generality, let us consider for a few brief moments the case $\alpha = 1$:

$$x_{k+1} = x_k + r_k.$$

Adding b to both sides and subtracting the equations multiplied by A, we obtain expressions for the residuals, as follows:

$$b - Ax_{k+1} = b - Ax_k - Ar_k.$$

In other words, we get $r_{k+1} = (I - A)r_k$, which gives us the relation

$$r_k = (I - A)^k r_0.$$

We thus conclude that the residuals satisfy a polynomial relation of the sort

$$r_k = p_k(A)r_0$$

with $p_k(0) = 1$. In other words, all residuals can be expressed as powers of A times the initial residual. Note that for convergence we must have $\|I - A\| < 1$ (why?). This restriction will be dropped for more advanced schemes.

It is now straightforward to show that the errors and the iterates also are connected by a similar relation. Indeed, since $Ae_k = r_k$, we have

$$A(x - x_k) = p_k(A)A(x - x_0)$$

and since A is assumed to be nonsingular we have $x - x_k = p_k(A)(x - x_0)$. For the iterates, we note that from $x_{k+1} = x_k + r_k$ it follows that

$$x_{k+1} = x_0 + r_0 + r_1 + \cdots + r_k = x_0 + \sum_{i=0}^{k} (I - A)^i r_0.$$

This shows us that in fact the kth iterate is nothing but a shift by x_0 of a certain space spanned by powers of A applied to the initial residual. This in fact forms what is known as a Krylov subspace. Later on we will provide a formal definition and see how this is useful.

Suppose now that A has a complete set of n eigenpairs $\{\lambda_i, v_i\}$, and suppose the initial residual satisfies

$$r_0 = \sum_{i=1}^{n} \alpha_i v_i.$$

Then

$$r_k = p_k(A)r_0 = \sum_{i=1}^{n} \alpha_i p_k(A)v_i = \sum_{i=1}^{n} \alpha_i p_k(\lambda_i)v_i.$$

So, this shows that the residual reduction depends on how well the polynomial damps the eigenvalues. In the case of Richardson the polynomial is given and there is nothing we can do about it. But this really shows what we can do in general: the basic idea of modern methods is to construct a "good" polynomial in this sense.

So, let us give it another shot, using multiple parameters. It may be useful to find a good polynomial that damps the components of the initial error or residual. With one parameter, though, it is impossible to do much. To that end, consider a non-stationary scheme that relies on setting up a different parameter α_k in every iteration as follows:

$$x_{k+1} = x_k + \alpha_k r_k.$$

It readily follows that

$$e_{k+1} = (I - \alpha_k A)e_k,$$

and hence

$$e_k = \prod_{i=0}^{k-1}(I - \alpha_i A)e_0 = p_k(A)e_0.$$

Again, we must have that $p_k(0) = 1$. Taking norms, we get

$$\frac{\|e_k\|}{\|e_0\|} \leq \|p_k(A)\|.$$

Suppose A is unitarily diagonalizable, i.e. there exists an orthogonal matrix Q (that is $QQ^T = I$) such that $A = Q\Lambda Q^T$. Then $p_k(A) = Qp_k(\Lambda)Q^T$, and we have that

$$\frac{\|e_k\|}{\|e_0\|} \leq \|p_k(\Lambda)\|.$$

But since $p_k(\Lambda)$ is just a diagonal matrix whose elements are $p_k(\lambda_i)$, we have that

$$p_k(\Lambda) \leq \max_{1\leq i\leq n} |p_k(\lambda_i)|.$$

So, what we may want can be posed as a problem in approximation theory. Suppose A has real eigenvalues. Then we may seek polynomials that satisfy a min-max property:

$$\min_{p_k(0)=1} \max_{\lambda_n\leq\lambda\leq\lambda_1} |p_k(\lambda)|.$$

This may indicate that an appropriate choice for good polynomials may be the Chebyshev polynomials.

2 Polynomial acceleration

We have our basic iterative scheme, which can be thought of as a two-stage scheme:

$$Mz_k = b - Ax_k = r_k,$$

followed by

$$x_{k+1} = x_k + z_k.$$

We now ask whether we can do any better with the information we have so far on the iterates. We may *accelerate* the convergence by forming a linear combination of the iterates:

$$y_k = \sum_{i=0}^{k} \alpha_{k,i} x_i,$$

with $\sum_i \alpha_{k,i} = 1$ for consistency. Denote the error for this new scheme as e_k. We thus have:

$$e_k = x - y_k = \sum_{i=0}^{k} \alpha_{k,i}(x - x_i) = \sum_i \alpha_{k,i} e_i = \left(\sum_i \alpha_{k,i} T^i \right) e_0 = p_k(T) e_0.$$

In other words, our error is a polynomial in the iteration matrix T, multiplied by the initial error: $e_k = p_k(T)e_0$. p_k is of degree k, and must satisfy $p_k(1) = 1$. What sort of polynomial do we want? One possibility is to take one that will make $\|p_k(T)z\|$ small over *all* vectors z.

Suppose $\{v_i\}$ is the (complete) set of eigenvectors of T, and let $z = \sum \beta_i v_i$. Then $Tz = \sum_i \lambda_i \beta_i v_i$, and so

$$p_k(T)z = \sum_i p_k(\lambda_i)\beta_i v_i.$$

Thus we are interested in a polynomial $p_k(x)$ that will be small for all values x that are eigenvalues of T. More generally, we can use a polynomial that is small on an interval R that contains all the eigenvalues of T. A few things to note:

- we assume that the eigenvalues of T are real;
- we assume the eigenvalues are inside an interval not containing 1;
- so far, the discussion implies that we may need to store all the iterates;
- we need to use a different set of coefficient $\{\alpha_{k,i}\}$ for each k.

Natural polynomial candidates are *orthogonal polynomials*, which satisfy the relation

$$p_{k+1}(x) = (\alpha_k x + \beta_k)p_k(x) + \gamma_k p_{k-1}(x).$$

Then,

$$p_{k+1}(T)e_0 = (\alpha_k T + \beta_k I)p_k(T)e_0 + \gamma_k p_{k-1}(T)e_0.$$

Hence,

$$x - y_{k+1} = (\alpha_k T + \beta_k I)(x - y_k) + \gamma_k(x - y_k - 1).$$

$$y_{k+1} = (\alpha_k T + \beta_k I)y_k + \gamma_k y_{k-1} + (1 - \beta_k - \gamma_k)x - \alpha_k T x.$$

Since $p_k(1) = 1$, we must have $\alpha_k + \beta_k + \gamma_k = 1$, and the last two terms in the above equation for y_{k+1} are equal to $\alpha_k(I - T)x = \alpha_k M^{-1}Ax = \alpha_k M^{-1}b$. Also, $\alpha_k T y_k = \alpha_k(I - M^{-1}A)y_k$, so

$$y_{k+1} = \alpha_k y_k + \alpha_k M^{-1}(b - Ay_k) + \beta_k y_k + \gamma_k y_{k-1}.$$

Using again the connection $\alpha_k + \beta_k + \gamma_k = 1$, we can express our new iteration as:

1. Solve $Mz_k = b - Ay_k$.
2. Form $y_{k+1} = y_k + \alpha_k z_k - \gamma_k(y_k - y_{k-1})$.

Note: This is a straightforward generalization of the basic stationary iteration, where $\alpha_k = 1$ and $\gamma_k = 0$. Choosing the Chebyshev polynomials, for example, will lead to the Chebyshev method discussed in class.

3 Krylov subspaces

We have seen above that it may make sense to seek a solution within a subspace that is actually known by the name *Krylov subspace*:

$$x_k \in x_0 + \mathcal{K}^k(A; r_0) = x_0 + span\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\}.$$

Without loss of generality, we can assume that $x_0 = 0$ (otherwise consider $Ay = b - Ax_0$ and take $y_0 = 0$). In that case $r_0 = b$ and we can write

$$x_k \in \mathcal{K}^k(A; b) = span\{b, Ab, A^2 b, \ldots, A^{k-1} b\}.$$

There are two immediate concerns we need to address:

1. We will need to make sure that we have a good basis for the subspace. This is because whatever method we decide upon will require expressing the iterates as linear combinations of vectors that span the Krylov subspace.
2. We will need to decide on an optimality criterion to impose, in order to obtain a "good" solution within the subspace.

3.1 Computing an orthogonal basis for the Krylov subspace

We need to spend some time thinking about an appropriate basis for the Krylov subspace because the most obvious choice of vectors, namely $\{b, Ab, A^2b, \ldots, A^{k-1}b\}$, is bad for exactly the reason the power method works! As k grows larger, vectors of the form A^kb approach the dominant eigenvector of A.

Define $u_j = A^{j-1}r_0$, $j = 0, \ldots, k$, and

$$U_k = [u_1, u_2, \ldots, u_k] = [r_0, Ar_0, \ldots, A^{k-1}r_0].$$

Then

$$AU_k = [u_2, u_3, \ldots, u_{k+1}] = U_kB_k + u_{k+1}e_k^T,$$

where B_k is a matrix with the subdiagonal immediately below the main diagonal having all 1s, and all zeros elsewhere. For example, for $k = 3$ we have

$$\begin{aligned} AU_3 &= A[u_1, u_2, u_3] = [Au_1, Au_2, Au_3] = [u_2, u_3, u_4] \\ &= [u_1, u_2, u_3] \cdot \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + u_4 \cdot (0, 0, 1). \end{aligned}$$

Suppose $U_k = Q_kR_k$ is a QR factorization of U_k, i.e. Q_k is $n \times k$ and orthogonal, and R_k is a $k \times k$ upper triangular matrix. Then

$$AQ_kR_k = Q_kR_kB_k + u_{k+1}e_k^T,$$

from which it follows, by multiplying by R_k^{-1} on the right, that $AQ_k = Q_kC_k + u_{k+1}e_k^TR_k^{-1}$, where $C_k = R_kB_kR_k^{-1}$. Now, since C_k is upper Hessenberg (why?) and $u_{k+1}e_k^T$ is nonzero only on its last column, it follows that there exists an upper Hessenberg matrix of size $k \times k$, denote it by $\tilde{H}_k$, such that

$$AQ_k = Q_k\tilde{H}_k + \frac{1}{r_{k,k}}u_{k+1}e_k^T,$$

where $\tilde{H}_k$ is upper Hessenberg. This interesting structure leads us to expect that the construction of an orthogonal basis for the Krylov subspace will produce as a side result also an upper Hessenberg matrix that will contain the 'coordinates' of A with respect to the orthogonal basis.

The Arnoldi method

We now explain how to formally construct the basis. Basically, we do it by brute force. Setting $AQ = QH$ (when we later discuss Householder transformations we will be persuaded that this can be done for any matrix) and construct Q column by column. The nice thing is that we can stop any time we like, for example after k columns have been constructed.

So, we set

$$A[q_1, q_2, \ldots, q_k] = [q_1, q_2, \ldots, q_{k+1}] \cdot \begin{pmatrix} h_{11} & \cdots & \cdots & \cdots & h_{1k} \\ h_{21} & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & h_{k,k-1} & h_{kk} \\ 0 & \ldots & 0 & 0 & h_{k+1,k} \end{pmatrix}.$$

From this it follows that for a given j,

$$Aq_j = h_{1j}q_1 + h_{2j}q_2 + \cdots + h_{j+1,j}q_{j+1} = \sum_{i=1}^{j+1} h_{ij}q_i.$$

(If you are not convinced, it may be a good idea to try a small 3×3 example and write out everything explicitly.)

We now use orthogonality: we know that for a given $1 \leq m \leq j$ we have that $q_m^T q_j = 0$ if $m \neq j$ and $q_m^T q_j = 1$ if $m = j$. Thus, we get

$$q_m^T A q_j = \sum_{i=1}^{j+1} h_{ij} q_m^T q_i = h_{mj}.$$

For $h_{j+1,j}$ we get

$$h_{j+1,j}q_{j+1} = Aq_j - \sum_{i=1}^{j} h_{ij}q_i.$$

Taking norms and using the fact that $\|q_{j+1}\|_2 = 1$, we get

$$h_{j+1,j} = \|Aq_j - \sum_{i=1}^{j} h_{ij}q_i\|_2.$$

We have just outlined the celebrated Arnoldi algorithm for computing the orthogonal basis for $\mathcal{K}^k(A;b)$ and obtaining the upper Hessenberg matrix associated with it:

$$AQ_k = Q_{k+1}H_{k+1,k},$$

where Q_{k+1} is the matrix containing the first $k+1$ columns of the orthogonal basis for the Krylov subspace, Q_k is the same matrix, but with the first k columns, and $H_{k+1,k}$ is upper Hessenberg of size $(k+1) \times k$. It is easy to show that

$$Q_k^T AQ_k = H_{k,k}.$$

This will come handy soon when we derive solution methods.
The Arnoldi method amounts to applying the Gram-Schmidt procedure for constructing an orthogonal basis for the Krylov subspace method. The algorithm is:

$q_1 = b/\|b\|_2$
for j=1 to k
    $z = Aq_j$
    for $i = 1$ to j
        $h_{i,j} = q_i^T z$
        $z = z - h_{i,j}q_i$
    end for
    $h_{j+1,j} = \|z\|_2$
    if $h_{j+1,j} = 0$, quit
    $q_{j+1} = z/h_{j+1,j}$
end for

The Lanczos method

When A is symmetric, $A = A^T$ implies that the upper Hessenberg matrix must be symmetric. Therefore, it must be tridiagonal; let us denote it by $T_{k+1,k}$. In this case the Arnoldi method reduces to the well known *Lancsoz* method. The way to derive the method is the same as for Arnoldi, but the fact that we now have a tridiagonal matrix simplifies the computations. In particular, instead of a loop over $j+1$ elements when we deal with the j-th column, we have to deal with three terms only.

We proceed as follows:

$$A[q_1, q_2, \ldots, q_k] = [q_1, q_2, \ldots, q_k] \cdot \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_{k-1} \\ 0 & \ldots & 0 & \beta_{k-1} & \alpha_k \end{pmatrix}.$$

From this it follows that for a given j,

$$Aq_j = \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1}.$$

Just like we did for the Arnoldi algorithm, we now use orthogonality and it readily follows that

$$q_j^T A q_j = \alpha_j.$$

When we deal with the jth column, β_{j-1} is already known, and so for computing β_j we use

$$\beta_j q_{j+1} = Aq_j - \beta_{j-1}q_{j-1} - \alpha_j q_j,$$

with α_j that we just computed. Taking norms we get

$$\beta_j = \|Aq_j - \beta_{j-1}q_{j-1} - \alpha_j q_j\|_2.$$

Once β_j has been computed, q_{j+1} is readily available. We have just outlined the Lanczos algorithm for computing the orthogonal basis for $\mathcal{K}^k(A; b)$ when A is symmetric. The matrix associated with the construction is tridiagonal:

$$AQ_k = Q_{k+1}T_{k+1,k}.$$

The algorithm for the Lanczos method is:

$q_1 = b/\|b\|_2, \beta_0 = 0, q_0 = 0$
for j=1 to k
  $z = Aq_j$
  $\alpha_j = q_j^T z$
  $z = z - \alpha_j q_j - \beta_{j-1}q_{j-1}$
  $\beta_j = \|z\|_2$
  if $\beta_j = 0$, quit
  $q_{j+1} = z/\beta_j$
end for

3.2 Optimality conditions

Now that the orthogonal basis construction is taken care of, we turn to discuss the optimality criterion that we would like our iterates to satisfy. There are a few possibilities here; let us list two.

1. Require that the norm of the residual $\|b - Ax_k\|_2$ is minimal over the Krylov subspace.

2. Require that the residual is orthogonal to the subspace.

4 Minimization of Residual Norm

By Arnoldi, we have that $AQ_k = Q_{k+1}H_{k+1,k}$, and we have $x_k = Q_k z$. We will want to find the solution to

$$\min_x \|b - Ax_k\|_2.$$

We have

$$\|b - Ax_k\|_2 = \|b - AQ_k z\|_2 = \|b - Q_{k+1}H_{k+1,k}z\|_2.$$

Now, since Q_{k+1} is orthogonal, we can multiply by Q_{k+1}^T inside the norm and obtain

$$\|b - Ax_k\|_2 = \|Q_{k+1}^T b - H_{k+1,k}z\|_2.$$

The reason we can do this, is because multiplying by an orthogonal matrix preserves the Euclidean norm; indeed for any vector x, if $Q^TQ = I$ then

$$\|Qx\|_2 = \sqrt{(Qx)^TQx} = \sqrt{x^TQ^TQx} = \sqrt{x^Tx} = \|x\|_2.$$

We also note that $Q_{k+1}^T b$ is nothing but the vector $\|b\|_2 e_1$, where e_1 is the first standard basis vector. This is because in our Arnoldi procedure the first vector is just $q_1 = b/\|b\|_2$ and all the other columns of Q_{k+1} are orthogonal to q_1, and hence to b. (Please convince yourself that you understand this argument!)

So, what we have is that the minimization problem $\min \|b - Ax_k\|_2$ is equivalent to $\min \|\rho e_1 - H_{k+1,k}z\|_2$, where $\rho = \|b\|_2 \equiv \|r_0\|_2$. (Remember that we assume that $x_0 = 0$ and thus $b = r_0$.)

Next, we consider the question how to solve the least squares problem. The standard way is via the QR factorization. Let

$$H_{k+1,k} = U_{k+1,k} R_{k,k}$$

be a QR factorization of the upper Hessenberg matrix. Then

$$\|\rho e_1 - H_{k+1,k} z\|_2 = \|\rho e_1 - U_{k+1,k} R_{k,k} z\|_2 = \|\rho U_{k+1,k}^T e_1 - R_{k,k} z\|_2.$$

We therefore have

$$z = R_{k,k}^{-1} U_{k+1,k}^T \|b\|_2 e_1.$$

Once we have z, we compute the kth iterate simply by setting $x_k = Q_k z$.

The procedure we have outlined above is nothing but the well-known Generalized Minimum Norm Residual method, also known as GMRES.

5 Orthogonality of Residual (Ritz-Galerkin)

By Arnoldi, we have that $AQ_k = Q_{k+1} H_{k+1,k}$. Requiring that the residual be orthogonal to the Krylov subspace amounts to

$$Q_k^T (b - A x_k) = 0.$$

If $x_k = Q_k z$ then

$$Q_k^T b = Q_k^T A x_k = Q_k^T A Q_k z = H_{k,k} z.$$

Since $Q_k^T b = \rho e_1$ with $\rho = \|b\|_2$, we have that

$$z = H_{k,k}^{-1} \rho e_1.$$

The algorithm is thus basically based on solving $H_{k,k} z = \|r_0\|_2 e_1$, followed by forming $x_k = Q_k z$.

If A is nonsymmetric, the above method is called FOM: Full Orthogonalization method. If A is symmetric, then this is the Lanczos method. If A is symmetric positive definite, then this is basically the conjugate gradient (CG) method. CG is in fact based on some further manipulations, in particular an implicit construction of the tridiagonal matrix associated with the Lanczos procedure.

References