GMRES: a Krylov method for linear systems

Goal: solving a large-scale linear system $Ax = b$, $A \in \mathbb{R}^{m\times m}$.

Idea we look for 'the closest thing to solution' inside $K_n(A, b)$.

$$\min\|Ax - b\|, \quad x = Q_n y.$$

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

I.e., $\min_{y\in\mathbb{R}^n}\|AQ_n y - b\|$: a least-squares problem.

Can be reduced to a smaller-size problem:

$$\|AQ_n y - b\| = \|Q_{n+1}\underline{H}_n y - b\| = \left\|\begin{bmatrix} Q_{n+1} & \widehat{Q}\end{bmatrix}^T (Q_{n+1}\underline{H}_n y - b)\right\|$$

$$= \left\|\begin{bmatrix} \underline{H}_n y - \|b\|e_1 \\ \\ 0 \end{bmatrix}\right\| = \left\|\underline{H}_n y - \|b\|e_1\right\|.$$

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

$$Q_n = \left[q_1 \middle| q_2 \middle| \cdots q_n \right] \in \mathbb{R}^{m \times n} \qquad Q_n^T Q_n = I$$

$$Q_n y = q_1 y_1 + q_2 y_2 + \cdots + q_n y_n \qquad y \in \mathbb{R}^n$$

Now looking for the vector $Q_n y \in K_n(A, b)$

s.t. $\|AQ_n y - b\|$ is minimized.

$\downarrow$

$AQ_n \in \mathbb{R}^{m \times n}$, tall thin matrix

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

Arnoldi produces

$q_1, q_2 \cdots q_n, q_{n+1}$ and $\underline{H_n} = \begin{bmatrix} * & \cdots & * \\ * & * & \vdots \\ & * & * \\ 0 & & * \end{bmatrix} \in \mathbb{R}^{(n+1)\times n}$

s.t. $\underline{AQ_n} = Q_{n+1}\underline{H_n}$

Problem we're now solving: minimize $\|AQ_n y - b\| = \|Q_{n+1}\underline{H_n} y - b\| = \left\| \begin{bmatrix} Q_{n+1} & \hat{Q} \end{bmatrix}^T \left(Q_{n+1}\underline{H_n} y - b \right) \right\| =$

($\begin{bmatrix} Q_{n+1} & \hat{Q} \end{bmatrix}$ any orthon. matrix with Q_{n+1} as its first columns)

$$\begin{bmatrix} Q_{n+1} & \hat{Q} \end{bmatrix}^T \begin{bmatrix} Q_{n+1} & \hat{Q} \end{bmatrix} = \underline{I} \Rightarrow \begin{matrix} Q_{n+1}^T Q_{n+1} = I \\ \hat{Q}^T Q_{n+1} = 0 \end{matrix}$$

$$= \left\| \begin{bmatrix} Q_{n+1}^T \\ \hat{Q}^T \end{bmatrix} \left(Q_{n+1} \underline{H}_n y - b \right) \right\| = \left\| \begin{bmatrix} Q_{n+1}^T \\ \hat{Q}^T \end{bmatrix} Q_{n+1} \underline{H}_n y - \begin{bmatrix} Q_{n+1}^T \\ \hat{Q}^T \end{bmatrix} b \right\| =$$

$$= \left\| \begin{bmatrix} I \cdot \underline{H}_n y \\ \hline 0 \end{bmatrix} - \begin{bmatrix} \|b\| \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \right\| = \left\| \begin{bmatrix} \underline{H}_n y - \|b\| e_1 \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \right\| \begin{matrix} \} n \\ \} m-n \end{matrix}$$

minimized if y solves $\min \| \underline{H}_n y - \|b\| e_1 \|$

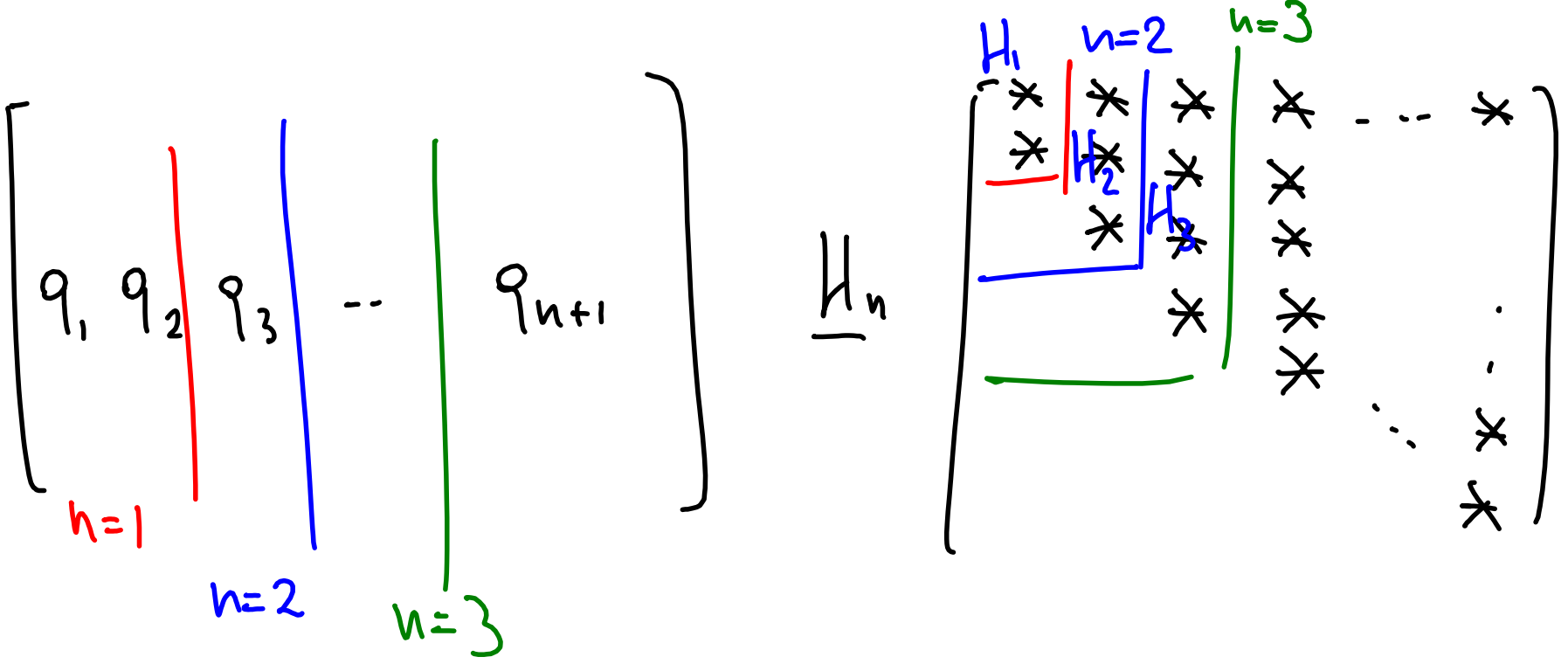
$\underline{H}_n \in \mathbb{R}^{(n+1) \times n}$, much smaller problem $\rightsquigarrow O(n^3)$

(Actually, one can construct the QR fact.n of $\underline{H}_n$ in time $O(n^2)$, by using its structure.)

Implementation details

- `qr(H)` can be computed in $O(n^2)$ using the special shape of H (Hessenberg matrix).
  Not such a big deal because Arnoldi costs more anyway.
- Instead of doing a QR at the end, we can compute QRs of $\underline{H}_1, \underline{H}_2, \ldots$ and update them at each step.
  This allows us to compute residuals $\|Ax_n - b\|$ (that we can use as stopping criterion).

We won't see an implementation with all these details.
Matlab has `gmres(A, b)` (and Python has `scipy.sparse.linalg.gmres`).



One can construct a QR fact. of H_1, then update it to one of H_2, then to one of H_3, etc....

Example

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

See how changing that factor 10 affects convergence speed.

The residual decreases monotonically:

calling x_n the approx. of the solution produced by a Kr. subspace of dim. n,

GMRES convergence

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

$$\min_{x \in K_n(A,b)} \|Ax - b\| = \min_{\substack{q(x)=1-xp(x) \\ \text{of degree} \le n}} \|q(A)b\|.$$

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

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

- If A has very few distinct eigenvalues ($k \le n$ of them), then we can find q such that $q(\lambda_i) = 0$ for all i.
- If A has few 'clusters' of eigenvalues, we can find q such that $q(\lambda_i)$ is small for all i.

Example Repeat the previous experiment with `A=bucky()`.

Exercises

-