

## Preconditioning techniques

- Basic concepts
- Gauss-Seidel and SSOR as preconditioners
- Incomplete LU factorizations
- Level-of-fill and threshold-based methods.
- Multi-elimination, ARMS
- See Chapter 10 of text for details.

## Preconditioning – Basic principles

**Basic idea** is to use the Krylov subspace method on a modified system such as

$$M^{-1}Ax = M^{-1}b.$$

- The matrix  $M^{-1}A$  need not be formed explicitly; only need to solve  $Mw = v$  whenever needed.
- Consequence: fundamental requirement is that it should be easy to compute  $M^{-1}v$  for an arbitrary vector  $v$ .

## Left, Right, and Split preconditioning

Left preconditioning

$$M^{-1}Ax = M^{-1}b$$

Right preconditioning

$$AM^{-1}u = b, \text{ with } x = M^{-1}u$$

Split preconditioning:  $M$  is factored as  $M = M_L M_R$ .

$$M_L^{-1}AM_R^{-1}u = M_L^{-1}b, \text{ with } x = M_R^{-1}u$$

## An observation. Introduction to Preconditioning

➤ Take a look back at basic relaxation methods: Jacobi, Gauss-Seidel, SOR, SSOR, ...

➤ Iterations of the form  $x^{(k+1)} = Mx^{(k)} + f$  where  $M$  is of the form  $M = I - P^{-1}A$ . For example for SSOR,

$$P_{SSOR} = (D - \omega E)D^{-1}(D - \omega F)$$

➤ Referred to as the SSOR preconditioner

➤ The iteration  $x^{(k+1)} = Mx^{(k)} + f$  is attempting to solve  $(I - M)x = f$ . Since  $M \equiv I - P^{-1}A$  this system can be rewritten as  $P^{-1}Ax = P^{-1}b$

In other words:

$$\text{Relaxation iter.} \iff \text{Preconditioned Fixed Point Iter.}$$

## Standard preconditioners

- Simplest preconditioner:  $M = \text{Diag}(A)$  ➤ poor convergence.
- Next to simplest: SSOR.

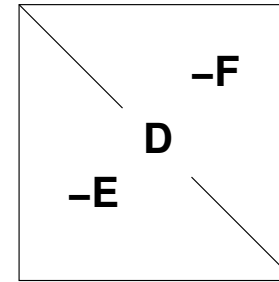
$$M = (D - \omega E)D^{-1}(D - \omega F)$$

- Still simple but often more efficient: ILU(0).
- ILU(p) – ILU with level of fill p – more complex.
- Class of ILU preconditioners with threshold
- Class of approximate inverse preconditioners
- Class of Multilevel ILU preconditioners
- Algebraic Multigrid Preconditioners

14-5

Csci 8314 – March 30, 2014

## The SOR/SSOR preconditioner



- SOR preconditioning

$$M_{SOR} = (D - \omega E)$$

- SSOR preconditioning

$$M_{SSOR} = (D - \omega E)D^{-1}(D - \omega F)$$

- $M_{SSOR} = LU$ ,  $L$  = lower unit matrix,  $U$  = upper triangular. One solve with  $M_{SSOR} \approx$  same cost as a MAT-VEC.

14-6

Csci 8314 – March 30, 2014

- $k$ -step SOR (resp. SSOR) preconditioning:

$$k \text{ steps of SOR (resp. SSOR)}$$

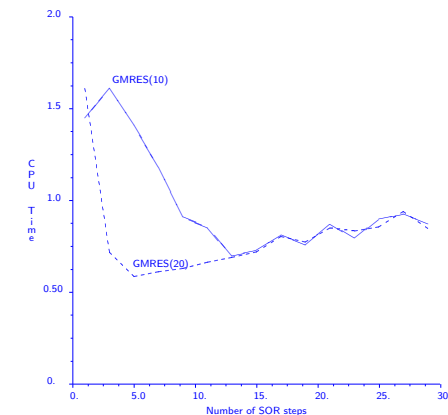
- Questions: Best  $\omega$ ? For preconditioning can take  $\omega = 1$

$$M = (D - E)D^{-1}(D - F)$$

Observe:  $M = LU + R$  with  $R = ED^{-1}F$ .

- Best  $k$ ?  $k = 1$  is rarely the best. Substantial difference in performance.

Iteration times versus  $k$  for SOR( $k$ ) preconditioned GMRES



14-8

Csci 8314 – March 30, 2014

## ILU(0) and IC(0) preconditioners

► **Notation:**  $NZ(X) = \{(i, j) \mid X_{i,j} \neq 0\}$

► Formal definition of ILU(0):

$$\begin{aligned} A &= LU + R \\ NZ(L) \cup NZ(U) &= NZ(A) \\ r_{ij} &= 0 \text{ for } (i, j) \in NZ(A) \end{aligned}$$

► This does not define  $ILU(0)$  in a unique way.

Constructive definition: Compute the LU factorization of  $A$  but drop any fill-in in  $L$  and  $U$  outside of  $\text{Struct}(A)$ .

► ILU factorizations are often based on  $i, k, j$  version of GE.

14-9

Csci 8314 – March 30, 2014

## What is the IKJ version of GE?

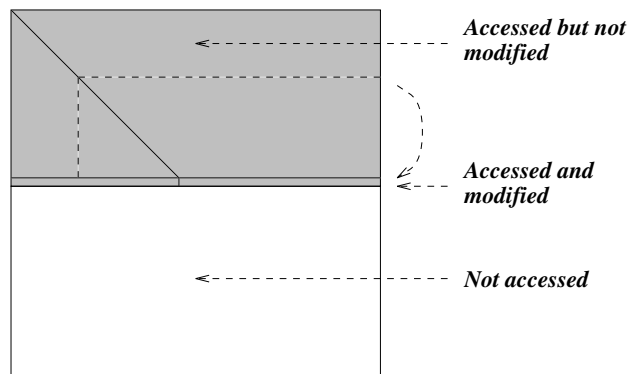
### ALGORITHM : 1. Gaussian Elimination – IKJ Variant

```

1.  For  $i = 2, \dots, n$  Do:
2.    For  $k = 1, \dots, i - 1$  Do:
3.       $a_{ik} := a_{ik} / a_{kk}$ 
4.      For  $j = k + 1, \dots, n$  Do:
5.         $a_{ij} := a_{ij} - a_{ik} * a_{kj}$ 
6.      EndDo
7.    EndDo
8.  EndDo
    
```

14-10

Csci 8314 – March 30, 2014



14-11

Csci 8314 – March 30, 2014

## ILU(0) – zero-fill ILU

### ALGORITHM : 2. $ILU(0)$

```

For  $i = 1, \dots, N$  Do:
  For  $k = 1, \dots, i - 1$  and if  $(i, k) \in NZ(A)$  Do:
    Compute  $a_{ik} := a_{ik} / a_{kk}$ 
    For  $j = k + 1, \dots$  and if  $(i, j) \in NZ(A)$ , Do:
      compute  $a_{ij} := a_{ij} - a_{ik} a_{k,j}$ .
    EndFor
  EndFor
EndFor
    
```

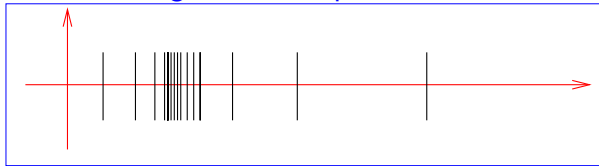
► When  $A$  is SPD then the ILU factorization = Incomplete Choleski factorization – IC(0). Meijerink and Van der Vorst [1977].

14-12

Csci 8314 – March 30, 2014

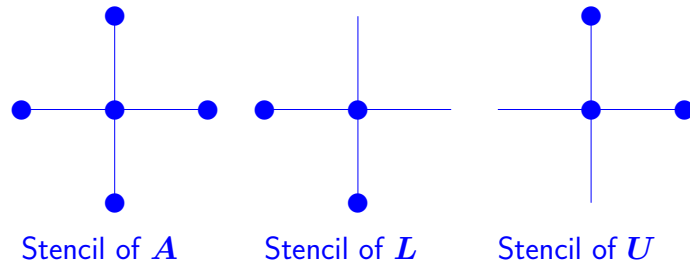
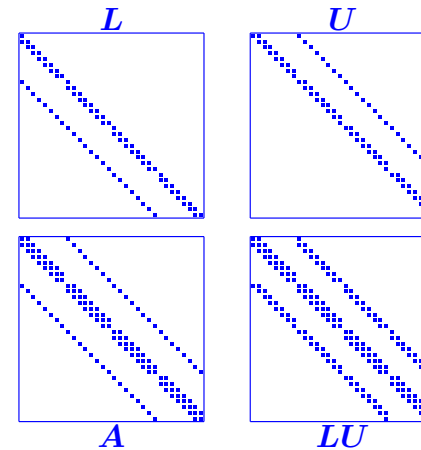
### Typical eigenvalue distribution

- More than anything else, what determines the convergence of an iterative method is the **distribution** of the eigenvalues of the matrix.
- Need to consider eigenvalues of preconditioned matrix  $M^{-1}A$



- Clustering around 1 results in fast convergence

### Pattern of ILU(0) for 5-point matrix



### Higher order ILU factorization

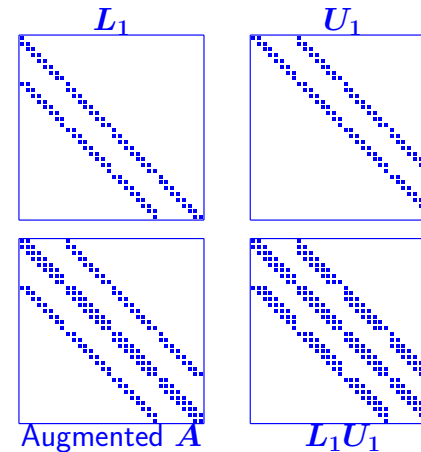
- Higher accuracy incomplete Choleski: for regularly structured problems, IC( $p$ ) allows  $p$  additional diagonals in  $L$ .
- Can be generalized to irregular sparse matrices using the notion of level of fill-in [Watts III, 1979]

- Initially  $Lev_{ij} = \begin{cases} 0 & \text{for } a_{ij} \neq 0 \\ \infty & \text{for } a_{ij} = 0 \end{cases}$
- At a given step  $i$  of Gaussian elimination:

$$Lev_{kj} = \min\{Lev_{kj}; Lev_{ki} + Lev_{ij} + 1\}$$

- ILU(p) Strategy = drop anything with level of fill-in exceeding  $p$ .
- \* Increasing level of fill-in usually results in more accurate ILU and...
- \* ...typically in fewer steps and fewer arithmetic operations.

## ILU(1)



## ALGORITHM : 3. ILU(p)

```

For  $i = 2, N$  Do
  For each  $k = 1, \dots, i - 1$  and if  $a_{ij} \neq 0$  do
    Compute  $a_{ik} := a_{ik}/a_{jj}$ 
    Compute  $a_{i,*} := a_{i,*} - a_{ik}a_{k,*}$ 
    Update the levels of  $a_{i,*}$ 
    In row  $i$ : if  $lev(a_{ij}) > p$  set  $a_{ij} = 0$ 
  EndFor
EndFor

```

- The algorithm can be split into a symbolic and a numerical phase.
- Level-of-fill ➤ in Symbolic phase

## ILU with threshold – generic algorithms

ILU(p) factorizations are based on structure only and not numerical values ➤ potential problems for non M-matrices.

- One remedy: ILU with threshold – (generic name ILUT.)

### Two broad approaches:

*First approach* [derived from direct solvers]: use any (direct) sparse solver and incorporate a dropping strategy. [Munksgaard ('78), Osterby & Zlatev, Sameh & Zlatev'90, D. Young, & al. (Boeing) etc...]

*Second approach* : [derived from 'iterative solvers' viewpoint]

1. use a (row or colum) version of the  $(i, k, j)$  version of GE;
2. apply a drop strategy for the elment  $l_{ik}$  as it is computed;
3. perform the linear combinations to get  $a_{i*}$ . Use full row expansion of  $a_{i*}$ ;
4. apply a drop strategy to fill-ins.

### *ILU with threshold: $ILUT(k, \epsilon)$*

- Do the  $i, k, j$  version of Gaussian Elimination (GE).
  - During each  $i$ -th step in GE, discard any pivot or fill-in whose value is below  $\epsilon \|row_i(A)\|$ .
  - Once the  $i$ -th row of  $L + U$ , (L-part + U-part) is computed retain only the  $k$  largest elements in both parts.
- Advantages: controlled fill-in. Smaller memory overhead.
- Easy to implement – much more so than preconditioners derived from direct solvers.
- can be made quite inexpensive.

### *Other preconditioners*

Many other techniques have been developed:

- Approximate inverse methods
- Polynomial preconditioners
- Multigrid - type methods
- Incomplete LU based on Crout factorization
- Multi-elimination and multilevel ILU (ARMS)