

GMRES: a Krylov method for linear systems

We know how to compute a basis of the Krylov subspace, now let's use it to solve a linear system.

Idea: look for best approximate solution inside $K_n(A, \mathbf{y})$:

$$\min \|A\mathbf{x} - \mathbf{y}\|, \text{ over all } \mathbf{x} \in K_n(A, \mathbf{y}), \text{ i.e., } \mathbf{x} = Q_n\mathbf{z}.$$

That is, $\min_{\mathbf{z} \in \mathbb{R}^n} \|AQ_n\mathbf{z} - \mathbf{y}\|$: we find \mathbf{z} by solving a least-squares problem.

Even better, we can reduce it to a smaller-size problem:

$$\begin{aligned} \|AQ_n\mathbf{z} - \mathbf{y}\| &= \|Q_{n+1}\underline{H}_n\mathbf{z} - \mathbf{y}\| = \left\| [Q_{n+1} \quad Q_c]^T (Q_{n+1}\underline{H}_n\mathbf{z} - \mathbf{y}) \right\| \\ &= \left\| \begin{bmatrix} \underline{H}_n\mathbf{z} - \mathbf{e}_1\|\mathbf{y}\| \\ 0 \end{bmatrix} \right\| = \|\underline{H}_n\mathbf{z} - \mathbf{e}_1\|\mathbf{y}\|. \end{aligned}$$

A LS problem of size $(n+1) \times n$.

Implementation

```
>> [Q, Hn] = arnoldi(A, y, n);  
>> v = eye(n+1, 1) * norm(y);  
>> z = H(:, 1:n) \ v;  
>> x = Q(:, 1:n) * z;
```

- ▶ The residual norm $\|A\mathbf{x}_n - \mathbf{y}\|$ can be computed for free (without another product with A) from the $(n+1) \times n$ LS.
- ▶ $\text{qr}(H_n)$ can be computed in $O(n^2)$ using the fact that H_n already has many zeros below the diagonal. (Details omitted; not such a big deal anyway because the most expensive part of the algorithm is the Arnoldi iteration.)
- ▶ One can merge this computation with the Arnoldi loop: we 'update' the QR of H_k to that of H_{k+1} after each step. This avoids the need to choose n in advance: we compute $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots$ and stop when $\|A\mathbf{x}_n - \mathbf{y}\|$ is small enough.

We won't see an implementation with all these details.

Matlab: `gmres(A, b)`, Python: `scipy.sparse.linalg.gmres`.

Example

```
m = 1000;
A = 10*speye(m) + sprandn(m, m, 0.01);
y = randn(size(A,1), 1);
[Q, H] = arnoldi(A, y, 50);
r = nan(50,1);
% tests several 'slices' of Q,H
% this simulates Arnoldi with various values of n.
for n = 1:50
    z = H(1:n+1, 1:n) \ eye(n+1, 1) * norm(y);
    x = Q(:, 1:n) * z;
    r(n) = norm(A*x - y);
end
semilogy(r)
```

Observe: changing that factor 10 affects convergence speed!

Lucky breakdown

An interesting phenomenon: when Arnoldi breaks down, we can obtain the **exact solution** to the linear system. Indeed, if $h_{n+1,n} = 0$, then in the $(n+1) \times n$ least-squares problem

$$\min \| \underline{H}_n \mathbf{z} - \mathbf{e}_1 \| \| \mathbf{y} \|$$

the last equation is $0 = 0$.

The remaining equations form a **square linear system** $H_n \mathbf{z} = \mathbf{e}_1 \| \mathbf{y} \|$. We can reach residual 0 \implies the linear system $A\mathbf{x} = \mathbf{y}$ is solved **exactly** at step n .

(Tricky question: why must H_n be invertible?)

GMRES convergence

We can mimic the convergence proof of CG.

$\mathbf{x} \in K_n(A, \mathbf{y}) \iff \mathbf{x} = p(A)\mathbf{y}$ for a polynomial $p(t)$ of degree $< n$.

$$\min_{\substack{\mathbf{x}=p(A)\mathbf{y} \\ p \text{ of degree } < n}} \|\mathbf{Ax} - \mathbf{y}\| = \min_{p \text{ of degree } < n} \|(Ap(A) - I)\mathbf{y}\|.$$

GMRES finds the best polynomial for us!

If $A = V\Lambda V^{-1}$ diagonalizable, then

$$Ap(A) - I = V \begin{bmatrix} \lambda_1 p(\lambda_1) - 1 & & \\ & \ddots & \\ & & \lambda_m p(\lambda_m) - 1 \end{bmatrix} V^{-1}.$$

- ▶ If A has very few distinct eigenvalues ($k \leq n$ of them), then we can find p such that $p(\lambda_i) = \frac{1}{\lambda_i}$ for all i : **interpolation!**
- ▶ If A has few 'clusters' of eigenvalues, we can find p such that $\lambda_i p(\lambda_i) - 1$ is small for all i .

GMRES convergence

Passing to norms,

$$\|\mathbf{r}_n\| = \min_{\mathbf{x} \in K_n(A, \mathbf{y})} \|\mathbf{A}\mathbf{x} - \mathbf{y}\| \leq \kappa(V) \min_{\rho(t)} \max_{\lambda_i} |\lambda_i \rho(\lambda_i) - 1| \|\mathbf{y}\|.$$

- ▶ If A has only n **distinct** eigenvalues, GMRES converges in n steps.
- ▶ (informally) If A has a few 'clusters' of eigenvalues well separated from 0, then GMRES converges fast.

Example Repeat the previous experiment with $A = \text{bucky}()$.

Wrap-up

GMRES computes the vector \mathbf{x} that minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{y}\|$ among all vectors in $K_n(\mathbf{A}, \mathbf{y})$.

It **provably** gives lower residual $\|\mathbf{A}\mathbf{x}_n - \mathbf{y}\|$ than any other algorithm that produces iterates in $K_n(\mathbf{A}, \mathbf{y})$, e.g.,

```
function x = secret_accelerated_descent(A, y)
m = length(y);
x = zeros(m, 1);
for k = 1:n
    w = some_combination_of_previous_iterates_and_rs(...);
    r = A*w - w;
    x = some_combination_of_previous_iterates_and_rs(...);
end
```

(Similarly, any algorithm that makes 2 products with A per step is no better than $2n$ steps of GMRES.)

[Book references](#) Trefethen–Bau, Lecture 35; Demmel, Section 6.6.6 (in part).

Lanczos = symmetric Arnoldi

If A is symmetric ($A = A^T$), then H_n is symmetric for each n and hence it is **tridiagonal**.

```
A = bucky(); y = randn(size(A, 1), 1);  
[Q, H] = arnoldi(bucky, y, 40);  
spy(H);
```

Indeed, $H_{ji} = \mathbf{q}_j^T A \mathbf{q}_i = (\mathbf{q}_j^T A \mathbf{q}_i)^T = \mathbf{q}_i^T A \mathbf{q}_j = H_{ij}$.

So we can shorten the orthogonalization loop:

```
w = A * Q(:, j);  
for i = j-1:j % only 2 vectors to check  
    %and actually we already know H(j-1,j)=H(j,j-1)  
    H(i,j) = Q(:, i)' * w;  
    w = w - H(i,j) * Q(:, i);  
end
```

This reduces the cost to n matrix products + $O(mn)$.

Uses for Lanczos

This 'symmetric Arnoldi' is known as **Lanczos iteration**.

Symmetry reduces the cost not only of the orthogonalization loop, but also of the small-scale least-squares systems: \underline{H}_n is tridiagonal.

The resulting 'symmetric GMRES' is called **MINRES**.

Final notes

Which Krylov method to use for $Ax = y$

- ▶ Is A very small, or completely without sparsity/structure? Probably you should be using a direct method instead. . .
- ▶ Is A posdef? Use conjugate gradient.
- ▶ Is A symmetric? Use MINRES (=symmetric GMRES).
- ▶ None of the above? Use GMRES.

Book refs Trefethen–Bau, Lecture 38; Demmel, Sections 6.6.3, 6.6.4.

Practical warning often, when computing with Krylov methods numerically, exact orthogonality is lost after a few iterations: the value of $\mathbf{q}_i^T \mathbf{q}_j$ slowly grows (by accumulation of errors, starting from machine precision) as i, j become more far apart.

Exercises

- ▶ Show that, if we have lucky breakdown at step n , then the matrix H_n is invertible whenever A is invertible. (Hint: use $AQ_n = Q_nH_n$. What happens if $H_n\mathbf{v} = 0$ for a vector $\mathbf{v} \neq \mathbf{0}$?)