Matthias Bollhöfer
TU Berlin / TU Braunschweig
Dept. of Mathematics

University of Minnesota, April 09, 2004

Outline

1. Theoretical background
2. Multilevel algorithms
3. ILUPACK
   (a) Names, formats and data types
   (b) Getting started
   (c) Templates that build up ILUPACK
   (d) Components from other packages
4. Numerical results
5. Conclusions

supported by the DFG research center “Mathematics for key technologies” in Berlin
\[ Ax = b \]

**INCOMPLETE LU DECOMPOSITION**

\( k \) steps of an incomplete \( LU \) decomposition

\[
A = \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ L_E & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix} + E_k,
\]

where \( L_B, U_B \) are unit diagonal, \( S_C \) remaining matrix ("approximate Schur complement").

**INVERSE ERROR**

For preconditioning we have to apply \( L_k^{-1} A U_k^{-1} \).

\[ \rightarrow F_k := L_k^{-1} E_k U_k^{-1} \text{ "inverse error"} \]

Problem. \( \| L_k^{-1} \| \) and \( \| U_k^{-1} \| \) may become very large \( \Rightarrow \) pivoting recommended.
CONSEQUENCES

Prescribe a bound $\kappa$ and control growth $\|L_k^{-1}\|, \|U_k^{-1}\| \leq \kappa$ by diagonal pivoting.

FACTOR OR SKIP STRATEGY

At step $k$: Rows/columns that exceed the prescribed bound are pushed to the end
SKETCH OF THE TEMPLATES

1. Compute a static partial reordering

\[ A \rightarrow P^\top A Q = \begin{pmatrix} B & F \\ E & C \end{pmatrix} \]

2. Factor \( P^\top A Q \) and control \( \|L_k^{-1}\|, \|U_k^{-1}\| \leq \kappa \).

\[ P^\top A Q \rightarrow \hat{P}^\top A \hat{Q} = \begin{pmatrix} \hat{B} & \hat{F} \\ \hat{E} & \hat{C} \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ L_E & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix} \]

3. Apply strategy recursively to the approximate Schur complement \( S_C \) (multilevel scheme)

\[ A \overset{1}{\rightarrow} P^\top A Q \overset{2}{\rightarrow} \hat{P}^\top A \hat{Q} \overset{3}{\rightarrow} \begin{pmatrix} L_B \cdot D_B \cdot U_B & F \\ E & S_C \end{pmatrix} \]

\[ S_C \overset{1}{\rightarrow} P_S^\top S_C Q_S \overset{2}{\rightarrow} \hat{P}_S^\top S_C \hat{Q}_S \overset{3}{\rightarrow} \begin{pmatrix} L_S \cdot D_S \cdot U_S & F_S \\ E_S & T \end{pmatrix} \]

\[ \ldots \]
Inverse–Based Multilevel ILUs

\[ A \rightarrow P^\top AQ \rightarrow \hat{P}^\top \hat{A}Q \rightarrow \left( \begin{array}{c|c|c|c} L & D & U & F \\ \hline E & & & \\ \hline & S & & \end{array} \right) \]
Names and Formats

- ILUPACK currently supports real single/double precision and complex single/double precision.

- Apart from the arithmetic ILUPACK distinguishes between two classes of matrices, namely general matrices and symmetric (Hermitian) positive definite matrices.

- Most commands require two prefixes, one character for the arithmetic and an additional three–character prefix for the class.

  - S: real single precision
  - D: real double precision
  - C: complex single precision
  - Z: complex double precision
  - GNL: general matrices
  - SPD: real symmetric positive definite matrices
  - HPD: complex Hermitian positive definite matrices

- Matrices are stored in SPARSKIT format (compressed sparse row format), indices starting with 1, FORTRAN–style.

- Symmetric matrices. Let $D$ be the diagonal part of $A$ and $A = D + U + U^T$, then $D + U$ are stored.
Examples

- A command that computes the incomplete $LU$ factorization of $A$ is available as

  General Case
  \[
  \text{S GNLAMGfactor(...)} \quad \text{D GNLAMGfactor(...)} \quad \text{C GNLAMGfactor(...)} \quad \text{Z GNLAMGfactor(...)}
  \]

  Symmetric Positive Definite Case
  \[
  \text{S SPDAMGfactor(...)} \quad \text{D SPDAMGfactor(...)} \quad \text{C HPDAMGfactor(...)} \quad \text{Z HPDAMGfactor(...)}
  \]

- Compressed sparse row format.

\[
\begin{pmatrix}
-0.1 & 0.2 & 0 \\
0 & 1.1 & 0 \\
0 & 0 & 2.3
\end{pmatrix}
\rightarrow
\begin{align*}
\text{ia} & | 1 & 3 & 4 & 5 \\
\text{ja} & | 1 & 2 & 2 & 3 \\
\text{a} & | -0.1 & 0.2 & 1.1 & 2.3
\end{align*}
\]

\[
\begin{pmatrix}
3.1 & -0.2 & 0 \\
-0.2 & 1.1 & 0.4 \\
0 & 0.4 & 2.3
\end{pmatrix}
\rightarrow
\begin{align*}
\text{ia} & | 1 & 3 & 5 & 6 \\
\text{ja} & | 1 & 2 & 2 & 3 & 3 \\
\text{a} & | 3.1 & -0.2 & 1.1 & 0.4 & 2.3
\end{align*}
\]
**Datatypes**

- **amat A** (i.e. **Smat**, **Dmat**, **Cmat** or **Zmat**). Sparse matrices:
  - `A.nr, A.nc`: number of rows and columns
  - `A.ia, A.ja, A.a`: pointer, indices and numerical values in SPARSKIT sparse row format.

- **AMGlevelmat PRE**. Multilevel preconditioner, double linked list, each level:
  - `PRE.n, PRE.nB`: total size and size of the leading block
  - `PRE.LU`: sparse matrix containing the ILU.
  - `PRE.LUperm`: permutation array (in case of ILUTP)
  - `PRE.E, PRE.F`: submatrices needed for multilevel ILU parts `E` and `F`.
  - `PRE.p, PRE.invq`: row and inverse column permutation
  - `PRE.rowscal, PRE.colscal`: numerical values for row and column scaling
  - `PRE.prev, PRE.next`: pointers to the previous and next level

- **ILUPACKPARAM param**. Parameters to control miscellaneous ILUPACK routines:
  - `param.ipar[40], param.fpar[40]`: integer and double parameters. many, many switches!
  - `param.ibuff, param.dbuff`: integer and numerical value buffers.
// init param with the default parameters
AMGinit(A, &param);

// get some of the most common parameters and overwrite them
AMGgetparams(param, &flags, &fill, droptols, &kappa, &restol,
             &max_it, &nrestart);

// main drop tolerance and elbow space measured by nnz(LU)/nnz(A)
droptols[1]=5e-3; fill=5.0;

// rewrite the updated parameters
AMGsetparams(&param, flags, fill, droptols, kappa, restol,
             max_it, nrestart);

// choose three permutation functions, initial/regular/final
permp=permnull; permr=permrcm; permf=permpq;

// main factorization routine for the multilevel ILU
ierr=AMGfactor(&A, &PRE, &nlev, &param, permp,permr,permf);

// iterative solver to solve A*sol=rhs
ierr=AMGsolver(A, PRE, nlev, &param, sol, rhs);
**Template 1: Static reorderings and their usage in ILUPACK**

Orderings used to partially define a well-suited leading block \( B \)

\[
P^\top AQ = \begin{pmatrix} B & F \\ E & C \end{pmatrix}
\]

Three orderings can be prescribed

1. **Initial preprocessing.**
   Ordering and scaling ONLY applied to the initial system.

2. **Regular reordering.**
   Used for any subsequent level, as long as possible.

3. **Final pivoting.**
   If the regular reordering fails to process a suitable leading block \( B \), switch to the final pivoting strategy.
**Template 1: Example**

Initial System

![Graph showing the initial system with nz = 25407]
Initial System, reordered here: no permutation
Initial Level, reordered after application of $ILU$

such that $\|L^{-1}\|, \|U^{-1}\| \leq \kappa$
Level 2, initial

Template 1: Example
Level 2, reordered
here Reverse Cuthill-McKee
 TEMPLATE 1: EXAMPLE

Level 2, reordered after application of ILU such that $\|L^{-1}\|, \|U^{-1}\| \leq \kappa$
**Template 1: Example**

Level 3, initial

\[ nz = 15502 \]
Level 3, reordered
here Reverse Cuthill-McKee
Level 3, reordered after application of \( ILU \) such that \( \|L^{-1}\|, \|U^{-1}\| \leq \kappa \)
Level 4, initial
Level 4, reordered
here reverse Cuthill-McKee
Level 4, reordered after application of \( ILU \) such that \( \|L^{-1}\|, \|U^{-1}\| \leq \kappa \).
Level 4, reordered again
here PQ (switched to final pivoting)
Level 4, reordered again after application of \( ILU \) such that \( \|L^{-1}\|, \|U^{-1}\| \leq \kappa \)
Template 1. Static reorderings

Supported GLOBAL orderings.

- RCM (Reverse Cuthill–McKee): banded systems, PDE-type problems.
- MMD (Multiple Minimum Degree) or AMF (Approximate Minimum Fill): almost symmetrically structured problems, not recommended for ILUs.
- ND (Nested Dissection).
- MC64 (from HSL): unstructured problems, improves diagonal dominance.

Included PARTIAL orderings.

- ddPQ [Saad ’03]. Permutations $P, Q$ as a compromise between diagonal dominance and fill. Applicable to all problems, in particular unstructured systems.
- INDSET (independent set, ARMS), symmetrically structured problems, PDE–type problems.
- FC (fine grid/coarse grid partitioning similar to Ruge/Stüben AMG), symmetrically structured problems, PDE–type problems.

Any ordering offers or can be combined with column and/or row scaling
TEMPLATE 1. static GLOBAL reordering

Initial System

Reordered by RCM (reverse Cuthill-McKee)
Template 1. static GLOBAL reordering
**Template 1. static GLOBAL reorderings**

Initial System

Reordered by MC64 (improves diagonal dominance)
TEMPLATE 1. static PARTIAL reordering

Initial System  
Reordered ddPQ

nz = 25407  
nz = 25407
Template 1. static PARTIAL reordering
AVAILABLE INTERFACES FOR REORDERINGS AND SCALINGS

- \texttt{permrcm, permmd, permamf, permnd, permmc64, permpq, permindset, permfc, permpp, permnull.}

E.g. \texttt{DGNL permpq} is associated with the ddPQ ordering.

- As nonsymmetric orderings, \texttt{permpq} and \texttt{permmc64} are only available in the \texttt{GNL} case.

- Conversely \texttt{permpp} is a symmetric variant that is only available in the \texttt{SPD,HPD} cases.

- routines perform scaling and permutation in one step (starting with scaling)

- ILUPACK allows to include custom permutation routines.

- any permutation routines simply needs to follow the fixed parameter list
**Prototypes**

```
perm... (mat A, FLOAT *prowscale, FLOAT *pcolscale, 
int *p, int *invq, int *nB, ILUPACKPARAM *param)
```

- **mat A** is a matrix in sparse row format, A will be rescaled by *prowscale, *pcolscale!
- **FLOAT** *prowscale, *pcolscale are vectors that return row/column scalings.
- **FLOAT** may either be float, double, complex or doublecomplex (defined in ilupack.h).
- **int *p, *invq** provide row and inverse column permutation vectors (starting with 1 not with 0!!!)
- **int *nB** returns the size of leading diagonal block B
- **ILUPACKPARAM *param** is a structure of parameter settings. It covers switches for scalings.
**Template 2. PILUC — Inverse-based ILU that controls $\|L^{-1}\|, \|U^{-1}\|$**

- ILU (Crout–version)

- $\|L_k^{-1}\|, \|U_k^{-1}\| \leq \kappa$ controlled by diagonal pivoting

- $\|L_k^{-1}\|, \|U_k^{-1}\|$ estimated [Cline, Moler, Stewart, Wilkinson ’77], [B. ’03]
PILUC AS PART OF ILUPACK

1. FORTRAN77 core routine, offers all–in–one solution, e.g.
   (a) partial incomplete $LU$ factorization $L_B D_B U_B$
   (b) different versions of approximate Schur complement $S_C$
   (c) diagonal pivoting (factor or skip strategy)
   (d) estimates of the inverse factors
   (e) diagonal compensation

2. MPILUC, variant of PILUC. If

   \[
   \tilde{P}^\top A \tilde{Q} = \begin{pmatrix} B & F_1 \\ E_1 & C_{11} \end{pmatrix} \begin{pmatrix} F_2 \end{pmatrix} \approx \begin{pmatrix} L_B & 0 \\ L_{E_1} & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix}
   \]

   has been computed, MPILUC tries to continue with $S_C$ without switching to a new level.

   Repeated multiple application of MPILUC leads to

   \[
   \tilde{P}^\top A \tilde{Q} = \begin{pmatrix} B & F_1 \\ E_1 & C_{11} \end{pmatrix} \begin{pmatrix} F_2 \end{pmatrix} \approx \begin{pmatrix} L_B & 0 \\ L_{E_1} & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & D_{C_{11}} \end{pmatrix} \begin{pmatrix} U_B & U_{F_1} & U_{F_2} \\ 0 & I & U_{C_{12}} \end{pmatrix}
   \]
**Template 3. Multilevel scheme**

\[
(\hat{P}^\top A\hat{Q})^{-1} = \begin{pmatrix} B & F \\ E & C \end{pmatrix}^{-1} \approx \begin{pmatrix} \hat{B}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -\hat{B}^{-1}F \\ I \end{pmatrix} S_C^{-1} \begin{pmatrix} -E\hat{B}^{-1} & I \end{pmatrix}, \text{ where } \hat{B} = L_B D_B U_B
\]

Recursive application to \(S_C \rightarrow \text{AMG}\), concept of ARMS [Saad, Suchomel'99], i.e. skip \(L_E, U_F\).

**Approximate Schur complements**

1. \(S_C = C - L_E D_B U_F\)
   
   use analogous approximation scheme for \(B \approx L_B D_B U_B\) \hspace{1cm} (S–version)

2. \(S_C = \begin{bmatrix} -L_E L_B^{-1} & I \end{bmatrix} A \begin{bmatrix} -U_B^{-1} U_F \\ I \end{bmatrix}\)
   
   use analogous approximation scheme for \(B \approx L_B D_B U_B\) \hspace{1cm} (T–version)

3. Use S–version for the computation of \(B \approx L_B D_B U_B\)
   Use T–version for the computation of \(S_C\) \hspace{1cm} (M–version) (default)
ILUPACK main AMG driver

- main factorization routine uses PILUC and MPILUC as templates and builds the multilevel structure around them.

- three permutation/scaling routines are supported (see previous slides)

- Three different versions of approximate Schur complements are supported (S–, M– and T–version). M–version is default.

- After the computation of the approximate Schur complement the parts $L_E$ and $U_F$ are discarded.

- if the fill exceeds a certain amount, the algorithm switches to full-matrix processing.

- if static final pivoting (3-rd permutation) fails, ILUTP with a small drop tolerance is taken (happens rarely in practice).
**Prototypes**

```
AMGfactor(mat *A, AMGlevelmat *PRE, int *nlev,
           ILUPACKPARAM *param,
           int (*permp)(...), int (*permr)(...), int (*permf)(...))
```

- `mat *A` is a source matrix in sparse row format, A will be rescaled!
- `AMGlevelmat *PRE` is multilevel preconditioner in a linked list
- `int *nlev` total number of levels
- `ILUPACKPARAM *param` is a structure of parameter settings. It controls several parts of the preconditioner setup.
- `int (*permp)(...), (*permr)(...), (*permf)(...)` three reordering/scaling routines for initial preprocessing, regular reordering and final pivoting.
**Template 4. Iterative Solvers**

- Multilevel ILU solver
- Iterative solvers GMRES, PCG
- Standard driver routine

**Prototypes**

\[
\text{AMGsol(AMGlevelmat } \text{PRE, int } *\text{nlev, FLOAT } *\text{rhs, FLOAT } *\text{sol, FLOAT } *\text{buff)}
\]

- AMGlevelmat *PRE is multilevel preconditioner in a linked list
- int *nlev total number of levels
- FLOAT *rhs, *sol, buff right hand side, solution and buffer.
**Prototypes**

```c
AMGsolver(mat A, AMGlevelmat PRE, int nlev,
           ILUPACKPARAM *param, FLOAT *sol, FLOAT *rhs)
```

- `mat A` is a matrix in sparse row format, $A$ already rescaled!
- `AMGlevelmat PRE` is multilevel preconditioner in a linked list
- `int nlev` total number of levels
- `ILUPACKPARAM *param` is a structure of parameter settings. It covers switches for scalings.
- `FLOAT *sol, *rhs` solution and right hand side.

**Comments**

- Default solvers are GMRES and PCG
- Initialization to standard settings supported by auxiliary routines
Preconditioning software package (C/FORTRAN 77) using inverse–based multilevel ILUs
- Iterative solvers GMRES, FGMRES
- PCG adapted from SPARSKIT–CG
- Preconditioners ILUTP and ILUT, now based on a binary search tree
- Preconditioners and solvers adapted with respect to complex arithmetic (and BLAS)
- Miscellaneous tools adapted (scaling, readmtc)

- Orderings indset and ddPQ
- Multilevel strategy ($L_E, U_F$ discarded)
- Miscellaneous tools adapted (matvec, spartran)
ILUPACK: Components

**BLAS**

- Iterative solvers GMRES, FGMRES and PCG now based on BLAS
- Miscellaneous tools as well

**Interfaces**

- SPARSPAK
- MC64
- Appr. Min. Fill

- Interfaces w.r.t. several reordering strategies provided
- MMD (multiple minimum degree), RCM (reverse Cuthill–McKee), ND (nested dissection)
  
- AMF (approximate minimum fill) by Patrick Amestoy
- MC64 (HSL, Harwell Subroutine Library)
**Iterative Solvers**

GMRES(30), PCG. Iteration stopped if $\|b - Ax^{(k)}\| \leq \sqrt{\text{eps}} \|b - Ax^{(0)}\|$

Iterative solver treated as failure, if a break down occurred or more than 500 steps were needed.

**Orderings**

PQ (PP resp.), RCM, MMD, AMF + row/column scaling

Matrices also separately preprocessed by MC64 and tested

**Preconditioners**

1. inverse–based multilevel ILU (referred as ILUPACK)
2. ILUTP
3. ILUC (single–level, inverse–based)
EXAMPLE. Matrix ATT/TWOTONE

Computation time

Memory requirement
Example. Matrix VAVASIS/AV41092
Almost no difference between $\kappa = 10$, $\kappa = 25$, $\kappa = 50$, $\kappa = 100$.

**64 LARGE TEST MATRICES, NUMBER OF SUCCESSFUL COMPUTATIONS**

**ILUPACK M–version**

**ILUPACK S–version**
Almost no difference between $\kappa = 10$, $\kappa = 25$, $\kappa = 50$, $\kappa = 100$.

64 LARGE TEST MATRICES, NUMBER OF SUCCESSFUL COMPUTATIONS

ILUPACK M–version

ILUTP + RCM
ILUPACK — dependence on inverse bound $\kappa$

$\kappa = 10, \kappa = 25, \kappa = 50, \kappa = 100$

64 LARGE TEST MATRICES, NUMBER OF SUCCESSFUL COMPUTATIONS

ILUPACK M–version

ILUTP + RCM + MC64
**Example.** cylindric shell problem

Tight bounds on $\|U^{-1}\| \leq \kappa$ implicitly yield FINE GRID/COARSE GRID selection.
Example. **cylindric shell problem**

Number of levels. $\kappa = 2, \kappa = 5, \kappa = 10, \kappa = 25, \kappa = 50, \kappa = 100$. 

![Graph showing number of levels depending on $\kappa$.](image-url)
1. ILUPACK offers inverse–based multilevel ILU preconditioners for a broad class of matrices

2. Main objective: inverse triangular factors are kept bounded

3. various reordering are incorporated or can easily be added

4. numerical experiments indicate robustness with respect to parameter tuning (time dependent problems, nonlinear equations)

ILUPACK being released by April/May, 2004

http://www.math.tu-berlin.de/ilupack/