Last time: Conjugate gradient

$A \in \mathbb{R}^{n\times n}$, $A^T = A$, positive definite

solve $Ax = b \iff \min_{x\in\mathbb{R}^n} f(x) = \min_{x\in\mathbb{R}^n} \frac{1}{2}x^TAx - b^Tx$

Given initial $x_0 \in \mathbb{R}^n$, $r_0 = b - Ax_0 \in \mathbb{R}^n$, $d_0 = r_0$

For $k = 1, 2, 3, \cdots$

$$w_{k-1} = A\,d_{k-1}$$

$$\alpha_{k-1} = \frac{d_{k-1}^T r_{k-1}}{d_{k-1}^T w_{k-1}}$$

$$x_k = x_{k-1} + \alpha_{k-1} d_{k-1}$$

$$r_k = r_{k-1} - \alpha_{k-1} w_{k-1}$$

$$d_k = r_k + d_{k-1}\frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$$

end

Key properties:

1) $r_i \perp r_k$, $i = 0, \cdots, k-1$
2) $d_i \perp_A d_k$, $i = 0, \cdots, k-1$
3) $d_j \perp r_k$, $j = 0, \cdots, k-1$

CG has faster convergence than GD. But still want to accelerate in some cases.

- Preconditioned CG

1) Convergence rate of CG depends on $\kappa_2(A)$

Let $M \in \mathbb{R}^{n\times n}$ be invertible. apply CG to

$$MAx = Mb$$

Goal: (Set up M cost) + (MA-iteration cost) < (A-iteration cost)

instead. Hopefully $\kappa_2(MA) << \kappa_2(A)$. ($M \approx A^{-1}$)

2) Immediate problem: MA is not even symmetric in general.

3) Solution: Let $S \in \mathbb{R}^{n\times n}$, invertible. Let $M = S^TS$

Solve $\underbrace{SAS^T}_{=:A'}\underbrace{(S^{-T}x)}_{=:x'} = \underbrace{Sb}_{=:b'}$ by CG

4) Forming SAS^T is costly. Rewrite algo. to implicity forming SAS^T.

Given initial $x_0 \in \mathbb{R}^n$, $r_0' = b' - A'x_0' \in \mathbb{R}^n$, $d_0' = r_0'$

$\Rightarrow$ $r_0' = S(b - Ax_0) = Sr_0$,

$\Rightarrow$ Change of variables: $r_k' = S r_k$, $x_k' = S^{-T} x_k$, $d_k' = S^{-T} d_k$

<u>Algorithm (PCG):</u>

Given initial $x_0 \in \mathbb{R}^n$, $r_0 = b - Ax_0 \in \mathbb{R}^n$, $d_0 = Mr_0$.

For $k = 1, 2, 3, \cdots$

$$w_{k-1} = A\, d_{k-1}$$

$$\alpha_{k-1} = \frac{d_{k-1}^T r_{k-1}}{d_{k-1}^T w_{k-1}}$$

$$x_k = x_{k-1} + \alpha_{k-1} d_{k-1}$$

$$r_k = r_{k-1} - \alpha_{k-1} w_{k-1}$$

$$d_k = Mr_k + d_{k-1} \frac{r_k^T M r_k}{r_{k-1}^T M r_{k-1}} \longleftarrow \quad d_k = Mr_k \text{ Preconditioned GD}$$

end

- Preconditioned GD (Newton-type method)

search direction $d_k \approx Jf(x_k)^{-1}(-\nabla f(x_k))$

$Jf(x_k)^{-1} = A^{-1}$, hard to compute

Instead, do $d_k = -M\nabla f(x_k) = Mr_k$

• Choice of preconditioners:

- Ideally, choose $M = A^{-1} \Rightarrow MA = I \Rightarrow \kappa_2(MA) = 1$

or choose $S^T S = A^{-1} \Rightarrow SAS^T = I \Rightarrow \kappa_2(SAS^T) = 1$

but need to solve $Mr = A^{-1} r$ ← doesn't make any sense

- In practice, choose $M \approx A^{-1}$ and Mr easy to evaluate

1) (block) Jacobi : $M = (\text{diagonal / bands / blocks of } A)^{-1}$

or $S = (\ \cdots\cdots\ )^{-1/2}$

only work if A "dominated" by diagonal parts

2) $M = \tilde{U}^{-1} \tilde{L}^{-1}$, $\tilde{L}\tilde{U} \approx A$ incomplete LU of A

drop large entries in LU decomp of A to make it sparse

or $M = \tilde{L}^{-T} \tilde{L}^{-1}$, $\tilde{L}\tilde{L}^T \approx A$ incomplete Cholesky
(s.p.d.)

$S = \tilde{L}^{-1}$

3) Use stationary iterative solvers as preconditioner

$$A = C - D \Rightarrow C x_+ = D x + b$$

$$\Rightarrow x_+ = C^{-1} D x + C^{-1} b$$

For a convergent method, $\rho(C^{-1}D) < 1$

$$\Rightarrow \rho(C^{-1}(C - A)) < 1 \Rightarrow \rho(I - C^{-1}A) < 1$$

in some sense
$\Rightarrow C^{-1} \approx A^{-1}$

take this to be M

ex. In Symmetric Gauss-Seidel

$$B = (D+U)^{-1} L (D+L)^{-1} U$$

Since $B = I - C^{-1}A$

$$\Rightarrow M = C^{-1} = (I-B)A^{-1} = (D+U)^{-1} D (D+L)^{-1}$$

or $S = D^{\frac{1}{2}} (D+L)^{-1}$

4) Incomplete Cholesky (enforce sparsity)

5) Randomized preconditioner (Sketching etc.)

- Biorthogonalization methods

1) From a perspective of projection methods, Krylov methods

Step 1: find basis V_k for $\mathcal{K}$ and basis W_k for $\mathcal{L}$

Step 2: then $(W_k^* A V_k) y_k = W_k^* r_0 \Rightarrow x_k = x_0 + V_k y_k$

2) In Arnoldi's iteration, when $A^* = A \in \mathbb{C}^{n \times n}$
(Lanczos)

$$H_k = V_k^* A V_k$$

is tridiagonal, Hermitian. V_k orthonormal basis of $\mathcal{K}_k$.

Lanczos is Computationally efficient:

improve $O(k^2 n)$ flops to $O(kn)$ flops; $O(kn)$ memory to $O(n)$ (CG)

3) For A non-Hermitian, still want tridiagonal structure.

what to do ?

- Idea: Relax the restriction on V_k, W_k being orthonormal basis $(V_k^* V_k = W_k^* W_k = I)$

Instead take $V_k, W_k \in \mathbb{C}^{n \times k}$ to be biorthogonal. i.e., $W_k^* V_k = I_k$

s.t. $W_k^* A V_k$ tridiagonal

- Take $\mathcal{K} = \mathcal{K}_k(A, r_0)$, $\mathcal{L} = \mathcal{K}_k(A^*, \tilde{r}_0)$

1) We can assume a Lanczos recurrence for the basis generated, i.e.

$$(\star)\quad \begin{aligned} A V_k &= V_{k+1} \tilde{T}_k \\ A^* W_k &= W_{k+1} \tilde{S}_k \end{aligned}$$

$\tilde{T}_k, \tilde{S}_k \in \mathbb{C}^{(k+1)\times k}$ tridiagonal matrices (may not be Hermitian)

2) We use biorthogonality condition, $w_i^* v_j = \delta_{ij}$.

i.e. $W_k^* V_k = I_k$

$$\Rightarrow \quad W_k^* A V_k = W_k^* V_{k+1} \tilde{T}_k = T_k \qquad \tilde{T}_k = \begin{bmatrix} T_k \\ \delta_{k+1} e_k \end{bmatrix}$$

$$V_k^* A W_k = V_k^* W_{k+1} \tilde{S}_k = S_k \qquad \tilde{S}_k = \begin{bmatrix} S_k \\ \beta_{k+1} e_k \end{bmatrix}$$

$$\Rightarrow \quad T_k = S_k^*$$

3) Let $T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & \delta_k & \alpha_k \end{bmatrix} \in \mathbb{C}^{k\times k}$ let $w_0 = v_0 = 0$

Then from $(\star)$ $\Rightarrow$

$$(\star\star)\quad \begin{aligned} A v_j &= \beta_j v_{j-1} + \alpha_j v_j + \delta_{j+1} v_{j+1} \\ A^* w_j &= \bar{\delta}_j w_{j-1} + \bar{\alpha}_j w_j + \bar{\beta}_{j+1} w_{j+1} \end{aligned} \qquad j = 1, \dots, k$$

and also

$$A V_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T$$

$$A^* W_k = W_k T_k^* + \beta_{k+1} w_{k+1} e_k^T$$

- How to compute $\alpha_j, \beta_j, \delta_j$ so that $W_k^* V_k = I_k$?

1) For $j = 1$, v_1, w_1 available, $v_1^* w_1 = 1$

$$(\star')\quad \begin{aligned} A v_1 &= \alpha_1 v_1 + \delta_2 v_2 . \\ A^* w_1 &= \bar{\alpha}_1 w_1 + \bar{\beta}_2 w_2 \end{aligned}$$

want to choose $\alpha_1, \delta_2, \beta_2$,
s.t. $v_1^* w_2 = w_1^* v_2 = 0$, $w_2^* w_2 = v_2^* v_2 = 1$

$$(\star') \xrightarrow[v_2^*\cdot]{w_2^*\cdot} \quad w_2^* A v_1 = \delta_2, \quad v_2^* A^* w_1 = \bar{\beta}_2$$

$$(\star') \xrightarrow{w_1^*\cdot} \quad w_1^* A v_1 = \alpha_1$$

2) Suppose now $\alpha_1, \cdots, \alpha_{j-1}$, $\beta_1, \cdots, \beta_j$, $\delta_1, \cdots, \delta_j$ are available

$v_1, \cdots, v_j$, $w_1, \cdots, w_j$, $v_s^* w_t = \delta_{st}$, $s,t = 1, \cdots, j$

want to find α_j, β_{j+1}, δ_{j+1}, s.t. v_{j+1}, w_{j+1} satisfy

$$w_{j+1}^* v_{j+1} = 1, \quad w_i^* v_{j+1} = v_i^* w_{j+1} = 0, \quad \forall i = 1, \cdots, j$$

- let $\alpha_j = w_j^* A v_j$

$$\hat{v}_{j+1} = A v_j - \beta_j v_{j-1} - \alpha_j v_j$$

$$\hat{w}_{j+1} = A^* w_j - \bar{\delta}_j w_{j-1} - \bar{\alpha}_j w_j$$

then $\left.\begin{aligned} w_j^* \hat{v}_{j+1} &= w_j^* A v_j - \alpha_j = 0 \\ v_j^* \hat{w}_{j+1} &= 0 \end{aligned}\right\}$ what about $w_i^* v_{j+1}$ and $v_i^* w_{j+1}$, $i = 1, \cdots, j-1$?

- let $v_{j+1} := \hat{v}_{j+1} / \delta_{j+1}$, $w_{j+1} := \hat{w}_{j+1} / \beta_{j+1}$

choose δ_{j+1}, β_{j+1} such that

$$w_{j+1}^* v_{j+1} = \frac{\hat{w}_{j+1}^* v_{j+1}}{\delta_{j+1} \beta_{j+1}} = 1$$

so we can choose $\delta_{j+1} := \sqrt{|\hat{w}_{j+1}^* \hat{v}_{j+1}|}$

$$\beta_{j+1} := (\hat{w}_{j+1}^* \hat{v}_{j+1}) / \delta_{j+1}$$

Algorithm (Lanczos biorthogonalization)

Given $v_1, w_1 \in \mathbb{C}^n$ s.t. $w_1^* v_1 = 1$, let $w_0 = v_0 = 0 \in \mathbb{C}^n$.

For $j = 1, 2, 3, \ldots$

$$\alpha_j = w_j^* A v_j$$

$$\hat{v}_{j+1} = A v_j - \beta_j v_{j-1} - \alpha_j v_j$$

$$\hat{w}_{j+1} = A^* w_j - \bar{\delta}_j w_{j-1} - \bar{\alpha}_j w_j$$

$$\delta_{j+1} = \sqrt{|\hat{w}_{j+1}^* \hat{v}_{j+1}|} \; . \quad \text{If } \underline{\delta_{j+1} = 0,} \text{ stop}$$

$$\beta_{j+1} = \hat{w}_{j+1}^* \hat{v}_{j+1} / \delta_{j+1}$$

$$w_{j+1} = \hat{w}_{j+1} / \delta_{j+1}$$

$$v_{j+1} = \hat{v}_{j+1} / \beta_{j+1}$$

end

If this happens for $j < k-1$ this is call serious breakdown as it provides little information compared to the "lucky" breakdown in Arnoldi

Exercise: The w_{j+1}, v_{j+1} constructed at step j satisfy

$$w_i^* v_{j+1} = v_i^* w_{j+1} = 0, \quad i = 1, 2, \cdots, j-1$$

The Lanczos biorthogonalization process produces

V_k as a basis for $K_k(A, r_0)$

W_k as a basis for $K_k(A^*, \tilde{r}_0)$

s.t. $W_k^* A V_k = T_k$

V_k, W_k can be used in different projection methods

- Biconjugate Gradient (BiCG / BCG)

1) Goal: Find $x_k \in x_0 + K_k(A, r_0)$

$\tilde{x}_k \in \tilde{x}_0 + K_k(A^*, \tilde{r}_0)$

s.t. $r_k := b - A x_k \perp K_k(A^*, \tilde{r}_0)$

$\tilde{r}_k := b - A^* \tilde{x}_k \perp K_k(A, r_0)$

$$\Longleftrightarrow \quad x_k = x_0 + V_k y_k$$

$$\tilde{x}_k = \tilde{x}_0 + W_k \tilde{y}_k$$

take $v_1 = \frac{r_0}{\|r_0\|_2} = \frac{r_0}{\beta}$

y_k solves $(W_k^* A V_k) y_k = W_k^* r_0 = \beta v_1$

$\tilde{y}_k$ solves $(V_k^* A^* W_k) \tilde{y}_k = V_k^* \tilde{r}_0 = \tilde{\beta} w_1$

i.e. $$\left.\begin{aligned} T_k y_k &= \beta v_1 \\ T_k^* \tilde{y}_k &= \tilde{\beta} w_1 \end{aligned}\right. \quad (\Delta)$$

2) We can solve (Δ) through LU decomposition

Let $T_k = L_k U_k$, $D_k = V_k U_k^{-1} \in \mathbb{C}^{n\times k}$, $z_k = L_k^{-1}(\beta e_1) \in \mathbb{C}^n$

then $x_k = x_0 + D_k z_k$

From the structure of L_k, U_k, we have

$$D_k = [D_{k-1}, d_{k-1}], \quad z_k = \begin{bmatrix} z_{k-1} \\ \zeta_{k-1} \end{bmatrix}$$

thus, $x_k = x_{k-1} + \zeta_{k-1} d_{k-1}$

Furthermore, we have (Why?) $$\left.\begin{aligned} r_k &= b - A x_k = \sigma_k v_{k+1}, \\ \tilde{r}_k &= b - A^* \tilde{x}_k = \tilde{\sigma}_k w_{k+1} \end{aligned}\right\} \Rightarrow r_j \perp \tilde{r}_i, \ i \neq j$$

If we define $\tilde{D}_k = W_k (L_k^{-1})^*$, then it can be showed that

$$\tilde{D}_k^* A D_k = I_k \longleftarrow \tilde{d}_i \perp_A d_j \quad i \neq j$$

exercise

Here $\tilde{D}_k = [\tilde{D}_{k-1}, \tilde{d}_{k-1}] \in \mathbb{C}^{n\times k}$

Algorithm (BiCG)

Given $x_0 \in \mathbb{C}^n$, compute $d_0 = b - A x_0$,

Choose $\tilde{r}_0 \in \mathbb{C}^n$, s.t. $\tilde{r}_0^* r_0 \neq 0$

let $d_0 = r_0 = b - A x_0$. $\tilde{d}_0 = \tilde{r}_0$

For $j = 1, 2, \cdots$

$$\alpha_j = \frac{\tilde{r}_j^* r_j}{\tilde{d}_j^* A d_j}$$

$$x_{j+1} = x_j + \alpha_j d_j$$

$$r_{j+1} = r_j - \alpha_j A d_j$$

$$\tilde{r}_{j+1} = \tilde{r}_j - \alpha_j A^* \tilde{d}_j$$

$$\beta_j = \frac{\tilde{r}_{j+1}^* r_{j+1}}{\tilde{r}_j^* r_j}$$

$$d_{j+1} = r_{j+1} + \beta_j d_j$$

$$\tilde{d}_{j+1} = \tilde{r}_{j+1} + \beta_j \tilde{d}_j$$

end

Other methods:

1) Quasi minimal residual (QMR)

From Lanczos, $AV_k = V_{k+1} \tilde{T}_k$

Since $x_k = x_0 + V_k y_k$,

the residual

$$\|b - Ax_k\|_2 = \|r_0 - AV_k y_k\|_2 = \|V_{k+1}(\beta \tilde{e}_1 - \tilde{T}_k y_k)\|_2$$

but now V_{k+1} is not a orthonormal basis!

can't remove V_{k+1} directly

Nevertheless we can find y_k by minimizing

$$\min_{y \in \mathbb{C}^k} \|\beta \tilde{e}_1 - \tilde{T}_k y_k\|_2 \longleftarrow \text{Quasi-residual norm}$$

2) Stabilized BiCG (BiCGSTAB)

Avoid A^*d in BiCG and in a numerically more stable way (compared to conjugate gradients squared CGS)

A lot others ...

In practice, which solver should I use?

- When $n < 10^4$, use dense solvers (LAPACK)
- When $n < 10^6$, use sparse direct solvers if accuracy is crucial

useful if A is generated from PDE / mesh

Otherwise, if $A^* = A$, use PCG

otherwise, use GMRES as a first try

if want ε_{mach} residual, use BiCGSTAB