Note Title 2024-12-16

$Ax = y \qquad A \in \mathbb{R}^{n \times n}$ large and sparse

Direct methods

general $A = LU$

symm. $A = LDL^T$

posdef Cholesky $A = R^T R$

Effective when the factors are sparse

("large-world models", e.g. road networks)

Iterative methods

GMRES

MINRES

Conjugate gradient

Effective when eigenvalues are clustered

Trick: reordering: we can apply a symmetric permutation

$$A \to P^T A P \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Without reordering:

$A = R^T R$

$$Ax = y \iff R^T \underbrace{Rx}_{z} = y \qquad \begin{cases} R^T z = y \\ Rx = z \end{cases}$$

1. $R = \text{chol } A$
2. $z = R^T \backslash y$ forward subst.
3. $x = R \backslash z$ back-substitution

With reordering:

$R^T R = P^T A P$

$$P^T A P P^T x = P^T y$$

$$\Updownarrow$$

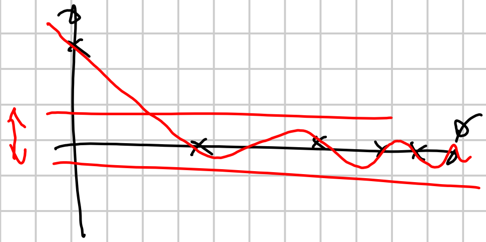
$$R^T \underbrace{R \underbrace{P^T x}_{z}}_{w} = P^T y \iff \begin{cases} R^T w = P^T y \\ R z = w \\ P^T x = z \end{cases} \qquad x = Pz$$

~30% cpu savings in our example, due to sparser R.

"fill-in" is reduced.

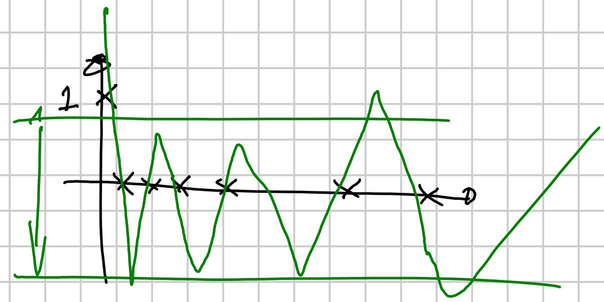
small

$\min_q \max_\lambda |q(\lambda)|$



large

$\min_q \max_\lambda |q(\lambda)|$



⚠ Don't use $\mathrm{inv}(A)$
- slower
- less accurate: $A^{-1} = \begin{bmatrix} A^{-1}e_1 & A^{-1}e_2 & \cdots & A^{-1}e_n \end{bmatrix}$, so when computing $A^{-1}y = A^{-1}e_1 y_1 + \dots + A^{-1}e_n y_n$ we are as accurate as the worse of these linear systems!

Let $A \in \mathbb{R}^{n \times n}$, non symmetric, with "bad" sparsity pattern and "bad" eigenvalues.

Transforming $Ax = y$ to $PAx = Py$, for any P invertible, gives an equivalent linear system but different eigenvalue location!

<u>Preconditioning</u>: P is a <u>preconditioner</u> for the problem (or P^{-1})

<u>Idea</u>: if $PA \approx I$, its eigenvalues are close to 1.

$P \approx A^{-1}$ improves eigenvalue location.

We want P such that (i) cheap to compute

~~(2) cheap to invert~~

(3) $PA \approx I$

Ex: $P = \begin{bmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & & \ddots & \\ 0 & & & a_{nn} \end{bmatrix}^{-1}$ (effective only on some problems)

Better strategy: incomplete LU factorization

Start computing an LU factorization, and set small entries to $\emptyset$ artificially along the way!

$$A \approx LU$$

not equal any more! — much more sparse!

$$P = \underbrace{U^{-1}L^{-1}} \approx A$$

For iterative methods, we do not need the entries of A, but just a way to compute $w \mapsto Aw$

Instead of passing $P \cdot A$, we pass the function

$$w \to U^{-1}L^{-1}Aw = U \backslash (L \backslash (Aw))$$

Better to use this "callback form" of gmres/pcg when working with preconditioners.

⚠ The residual $\boxed{A\tilde{x} - y}$ is not equal in general to the "preconditioned residual" $\boxed{PA\tilde{x} - Py}$.

This is called "left preconditioning". There is also "right preconditioning":

$$(AP)(P^{-1}x) = y$$

Is there a way to use preconditioning for <u>symmetric</u> systems? Note that even if A is symmetric, PA usually is not $\Rightarrow$ pcg(PA, Py) fails!

<u>Idea</u>: multiply from both sides to keep symmetry!

Look for P such that

$$\boxed{P^T A P \approx I}$$

and solve with an iterative method

$$\underbrace{P^T A P}_{\hat{A}} \underbrace{P^{-1} x}_{\hat{x}} = \underbrace{P^T y}_{\hat{y}} \iff Ax = y$$

symmetric preconditioning

A symm. pos. def. $\iff$ $P^T A P$ symm. pos. def.

General preconditioner for A symm. pos. def.:

<u>incomplete Cholesky</u>

>> R = ichol(A) returns a sparse R s.t. $\boxed{A \approx R R^T}$

$A \approx R R^T$ $\qquad$ $R^{-1} A (R^T)^{-1} \approx I$

$$\underbrace{R^{-1} A R^{-T}}_{\hat{A}} \underbrace{R^T x}_{\hat{x}} = \underbrace{R^{-1} y}_{\hat{y}}$$

$$\hat{A} w = R^{-1} A R^{-T} w$$