Lecture Note #8 **MTH652: Advanced Numerical Analysis**
Winter 2025 Instructor: Will Pazner

Krylov Methods Continued

1 GMRES

Last time we looked at the **generalized minimum residual** (GMRES) method, which is an iterative Krylov method for approximating the solution

$$A\boldsymbol{x} = \boldsymbol{b}$$

for any invertible $A \in \mathbb{R}^{N\times N}$ by selecting at each iteration the approximate solution $\boldsymbol{x}_m$ satisfying

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\| = \inf_{\boldsymbol{y}\in\mathcal{K}_m(A,\boldsymbol{b})} \|\boldsymbol{b} - A\boldsymbol{y}\|.$$

Recall that $\mathcal{K}_m(A,\boldsymbol{b})$ is the Krylov subspace of $\mathbb{R}^N$ given by

$$\mathcal{K}_m(A,\boldsymbol{b}) := \text{span}\left\{\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, \ldots, A^{m-1}\boldsymbol{b}\right\}.$$

This basis is **ill-conditioned** because the vectors $A^j\boldsymbol{b}$ become almost parallel to the dominant eigenvector of A (and hence almost parallel to each other), and so the vectors are very close to being linearly dependent.

To address this issue, we used the **Arnoldi iteration** to build up an **orthonormal basis** $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m$ for the space $\mathcal{K}_m$. Let Q_m denote the $N \times m$ matrix whose columns are the first m of these orthonormal vectors. The Arnoldi method gives the **key identity**

$$AQ_m = Q_{m+1}H_m,$$

where H_m is the $(m+1)\times m$ matrix

$$H_m = \begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1m} \\ H_{21} & H_{22} & \cdots & H_{2m} \\ 0 & H_{32} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_{m+1,m} \end{pmatrix},$$

where the coefficients H_{ij} are express $A\boldsymbol{q}_j$ in terms of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{j+1}$,

$$A\boldsymbol{q}_j = H_{1j}\boldsymbol{q}_1 + H_{2j}\boldsymbol{q}_2 + \cdots + H_{jj}\boldsymbol{q}_m + H_{j+1,j}\boldsymbol{q}_{j+1}.$$

This identity means the solution to the least-squares problem

$$\|\boldsymbol{b} - A\boldsymbol{x}_m\| \to \min, \quad \boldsymbol{y} \in \mathcal{K}_m(A,\boldsymbol{b})$$

is given by $\boldsymbol{x}_m = Q_m \boldsymbol{y}_m$, where $\boldsymbol{y}_m$ is the solution to the least-squares problem

$$\left\| \|\boldsymbol{b}\| \boldsymbol{e}_1 - H_m \boldsymbol{y}_m \right\| \to \min .$$

This is a $(m+1) \times m$ least-squares problem with the **upper Hessenberg** matrix H (a matrix is called upper Hessenberg if $H_{ij} = 0$ for all $i > j+1$).

Step m of GMRES can be summarized as follows:

1. Given $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{m-1}$, compute the next orthonormal basis vector $\boldsymbol{q}_m$ using one Arnoldi iteration
   - This requires one matrix-vector product with A, and $m-1$ dot products to compute the coefficients H_{im}.
   - The coefficients H_{im} are appended as a new column to H_{m-1} to obtain the Hessenberg matrix H_m.
2. Compute the least-squares solution to $\left\| \|\boldsymbol{b}\| \boldsymbol{e}_1 - H_m \boldsymbol{y}_m \right\| \to \min$.
3. If the residual is small enough (below a user-specified tolerance), the iteration terminates (with solution $\boldsymbol{x}_m = Q_m \boldsymbol{y}_m$), otherwise continue to the next iteration.

There are a number of questions about this algorithm that we should address:

(Q1) Is the Arnoldi iteration numerically stable? (Recall that Gram–Schmidt can suffer from numerical instability, with round-off errors causing the vectors not to be orthonormal).

(Q2) Is there a good algorithm for solving the least-squares problem with an upper Hessenberg matrix?

(Q3) What is the computational complexity of the algorithm in terms of N and number of iterations n_{it}?

(Q4) How does the convergence behavior depend on the matrix A?

(Q5) Can convergence be accelerated using a preconditioner?

In what follows, we will give (sometimes partial) answers to the first three questions. Questions (Q4) and (Q5) are more involved, and the theory is often problem-specific. We mention only that for general matrices, the convergence is not bounded in a simple way in terms of the extremal eigenvalues of the matrix (like it was for the conjugate gradient method). This means that (even when the spectral of A is known), it can be difficult to predict the convergence behavior of GMRES. Preconditioners $M \approx A^{-1}$ can be used, applying GMRES either to $M^{-1}A$ or to AM^{-1}. Often M is taken to be an **incomplete factorization** of A (i.e. an approximation to A^{-1} obtained by modifying the LU algorithm, e.g. by dropping fill-in entries). These two options are called "left preconditioning" and "right preconditioning".

1.1 (Q1) Householder Algorithm

The Arnoldi method with standard Gram–Schmidt will not be numerically stable. A simple modification to the algorithm (using *modified* Gram–Schmidt) improves the stability significantly.

Algorithm 1 Arnoldi Iteration

$\boldsymbol{q}_1 \leftarrow \boldsymbol{u}/\|\boldsymbol{u}\|$
for $k = 1, 2, \ldots, m-1$ **do**
  $\boldsymbol{z} \leftarrow A\boldsymbol{q}_k$
  for $i = 1, \ldots, k$ **do**
    $H_{ik} \leftarrow \boldsymbol{q}_i^T \boldsymbol{z}$
  end for
  $\boldsymbol{z} \leftarrow \boldsymbol{z} - H_{1k}\boldsymbol{q}_1 - \cdots - H_{kk}\boldsymbol{q}_k$
  $H_{k+1,k} \leftarrow \|\boldsymbol{z}\|$
  $\boldsymbol{q}_{k+1} \leftarrow \boldsymbol{z}/H_{k+1,k}$
end for

Algorithm 2 Modified Arnoldi Iteration

$\boldsymbol{q}_1 \leftarrow \boldsymbol{u}/\|\boldsymbol{u}\|$
for $k = 1, 2, \ldots, m-1$ **do**
  $\boldsymbol{z} \leftarrow A\boldsymbol{q}_k$
  for $i = 1, \ldots, k$ **do**
    $H_{ik} \leftarrow \boldsymbol{q}_i^T \boldsymbol{z}$
    $\boldsymbol{z} \leftarrow \boldsymbol{z} - H_{ik}\boldsymbol{q}_i$
  end for
  $H_{k+1,k} \leftarrow \|\boldsymbol{z}\|$
  $\boldsymbol{q}_{k+1} \leftarrow \boldsymbol{z}/H_{k+1,k}$
end for

In exact arithmetic, since $\boldsymbol{q}_i$ are all orthogonal to each other, these two versions of the algorithm are identical. However, in finite precision, the vectors are only approximately orthogonal, and modifying $\boldsymbol{z}$ within the inner for-loop improves the accuracy.

An alternative approach to orthogonalization is the **Householder algorithm** (note: for those interested in opportunities at the national labs, there is a prestigious "Householder postdoctoral fellowship" at Oak Ridge National Laboratory). This algorithm makes use of **Householder reflectors**, which are matrices of the form

$$P = I - 2\boldsymbol{w}\boldsymbol{w}^T,$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is any unit vector ($\|\boldsymbol{w}\| = 1$). The matrix P represents **reflection** about the plane normal to $\boldsymbol{w}$ passing through the origin. Note that $P^T = P$ and $P^{-1} = P^T = P$ (exercise). This means that P is an **orthogonal matrix**, and so the product of multiple Householder reflectors is also orthogonal.

Finding an orthonormal basis for the space spanned by the columns of an $N \times n$ matrix K is equivalent to computing the QR factorization,

$$K = QR,$$

where R is upper-triangular, and Q is orthogonal. We proceed column-by-column,

$$\begin{aligned} K_1 &= P_1 K \\ K_2 &= P_2 K_1 = P_2 P_1 K \\ K_3 &= P_3 K_2 = P_3 P_2 P_1 K \\ &\vdots \\ K_i &= P_i K_{i-1} \\ &\vdots \\ R &= K_m = P_m \cdots P_1 K \end{aligned}$$

where our goal is to make each matrix K_i upper-triangular up to column i. This means that $R = K_m$ will be upper triangular; we can then set the matrix $Q = (P_m \cdots P_1)^T$, which is orthogonal, and we have obtained $K = QR$ as desired. At each step, we want to find a Householder matrix $P_i = I - 2\boldsymbol{w}_i \boldsymbol{w}_i^T$ that leaves the first $i-1$ rows of K_{i-1} unchanged, but zeros out all the rows $j > i$ in column i.

To leave the first $i-1$ rows (and columns) of K_{i-1} unchanged, the vector $\boldsymbol{w}_i$ can be chosen to have zeros in the first $i-1$ places, i.e.

$$\boldsymbol{w}_i = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \boldsymbol{w}_i' \end{pmatrix}$$

where $\boldsymbol{w}_i$ is a vector of length $N - i + 1$. Let $\boldsymbol{x}_i$ denote the last $N - i + 1$ entries of the ith column of K. The problem has been reduced to determining $\boldsymbol{w}_i'$ such that

$$(I - 2\boldsymbol{w}_i'(\boldsymbol{w}_i')^T)\boldsymbol{x}_i = \alpha \boldsymbol{e}_1,$$

for some $\alpha \neq 0$, where $\boldsymbol{e}_1$ is the standard basis vector. To simplify notation, we look for $\boldsymbol{w}$ such that

$$P\boldsymbol{x} = (I - \boldsymbol{w}\boldsymbol{w}^T)\boldsymbol{x} = \alpha \boldsymbol{e}_1$$

This yields

$$2(\boldsymbol{w}^T \boldsymbol{x})\boldsymbol{w} = \boldsymbol{x} - \alpha \boldsymbol{e}_1. \tag{1}$$

This means that $\boldsymbol{w}$ is a scalar multiple of $\boldsymbol{x} - \alpha \boldsymbol{e}_1$, and since $\boldsymbol{w}$ is unit-length, we have

$$\boldsymbol{w} = \pm \frac{\boldsymbol{x} - \alpha \boldsymbol{e}_1}{\|\boldsymbol{x} - \alpha \boldsymbol{e}_1\|}.$$

We are left to determine the unknown scalar α. Using this in (1), we obtain

$$2\left(\pm \frac{(\boldsymbol{x} - \alpha \boldsymbol{e}_1)^T}{\|\boldsymbol{x} - \alpha \boldsymbol{e}_1\|}\boldsymbol{x}\right)\boldsymbol{w} = \pm \|\boldsymbol{x} - \alpha \boldsymbol{e}_1\| \boldsymbol{w},$$

giving

$$2(\boldsymbol{x} - \alpha \boldsymbol{e}_1)^T \boldsymbol{x} \boldsymbol{w} = \pm \|\boldsymbol{x} - \alpha \boldsymbol{e}_1\|^2 \boldsymbol{w},$$

and so we can impose the condition

$$2(\boldsymbol{x} - \alpha \boldsymbol{e}_1)^T \boldsymbol{x} = \|\boldsymbol{x} - \alpha \boldsymbol{e}_1\|^2.$$

This expands to

$$2(\|\boldsymbol{x}\|_2 - \alpha x_1) = \|\boldsymbol{x}\|^2 - 2\alpha x_1 + \alpha^2$$

(where x_1 is the first component of $\boldsymbol{x}$). From this, it follows that $\alpha^2 = \|\boldsymbol{x}\|^2$, or

$$\alpha = \pm \|\boldsymbol{x}\|.$$

Typically, we choose

$$\alpha = -\operatorname{sign}(x_1)\|\boldsymbol{x}\|,$$

which prevents cancellation (which could lead to small $\boldsymbol{w}$). This gives the expression for $\boldsymbol{w}$,

$$\boldsymbol{w} = \frac{\boldsymbol{x} + \operatorname{sign}(x_1)\|\boldsymbol{x}\|\boldsymbol{e}_1}{\|\boldsymbol{x} + \operatorname{sign}(x_1)\|\boldsymbol{x}\|\boldsymbol{e}_1\|}.$$

Note that the matrices $P = I - 2\boldsymbol{w}\boldsymbol{w}^T$ do not need to be formed explicitly, since they are rank-1 updates of the identity matrix. To multiply PK for any K, we have $PK = K - 2\boldsymbol{w}\boldsymbol{w}^T$, which requires only matrix–vector operations. This procedure can replace Gram–Schmidt in the Arnoldi iteration to obtain a more numerically stable algorithm.

1.2 (Q2) Least-squares solutions using Givens rotations

To obtain the approximate solution $\boldsymbol{x}_m$ at step m of GMRES, a least-squares problem needs to be solved. Note that the solution to this problem is not required for the next step of the algorithm, so it actually does not need to performed every iteration. However, because of the special structure of H_m (it is upper Hessenberg), and because H_{m+1} is obtained from H_m by adding a column, we can solve these least-squares very efficiently. A sequence of small 2×2 rotation matrices ("Givens rotations"; there is a Givens postdoctoral fellowship at Argonne national laboratory) are applied to the matrix H_m to transform it to upper-triangular form. Since H_{m+1} is obtained from H_m by appending a column, these rotations can be applied in a progressive manner, where each additional step is very inexpensive. (Details are omitted here; the curious reader is directed to Saad's 2003 textbook for details).

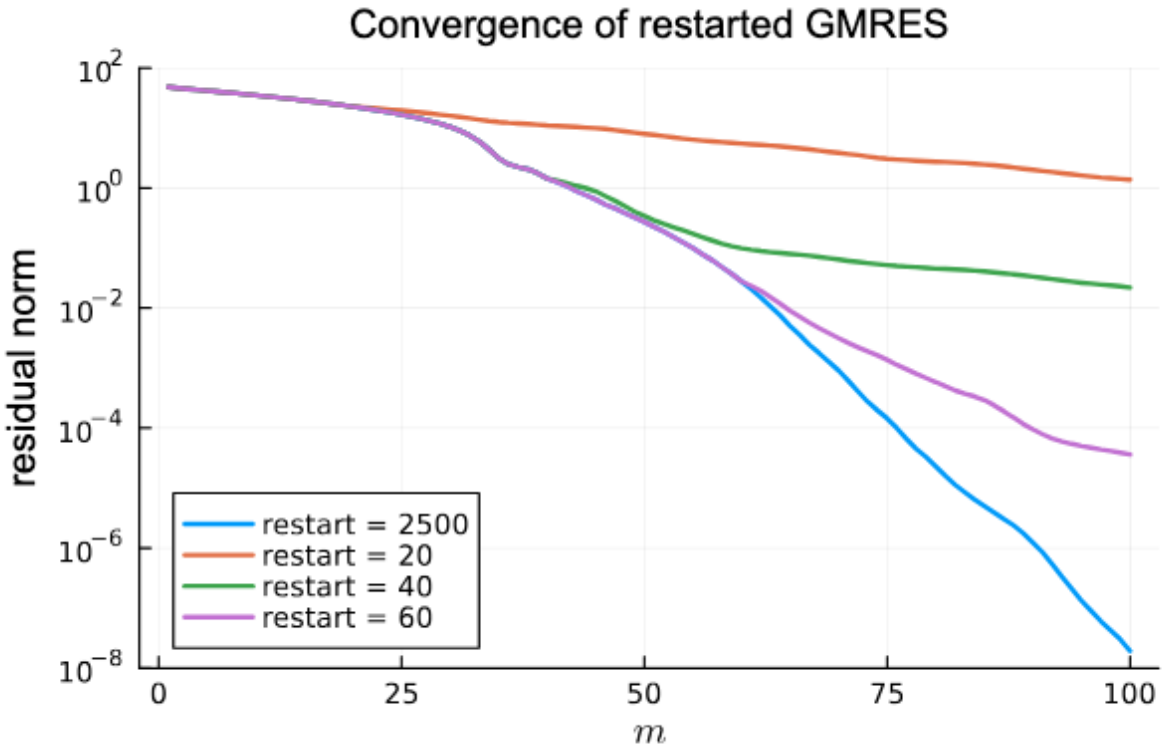
1.3 (Q3) Computational complexity and memory requirements

Unlike the conjugate gradient, where fortunate cancellations made Gram–Schmidt orthogonalization very inexpensive (and rendered it unnecessary to store previous search directions), the basis for the Krylov subspace (usually called the "Krylov vectors") need to be stored in GMRES. This means that at step m, we need to store m vectors of length N. The process of orthogonalizing the vectors (using either Gram–Schmidt or Householder) is quadratic in m and linear in N, so $\mathcal{O}(m^2 N)$ operations are required. If we assume that nnz $\sim N$ (as

in the case of finite element matrices), then for n_{it} iterations, the memory requirements are $\mathcal{O}(n_{\text{it}}N)$ and the number of operations is $\mathcal{O}(n_{\text{it}}^2 N)$. GMRES is guaranteed to converge in $n_{\text{it}} \leq N$ iterations, so the worst case is $\mathcal{O}(N^2)$ memory and $\mathcal{O}(N^3)$ operations, which is equivalent to treating A as a dense matrix and using a direct method(!).

The goal when using iterative methods is to have $n_{\text{it}} \ll N$ (and ideally $n_{\text{it}} = \mathcal{O}(1)$), in which case the memory and operations are both $\mathcal{O}(N)$, which is optimal. Whether it is possible to obtain $\mathcal{O}(1)$ iterations is problem-dependent, and requires effective preconditioners.

Another option is to **restart** GMRES after $M = \mathcal{O}(1)$ iterations. Here, M is a user-defined parameter; theory for picking M is available in very special cases. After M iterations, the previous Krylov vectors are thrown away, and the Arnoldi process is started again. This means that the memory requirements are $\mathcal{O}(MN)$, and the computational complexity is $\mathcal{O}(n_{\text{it}}MN)$. Unfortunately, restarting GMRES generally makes the convergence worse, and so n_{it} will be larger. This means that M can be tuned to trade off between the number of operations and memory usage. The following figure (from the book "Fundamentals of Numerical Computation" by Driscoll and Braun) illustrates what can happen with different restart parameters.



2 MINRES

Note that if the matrix A is symmetric, then we can take the Arnoldi identity

$$AQ_m = Q_{m+1}H_m,$$

and multiply on the left by Q_m^T to obtain

$$Q_m^T A Q_m = Q_m^T Q_{m+1} H_m =: H'_m.$$

Since A is symmetric, the left-hand side is symmetric, and so H'_m is also symmetric. By orthogonality of Q_m, we see that H'_m is upper Hessenberg (it is the first m rows of H_m). Since it is also symmetric, it must be tridiagonal, leading to the simplified relationship

$$A\boldsymbol{q}_m = H_{m-1,m}\boldsymbol{q}_{m-1} + H_{mm}\boldsymbol{q}_m + H_{m+1,m}\boldsymbol{q}_{m+1}.$$

This is called the **Lanczos iteration**. The resulting method is called **MINRES**, and gives rise to a short-term recurrence; only the past two Krylov vectors need to be stored, so restarting is not required.