Efficient Methods for Sparse Linear Systems

If spectral $\Rightarrow$ FFT !

If non-spectral (FD, FE):

- elimination (direct)
- iterative $\rightarrow$ multigrid
- Krylov methods (e.g. conjugate gradients)

Elimination Methods

Solve $A \cdot x = b$

Matlab: `x = A\b`

If A square, regular: can use elimination methods

$$\begin{Bmatrix} A \text{ symmetric positive definite:} & \text{Cholesky factorization} \\ \text{Otherwise:} & \text{LU factorization} \end{Bmatrix}$$

Fill-in

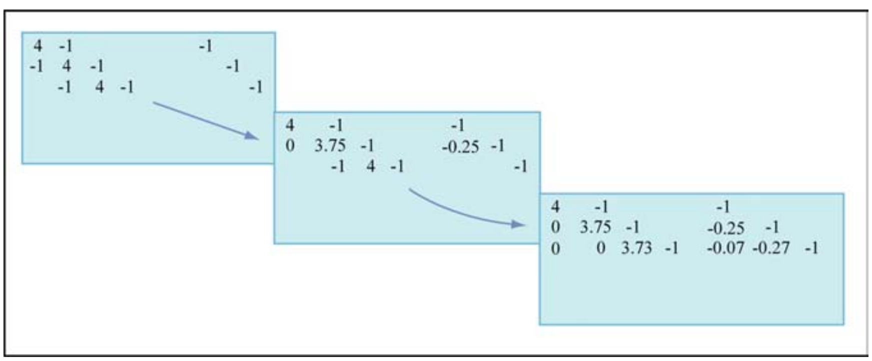


Image by MIT OpenCourseWare.

Adding rows creates nonzero entries and may thus destroy sparsity.

Matlab:

```
[L,U] = lu(A);
spy(L)
spy(U)
```

Minimum degree algorithms:

Reduce fill-in by reordering of rows and columns

Ex.: Red-black ordering for K2D

Matlab:

A non-symmetric	A symmetric

`p = colamd(A)`
`[L,U] = lu(A(p,:))`
 	
Strategy: Choose remaining column with fewest nonzeros
	`p = symamd(A)`
	`[L,U] = lu(A(p,p))`
	or
	`[L,U] = chol(A(p,p))`
	Strategy: Choose remaining meshpoint with fewest neighbors

Further:
- Graph separators
- Nested dissection

Elimination is great for small matrices whose entries are directly accessible.

Preconditioning

$A \cdot x = b$

Condition number: $\kappa = \text{cond}(A) = ||A|| \cdot ||A^{-1}||$

A symmetric: $\text{cond}_2(A) = \dfrac{|\lambda_{\max}|}{|\lambda_{\min}|}$

$\text{cond}(A) \gg 1 \Rightarrow$ small error in b can yield large error in x

Formulate equivalent system which is better conditioned.

Left preconditioning:		solve	$(P^{-1}A) \cdot x = P^{-1}b$
Right preconditioning:	1.	solve	$(AP^{-1}) \cdot y = b$
	2.	solve	$P \cdot x = b$

Ex.: $A = \begin{bmatrix} 1 & 1 \\ 1 & 1000 \end{bmatrix}$ $\qquad \lambda \in \{0.999, 1000.001\} \Rightarrow \text{cond}(A) \approx 1000$

$P = \text{diag}(A) = \begin{bmatrix} 1 & 0 \\ 0 & 1000 \end{bmatrix}$

$\text{cond}_2(P^{-1}A) = \text{cond}(AP^{-1}) \approx 2.65$

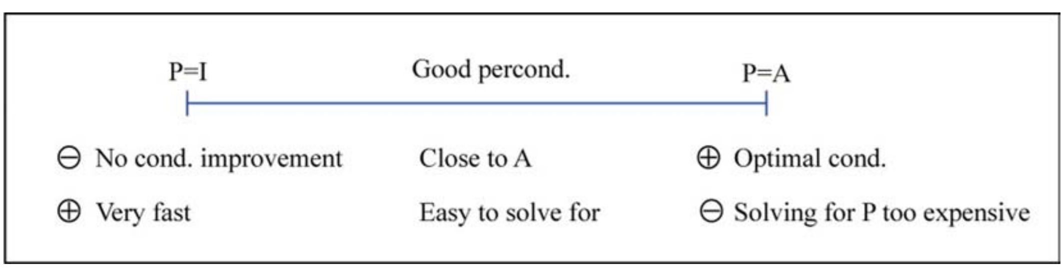


Image by MIT OpenCourseWare.

Ex.:

- $P = D$
- $P = D + L$

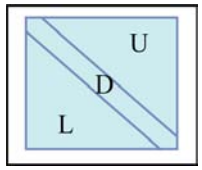
$\left.\right\} A =$ 

Image by MIT OpenCourseWare.

- $P = L_{\text{app}} \cdot U_{\text{app}}$ (ILU = Incomplete LU factorization) [Matlab: `luinc`]

Iterative Methods

$A \cdot x = b$

$\Leftrightarrow x = (I - A) \cdot x + b$ splitting

$$\left\{ \begin{array}{rcl} x^{(k+1)} & = & (I - A) \cdot x^{(k)} + b \\ x^{(0)} & = & x_0 \end{array} \right\} \quad \text{iteration}$$

Apply to precondition system: $\left\{ \begin{array}{rcl} (AP^{-1})y & = & b \\ Px & = & y \end{array} \right\}$

$$y^{(k+1)} = (I - AP^{-1})y^{(k)} + b$$

$$\Leftrightarrow Px^{(k+1)} = (P - A)x^{(k)} + b$$

$$\Leftrightarrow x^{(k+1)} = \underbrace{(I - P^{-1}A)}_{=M} x^{(k)} + P^{-1}b$$

$$\Leftrightarrow P \underbrace{(x^{(k+1)} - x^{(k)})}_{\substack{=z^{(k)} \\ \text{update}}} = \underbrace{b - A \cdot x^{(k)}}_{\substack{=r^{(k)} \\ \text{residual}}}$$

Error:

$$x = A^{-1}b$$

$$e^{(k)} = x - x^{(k)}$$

$$\Rightarrow e^{(k+1)} = M \cdot e^{(k)} \text{ [independent of } b \text{]}$$

Iteration converges if $\rho(M) < 1$

Spectral radius $\rho(M) = \max |\lambda(M)|$

Popular Preconditioners:

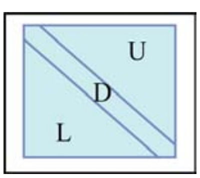
$A =$ 

Image by MIT OpenCourseWare.

$P = D$	Jacobi	$M = I - D^{-1}A$	
$P = D + L$	Gauß-Seidel	$M = I - (D + L)^{-1}A$	(overwrite entries as computed)
$P = D + wL$	SOR (Successive OverRelaxation) [better: SSOR]		

Theorem:

- If A diagonal dominant $\left(|a_{ii}| > \sum_{j \neq 1} |a_{ij}|\right) \Rightarrow$ Jacobi converges
- If Jacobi converges $\Rightarrow$ Gauß-Seidel converges ($\times$ 2 faster)
- If $0 < w < 2 \Rightarrow$ SOR converges

$$w_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \mu^2}} \ , \ \mu = \rho(I - (D + I)^{-1} A).$$

Multigrid

Heat Equation | Iterative Scheme

$$u_t - \nabla^2 u = f \quad \xrightarrow{p \approx \frac{1}{\Delta t} I} \quad P(u^{(k+1)} - u^{(k)}) = -A \cdot u^{(k)} + f$$

(A ↑ Poisson matrix)

Iterative schemes behave like heat equation.
Slow convergence, fast smoothing of error.

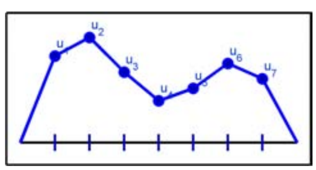
Smoothers:

$P = \frac{3}{2} D$	Weighted Jacobi	
$P = D + L$	Gauß Seidel	(popular)
$P = D + wL$	SOR	(costly)

Smoother reduces high frequency error components fast.
Smooth error is rough on coarser grid.

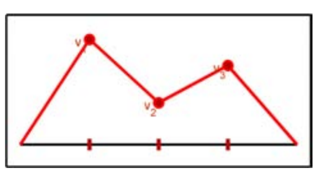
Ex.: $\left\{ \begin{array}{l} -u_{xx} = 1 \\ u(0) = 0 = u(1) \end{array} \right\}$

fine grid ($h = \frac{1}{8}$)



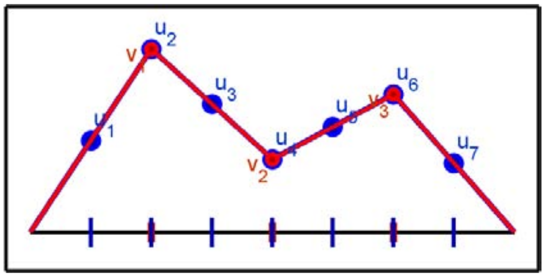
$$A_h = 64 \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

coarse grid



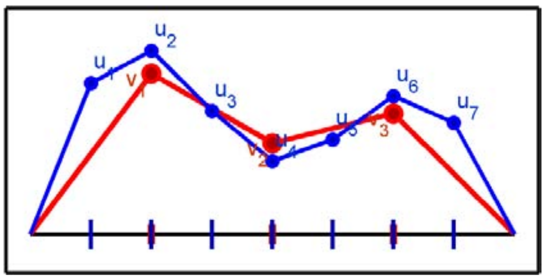
$A_{2h} = \dots$

Interpolation: Linear



$$I = \frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix} \in \mathbb{R}^{7\times 3}$$

Restriction: Full Weighting



$$R = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix}$$

$$\boxed{R = \tfrac{1}{2} I^T}$$

Coarse Grid Matrix:

Galerkin: $A_{2h} = R \cdot A_h \cdot I = 16 \begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix}$

18.336 Numerical Methods for Partial Differential Equations
Spring 2009