QR Iteration

$A_0 = A$

for $k = 0, 1, 2, \ldots$

$\quad A_k = Q_k R_k$ QR decomposition of A_k

$\quad A_{k+1} = R_k Q_k$

end

Note:

- $A_{k+1} = R_k Q_k = Q_k^T A_k Q_k$

  so A_{k+1} is similar to A_k (same λ's)

- stable algorithm, since it is based on orthogonal similarity transforms.

- under certain conditions, A_k converges to Schur form of A: $A = QTQ^*$

  T triangular

The above the basic QR Algorithm for finding eigenvalues of the matrix A. In practice, this basic algorithm is further accelerated in several ways, including:

- Reduce cost per iteration: Make the QR decomposition step cheaper. For a general nxn matrix A, it is O(n^3). The matrix A can be first reduced to upper Hessenberg form (see below), so that QR will be O(n^2). If A is symmetric, the upper Hessenberg form is even better — it's tridiagonal, making QR decomposition computable in O(n) operations.
- Reduce the number of iterations: Apply carefully chosen shifts to increase the separation of the eigenvalues and thus accelerate the convergence of the method.

Upper Hessenberg form via Householder

since we are trying to preserve the eigenvalues, want similarity transform.

$$H_1 A = \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix}$$

red entries changed by mult by H_1

$H_1 A H_1^T$ doesn't work

$$H_1 A = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{pmatrix}, \quad H_1 = \begin{pmatrix} 1 & \\ & \bar{H}_1 \end{pmatrix}$$

$$(H_1 A) H_1^T = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{pmatrix}$$

$$H_2 = \begin{pmatrix} I_2 & \\ & \bar{H}_2 \end{pmatrix}, \ldots, \quad H_{n-2} = \begin{pmatrix} I_{n-2} & \\ & \bar{H}_{n-2} \end{pmatrix}$$

$$\underbrace{H_{n-2} \cdots H_2 H_1}_{Q} \, A \, \underbrace{H_1^T H_2^T \cdots H_{n-2}^T}_{Q^T} = H$$

$$Q \, A \, Q^T = H$$

$$A = Q^T H Q$$

Eigenvalues of tridiagonal T by QR iteration

$T = T_0$

$T_0 = Q_0 R_0$

$T_1 = R_0 Q_0$

$T_1 = Q_1 R_1$

$T_2 = R_1 Q_1$

$\vdots$

$$\boxed{\begin{array}{l} T_0 = T \\ \text{for } k = 0, 1, 2, \ldots \\ \quad T_k = Q_k R_k \\ \quad T_{k+1} = R_k Q_k \\ \text{end} \end{array}}$$

Note:
- T_{k+1} similar to T_k
- T_k's all tridiagonal
- T_k's converging to diagonal matrix Λ

Accelerated convergence: use shifts

$$\boxed{\begin{array}{l} \text{Choose shift } s_k \\ T_k - s_k I = Q_k R_k \\ T_{k+1} = R_k Q_k + s_k I \end{array}}$$

shifted QR

Shifted QR achieves cubic convergence

Note: we still have the condition

T_{k+1} is <u>similar</u> to T_k

$$T_k - s_k I = Q_k R_k$$

$$\Rightarrow \quad T_k = Q_k R_k + s_k I$$

$$\Rightarrow \quad R_k = Q_k^T T_k - Q_k^T s_k$$

$$\begin{aligned} T_{k+1} &= R_k Q_k + s_k I \\ &= \left(Q_k^T T_k - Q_k^T s_k\right) Q_k + s_k I \\ &= Q_k^T T_k Q_k - s_k Q_k^T Q_k + s_k I \\ &= Q_k^T T_k Q_k \quad \checkmark. \end{aligned}$$

Simultaneous Iteration

$$\underline{Q}^{(0)} = I$$
$$Z = A\,\underline{Q}^{(k-1)}$$
$$Z = \underline{Q}^{(k)} R^{(k)}$$
$$A^{(k)} = (\underline{Q}^{(k)})^T A\, \underline{Q}^{(k)}$$

Aside: Why does the QR algorithm work? It may be easier to understand why Simultaneous Iteration works, because Simultaneous Iteration directly extends the simple power method. Then, one can show a direct correspondence between the iterates in Simultaneous Iteration and the QR algorithm. (See Trefethen & Bau).

Unshifted QR Algorithm

$$A^{(0)} = A$$
$$A^{(k-1)} = Q^{(k)} R^{(k)}$$
$$A^{(k)} = R^{(k)} Q^{(k)}$$
$$\underline{Q}^{(k)} = Q^{(1)} Q^{(2)} \cdots Q^{(k)}$$

$$\underline{R}^{(k)} = R^{(k)} R^{(k-1)} \cdots R^{(1)}$$

---

* $\underline{R}^{(k)}$, $\underline{Q}^{(k)}$, and $A^{(k)}$ equivalent, and $A^k = \underline{Q}^{(k)} \underline{R}^{(k)}$, $A^{(k)} = \underline{Q}^{(k)T} A\, \underline{Q}^{(k)}$

Proof. induction in k

$\boxed{k=0}$

SI: $A^0 = \underline{Q}^{(0)} = \underline{R}^{(0)} = I$, $A^{(0)} = A$

QR: $A^0 = \underline{Q}^{(0)} = \underline{R}^{(0)} = I$ $\quad A^{(0)} = A$

$\boxed{k \geq 1}$

SI: $A^{(k)} = \underline{Q}^{(k)T} A\, \underline{Q}^{(k)}$ ✓

$$A^k = A A^{k-1} = A\, \underline{Q}^{(k-1)} \underline{R}^{(k-1)} = \underline{Q}^{(k)} R^{(k)} \underline{R}^{(k-1)}$$

$\Rightarrow$ $A^k = \underline{Q}^{(k)} \underline{R}^{(k)}$ ✓

QR: ✓ $A^k = A A^{k-1} = A\, \underline{Q}^{(k-1)} \underline{R}^{(k-1)} = \underline{Q}^{(k-1)} A^{(k-1)} \underline{R}^{(k-1)} = \underline{Q}^{(k)} \underline{R}^{(k)}$

$$A^{(k)} = Q^{(k)T} A^{(k-1)} Q^{(k)} = \underline{Q}^{(k)T} A\, \underline{Q}^{(k)} \quad ✓$$

Krylov Subspaces and Arnoldi Iteration

For very large matrices, it is not practical to compute the direct reduction to Hessenberg form above, or to do QR iteration on the full matrix A. Instead, Krylov subspace methods construct projections of A into small, Krylov subspaces. The problem is solved on successively larger Krylov subspaces to obtain approximate solutions. Krylov subspace methods are particularly appropriate when A is sparse, since they only use A to form Krylov vectors through application of A to a vector. A is treated as a black box and need not even be explicitly represented as a matrix.

Krylov vectors

$$b, Ab, A^2 b, \ldots$$

Krylov subspace

$$K_r = \text{span}\{\underbrace{b, Ab, \ldots, A^{r-1}b}_{\text{first } r \text{ Krylov vectors}}\}$$

not generally orthogonal, so use Gram-Schmidt to orthogonalize.

This is the Arnoldi Iteration.

After iteration k, we have

$$A Q_k = Q_{k+1} H_{k+1,k}$$

Multiply both sides by Q_k^T, we get

$$Q_k^T A Q_k = Q_k^T Q_{k+1} H_{k+1,k}$$

$$= \left[I_{k \times k} \ \vec{0}_{k \times 1} \right] H_{k+1,k} =$$

$$= H_k \quad (\text{first } k \text{ rows of } H_{k+1,k})$$

$$H_k = Q_k^T A Q_k$$

projection of A onto k^{th} Krylov space.

Arnoldi Iteration

$$q_1 = b / \|b\|$$

$$A q_1 \rightsquigarrow q_2$$

after iteration k, $q_1, q_2, \ldots, q_k$

$$v = A q_k$$

<u>orthogonalize</u> w.r.t. to $q_1, \ldots, q_k$

$$v \leftarrow v - \underbrace{(q_j^T v)}_{= h_{jk}} q_j \qquad j = 1, \ldots, k$$

<u>normalize</u>

$$q_{k+1} = v / \underbrace{\|v\|}_{= h_{k+1,k}}$$

$$A q_k = h_{1k} q_1 + h_{2k} q_2 + \cdots + h_{kk} q_k + h_{k+1,k} q_{k+1}$$

$$A q_k = \begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_{k+1} \\ | & | & & | \end{bmatrix} \begin{bmatrix} h_{1k} \\ h_{2k} \\ \vdots \\ h_{kk} \\ h_{k+1,k} \end{bmatrix}$$

$\cancel{A\,[}$

$$A q_1 = \begin{pmatrix} | & | \\ q_1 & q_2 \\ | & | \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{21} \end{pmatrix}$$

$$A q_2 = \begin{pmatrix} | & | & | \\ q_1 & q_2 & q_3 \\ | & | & | \end{pmatrix} \begin{pmatrix} h_{12} \\ h_{22} \\ h_{32} \end{pmatrix}$$

$$\vdots$$

$$A q_k$$

$$\begin{pmatrix} | & | & & | \\ Aq_1 & Aq_2 & \cdots & Aq_k \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_{k+1} \\ | & | & & | \end{pmatrix} \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & & h_{2k} \\ & h_{32} & \ddots & \\ & & \ddots & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix}$$

$$A Q_k = Q_{k+1} H_{k+1,k}$$

$$Q_k^T A Q_k = \underset{k \times k}{} \; \underset{k \times (k+1)}{Q_k^T Q_{k+1}} \; \underset{(k+1) \times k}{H_{k+1,k}}$$

$$= \underset{k \times (k+1)}{\begin{bmatrix} I & \vec{0} \end{bmatrix}} \; \underset{(k+1) \times k}{H_{k+1,k}}$$

$$\boxed{Q_k^T A Q_k = H_k}$$

$$q_1 = \frac{b}{\|b\|}$$

$$v = A q_1$$

$$h_{11} = q_1^T v$$

$$v \leftarrow v - h_{11} q_1$$

$$h_{21} = \|v\|$$

$$q_2 = v / h_{21}$$

Arnoldi Iteration

in each iteration k:

($q_1 = \frac{b}{\|b\|}, q_2, \cdots, q_k$ are known)

$$v = A q_k$$

for $j = 1, \dots, k$

$$h_{jk} = q_j^T v$$

$$v \leftarrow v - h_{jk} q_j$$

$$h_{k+1,k} = \|v\|$$

$$q_{k+1} = v / h_{k+1,k}$$

$$\Rightarrow A q_1 = h_{11} q_1 + h_{21} q_2$$

$$\Rightarrow A q_k = h_{1k} q_1 + \dots + h_{kk} q_k + h_{k+1,k} q_{k+1}$$

In matrix form, we are computing this factorization:

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} | & & | \\ q_1 & \cdots & q_k \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_{k+1} \\ | & & | \end{bmatrix} \begin{bmatrix} h_{11} & \cdots & h_{1k} \\ h_{21} & \cdots & \vdots \\ & \ddots & \vdots \\ & & h_{k+1,k} \end{bmatrix}$$

upper Hessenberg matrix

$$A Q_k = Q_{k+1} H_{k+1,k}$$

$$A q_k = h_{1k} q_1 + h_{2k} q_2 + \dots + h_{kk} q_k + h_{k+1\,k} q_{k+1}$$

$$h_{k+1,k} q_{k+1} = A q_k - h_{1k} q_1 - h_{2k} q_2 - \dots - h_{kk} q_k$$

Eigenvalues from Arnoldi

The Arnoldi iteration is computing

$$H_k = Q_k^T A Q_k$$

If we continue until $k =$ size of A, we have

$$H = Q^T A Q$$

a Hessenberg matrix similar to A.
It therefore has the same eigenvalues.

In practice, we don't continue that far, but stop for some k. The eigenvalues of H_k are usually good approximations to the extreme eigenvalues of A.

Symmetric Matrices

$A = S$

1. Then $H_k = Q_k^T S Q_k$ is also symm.
2. H_k is tridiagonal

only 1 orthogonalization is needed in the Arnoldi iteration!

Lanczos iteration

$q_0 = 0, \quad q_1 = b/\|b\|$

for $k = 1, 2, 3, \dots$

$\qquad v = S q_k$

$\qquad a_k = q_k^T v$

$\qquad v = v - b_{k-1} q_{k-1} - a_k q_k$

$\qquad b_k = \|v\|$

$\qquad q_{k+1} = v / b_k$

$$\begin{pmatrix} & & \\ & S & \\ & & \end{pmatrix} \begin{pmatrix} | & & | \\ q_1 & \cdots & q_k \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ q_1 & \cdots & q_{k+1} \\ | & & | \end{pmatrix} \begin{pmatrix} a_1 & \boxed{b_1} & & & \\ b_1 & \boxed{a_2} & b_2 & & \\ & b_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{k-1} \\ & & & & a_k \\ & & & & b_k \end{pmatrix}$$

$$T_k = Q_k^T S Q_k$$

$$S Q_k = Q_{k+1} T_{k+1,k}$$

Lanczos algorithm presented above is unstable numerically.

Lanczos compared with Householder tridiagonalization:

- Lanczos takes advantage of sparsity
  Householder has fill-in.
- Lanczos uses A as a black box
- each iteration of Lanczos produces q_k
  Housholder produces factor H_k of Q
- Householder is stable

Residual + Stopping criteria

$$r = b - Ax \qquad e_k = x_k - x$$
$$r_k = b - Ax_k$$

$\|r_k\|$ small

When is that good enough?

We actually want $\|e_k\|$ small

$$\begin{aligned}\|r_k\| &= \|b - Ax_k\| \\ &= \|Ax - Ax_k\| \\ &= \|A(x - x_k)\| \\ &= \|Ae_k\| \le \|A\|\,\|e_k\|\end{aligned}$$

$$r_k = -Ae_k$$

$$\Rightarrow \quad e_k = -A^{-1}r_k$$

$$\|e_k\| = \|A^{-1}r_k\| \le \|A^{-1}\|\,\|r_k\|$$

divide both sides by $\|x_k\|$

$$\frac{\|e_k\|}{\|x_k\|} \le \frac{\|A^{-1}\|\,\|r_k\|}{\|x_k\|}$$

multiply numerator + denominator on rhs by $\|A\|$

$$\frac{\|e_k\|}{\|x_k\|} \le \frac{\|A^{-1}\|\,\|A\|\,\|r_k\|}{\|x_k\|\,\|A\|} = \mathrm{cond}_2(A)\,\frac{\|r_k\|}{\|A\|\,\|x_k\|}$$

small relative residual and well-conditioned $A \Rightarrow \boxed{\text{small relative error!}}$

Q What if relative residual is large?

Let E be s.t. $(A+E)\,x_k = b$

i.e., x_k is the exact sol'n to $(A+E)x = b$

$$\|r_k\| = \|b - Ax_k\| = \|Ex_k\| \le \|E\|\,\|x_k\|$$

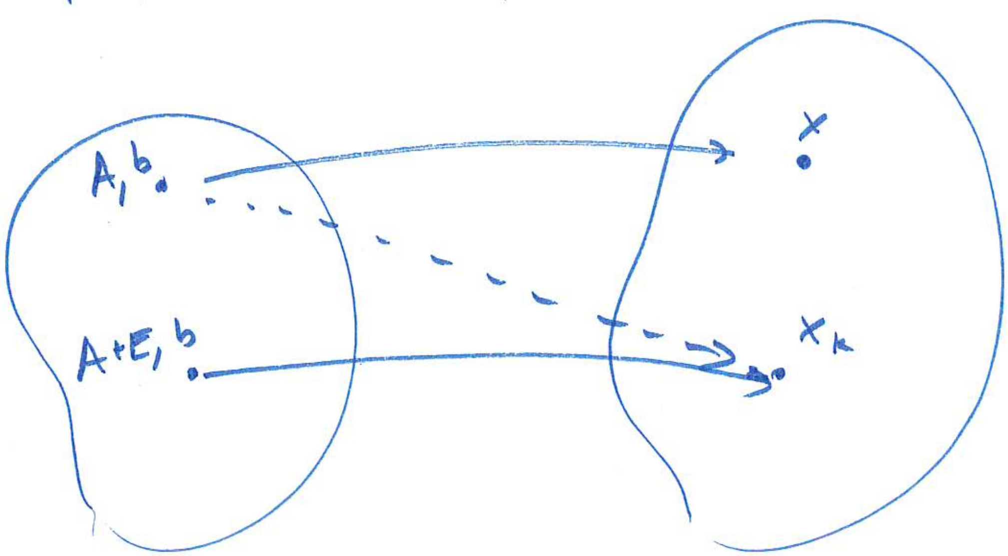
Divide both sides by $\|A\|\,\|x_k\|$:

$$\frac{\|r_k\|}{\|A\|\,\|x_k\|} \le \frac{\|E\|}{\|A\|}\,\frac{\cancel{\|x_k\|}}{\cancel{\|x_k\|}}$$

result:

$$\boxed{\frac{\|r_k\|}{\|A\|\,\|x_k\|} \le \frac{\|E\|}{\|A\|}}$$

large relative residual $\Longrightarrow$ large backward error



Linear Systems by Arnoldi & GMRES

Solve $Ax = b$

GMRES = generalized minimal residuals

$$\mathcal{K}_r = \text{span}\{b, Ab, \dots, A^{r-1}b\}$$

Krylov subspace

main idea of GMRES:

at step k, choose $x_k \in \mathcal{K}_k$ that minimizes the norm of the residual $r_k = b - Ax_k$.

$$\underset{x_k \in \mathcal{K}_k}{\arg\min} \|b - Ax_k\|_2 = x_k$$

Arnoldi gives us an orthonormal basis for $\mathcal{K}_k$.

We can write the L.S. problem above as

$$y_k = \underset{y}{\arg\min} \|b - AQ_k y\|_2, \quad \text{where } x_k = Q_k y_k$$

Recall from Arnoldi,

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

orthogonal:

$$Q = \left(Q_{k+1} \middle| \tilde{Q}_{k+1} \right)$$

$$\Rightarrow \|b - AQ_k y\| = \|b - Q_{k+1} H_{k+1,k} y\|$$

$$= \|Q_{k+1}^T b - H_{k+1,k} y\| + \|\tilde{Q}_{k+1}^T b - \cancelto{0}{\tilde{Q}_{k+1}^T Q_{k+1}} H_{k+1,k} y\|$$

$$\| Q_{k+1}^T b - H_{k+1,k} y \|$$

Note $q_1 = \dfrac{b}{\|b\|}$ in Arnoldi. Then

$$Q_{k+1}^T b = \|b\| \vec{e}_1$$

So the least squares problem solved by GMRES is

$$\min_y \| \, \|b\| e_1 - H_{k+1,k} y \|_2$$

At each step k, solve for y. Set $x_k = Q_k y$.

GMRES Algorithm (high level)

$q_1 = b / \|b\|$

for $k = 1, 2, \ldots$

do step k of Arnoldi

$\rightsquigarrow \quad A Q_k = Q_{k+1} H_{k+1,k}$

find y that minimizes $\| \, \|b\| e_1 - H_{k+1,k} y \|_2$

$x_k = Q_k y$

end

Linear Systems by Arnoldi and GMRES

Arnoldi gives an orthonormal basis for each Krylov subspace $\mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_r$

GMRES : find a vector x_k in $\mathcal{K}_k$ that minimizes $\| b - Ax_k \|$

GMRES = Generalized Minimum RESidual

I.e. $\quad x_k = Q_k y_k$

$$\min \| b - Ax_k \|_2^2$$

$$= \| b - AQ_k y_k \|_2^2$$

$$= \| Q_{k+1}^T b - Q_{k+1}^T A Q_k y_k \|_2^2 + \| \cancelto{0}{\tilde{Q}_{k+1}^T b} - \tilde{Q}_{k+1}^T A Q_k y_k \|_2^2$$

$$= \underbrace{\| \, \|b\| \vec{e}_1 - H_{k+1,k} \, y_k \|_2^2}_{\text{least squares problem}} + \cancelto{0}{\| \underset{(n-k)\times n}{\tilde{Q}_{k+1}^T} \, \underset{n\times k+1}{Q_{k+1}} \, \underset{k+1\times k}{H_{k+1,k}} \, y_k \|_2^2}$$

The zeros below the first subdiagonal in $H_{k+1,k}$ make this fast.

<u>GMRES</u> :
- calculate q_{k+1} with Arnoldi
- find y_k which minimized $\| r_k \|_2$
- compute $x_k = Q_k y_k$
- stop if residual is small enough.

The L.S. problem can be solved by QR. It is only necessary to update the QR factorization in each iteration by 1 Given rotation (orthogonal matrix)