# A Supernodal Approach to ILU with Partial Pivoting

**X. Sherry Li**

xsli@lbl.gov

Lawrence Berkeley National Laboratory

**Meiyue Shao**

Umeå University, Sweden

Sparse Days 2010 at CERFACS

June 15-17, 2010

# Outline

- **Supernodal LU factorization (SuperLU)**
- **Supernodal ILUTP with adaptive dual dropping**
  - **Threshold dropping in supernode**
  - **Secondary dropping for memory concern**
- **Variants: Modified ILU (MILU)**
- **Extensive experiments, comparison with other approaches**
  - **232 matrices**
- **Software available in SuperLU 4.0**

# ILU preconditioner

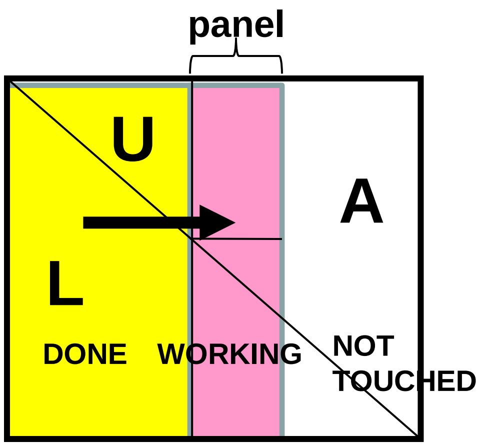
- **Structure-based dropping: level-of-fill**
  - **ILU(0), ILU(k)**
  - **Rationale: the higher the level, the smaller the entries**
  - **Separate symbolic factorization to determine fill-in pattern**
- **Value-based dropping: drop truly small entries**
  - **Fill-in pattern must be determined on-the-fly**

- **ILUTP[Saad]: among the most sophisticated, and (arguably) robust; implementation similar to direct solver**
  - **"T" = threshold, "P" = pivoting**
  - **Dual dropping: ILUTP(p,tau)**
    1) **Remove elements smaller than tau**
    2) **At most p largest kept in each row or column**

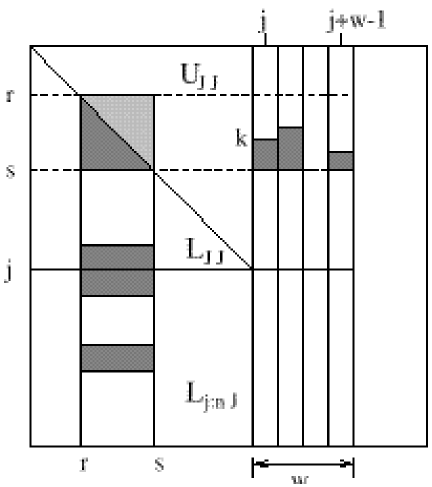
# SuperLU [Demmel/Eisenstat/Gilbert/Liu/Li ’99]

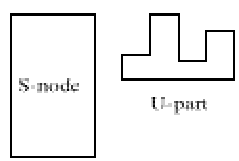
http://crd.lbl.gov/~xiaoye/SuperLU

- **Left-looking, supernode**




1. Sparsity ordering of columns
   use graph of A’*A
2. Factorization
   For each panel …
   - Partial pivoting
   - Symbolic fact.
   - Num. fact. (BLAS 2.5)
3. Triangular solve



# Primary dropping rule: S-ILU(tau)

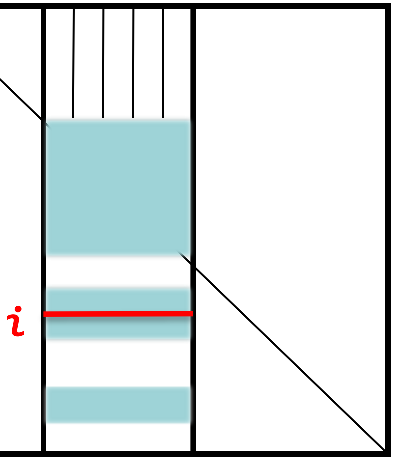
- **Similar to ILUTP, adapted to supernode**
  1. **U-part:**

     $$\text{If } |u_{ij}| < \tau \cdot \|A(:,j)\|_\infty, \text{ then set } u_{ij} = 0$$

  2. **L-part: retain supernode**

$$\text{Supernode } L(:,s:t), \text{ if } RowSize(i,s:t) < \tau, \text{ then set the entire } i\text{-th row to zero}$$

- **Remarks**
  1) **Delayed dropping**
  2) **Entries computed first, then dropped.**
     **May not save many flops compared to LU**
  3) **Many choices for RowSize() metric**



# Dropping in supernode

Supernode $L(:, s:t)$, if $RowSize(i, s:t) < \tau$, then set the entire $i$-th row to zero

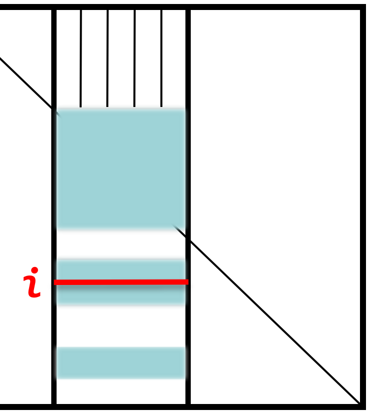
**RowSize() metric: let m = t-s+1, supernode size**

**1) Mean:** $RowSize(x) = \frac{\|x\|_1}{m}$ [ used by Gupta/George for IC ]

**2) Generalized-mean:** $RowSize(x) = \frac{\|x\|_2}{\sqrt{m}}$

**3) Infinity-norm:** $RowSize(x) = \|x\|_\infty$

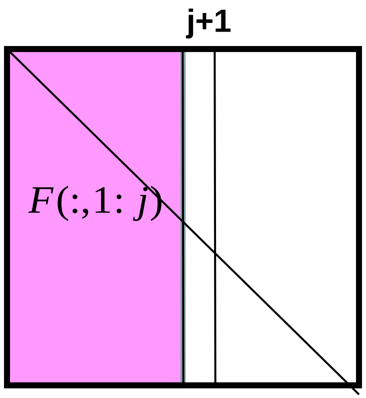
**Every dropped entry in L would also be dropped in a column-wise algorithm**




Since $\frac{\|x\|_1}{m} \le \frac{\|x\|_2}{\sqrt{m}} \le \|x\|_\infty$, 1) is most aggressive, 3) is conservative

# Secondary dropping rule: S-ILU(p,tau)

- **Control fill ratio with a user-desired upper bound $\gamma$**
- **Earlier work, column-based**
  - **[Saad]: ILU(p, tau), at most p largest nonzeros allowed in each row**
  - **[Gupta/George]: p adaptive for each column** $p(j) = \gamma \cdot nnz(A(:, j))$
    **May use interpolation to compute a threshold function, no sorting**




- **Our new scheme is area-based**
  - Look at fill ratio from column 1 up to j:
    $$fr(j) = nnz(F(:,1:j)) / nnz(A(:,1:j))$$
  - **Define adaptive upper bound function** $f(j) \in [1, \gamma]$
    If $fr(j)$ exceeds $f(j)$, retain only p largest, such that $fr(j) \le f(j)$
  - **More flexible, allow some columns to fill more, but limit overall**

# Experiments: GMRES + ILU

- **Use restarted GMRES with ILU as a right preconditioner**

  Solve $PA(\tilde{L}\tilde{U})^{-1} y = Pb$
- **Size of Krylov subspace set to 50**
- **Initial guess is a 0-vector**
- **Stopping criteria:** $\|b\text{-}Ax_k\|_2 \le 10^{-8}\|b\|_2$ and $\le 500$ iterations

- **232 unsymmetric test matrices; RHS is generated so the true solution is 1-vector**
  - **227 from Univ. of Florida Sparse Matrix Collection**

    **dimension 5K – 1M, condition number below $10^{15}$**
  - **5 from MHD calculation in tokmak design for plasma fusion energy**
- **AMD Opteron 2.4 GHz quad-core (Cray XT5), 16 GBytes memory, PathScale pathcc and pathf90 compilers**

# Compare with column C-ILU(p, tau)

- **C-ILU: set maximum supernode size to be 1**
- **Maxsuper = 20, gamma = 10, tau = 1e-4**

| | Factor construction | | | | GMRES | | Total Sec. |
|---|---|---|---|---|---|---|---|
| | Fill-ratio | S-node Cols | Flops ($10^9$) | Fact. sec. | Iters | Iter sec. | |
| | $RowSize(x) = \|\| x \|\|_2 / \sqrt{m}$ | | | | 138 matrices succeeded | | |
| S-ILU | 4.2 | 2.8 | 7.60 | 39.69 | 21.6 | 2.93 | 42.68 |
| C-ILU | 3.7 | 1.0 | 2.65 | 65.15 | 20.0 | 2.55 | 67.75 |
| | $RowSize(x) = \|\| x \|\|_\infty$ | | | | 134 matrices succeeded | | |
| S-ILU | 4.2 | 2.7 | 9.45 | 54.44 | 20.5 | 3.4 | 57.0 |
| C-ILU | 3.6 | 1.0 | 2.58 | 74.10 | 19.8 | 2.88 | 77.04 |

# Supernode vs. column

- **Less benefit using supernode compared to complete LU**
  - **Better, but Less than 2x speedup**

- **What go against supernode:**
  - **The average supernode size is smaller than in LU.**
  - **The row dropping rule in S-ILU tends to leave more fill-ins and operations than C-ILU … we must set a smaller "maxsuper" parameter.**

    **e.g., 20 in ILU vs. 100 in LU**

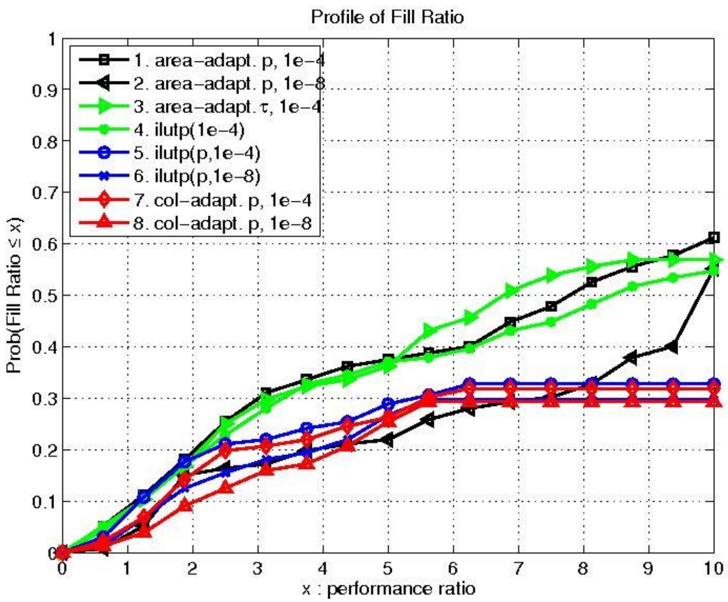
# S-ILU for extended MHD calculation (fusion)

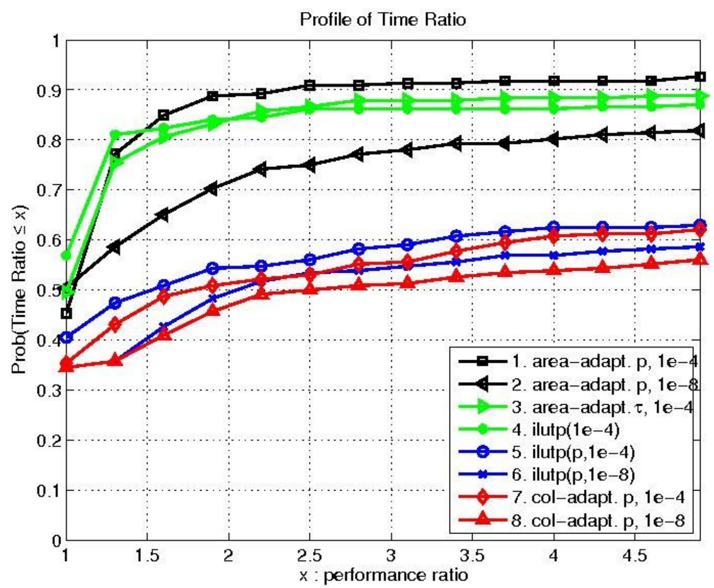
- **ILU parameters:** $\tau = 10^{-4}, \gamma = 10$
- **Up to 9x smaller fill ratio, and 10x faster**

| Problems | order | Nonzeros (millions) | ILU time | ILU fill-ratio | GMRES time | GMRES iters | SuperLU time | SuperLU fill-ratio |
|---|---|---|---|---|---|---|---|---|
| matrix31 | 17,298 | 2.7 m | 8.2 | 2.7 | 0.6 | 9 | 33.3 | 13.1 |
| matrix41 | 30,258 | 4.7 m | 18.6 | 2.9 | 1.4 | 11 | 111.1 | 17.5 |
| matrix61 | 66,978 | 10.6 m | 54.3 | 3.0 | 7.3 | 20 | 612.5 | 26.3 |
| matrix121 | 263,538 | 42.5 m | 145.2 | 1.7 | 47.8 | 45 | fail | - |
| matrix181 | 589,698 | 95.2 m | 415.0 | 1.7 | 716.0 | 289 | fail | - |

# S-ILU comprehensive tests

- **Performance profile of fill ratio – fraction of the problems a solver could solve within a fill ratio of X**
- **Performance profile of runtime – fraction of the problems a solver could solve within a factor X of the best solution time**




- **Conclusion:**
  - **New area-based heuristic is much more robust than column-based one**
  - **ILUTP(tau) is reliable; but need secondary dropping to control memory**

# Other features in the software

- **Zero pivot ?**

if $u_{jj} = 0$, set it to $\hat{\tau}(j) \| A(:, j) \|_\infty$

$\hat{\tau}(j) = 10^{-2(1-j/n)}$, adaptive, increasing with $j$, so $U$ is not too ill-conditioned

- **Threshold partial pivoting**

- **Preprocecssing with MC64** [Duff-Koster]
  - **With MC64, 203 matrices converge, avg. 12 iterations**
  - **Without MC64, 170 matrices converge, avg. 11 iterations**

- **Modified ILU (MILU)**
  - **Reduce number of zero pivots**

# Modified ILU (MILU)

- **Reduce the effect of dropping: for a row or column, add up the dropped elements to the diagonal of U**

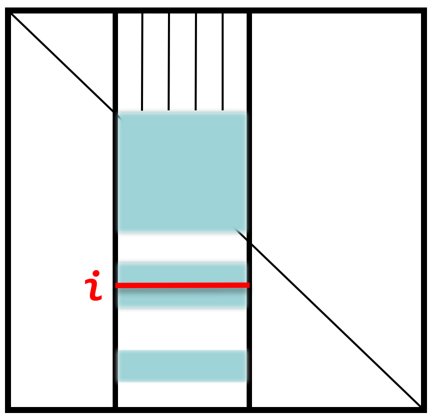
- **Classical approach has the following property:**
  - **Maintain row-sum for a row-wise algorithm:** $\tilde{L}\tilde{U}e = Ae$
  - **Maintain column-sum for a column-wise algorithm:** $e^T\tilde{L}\tilde{U} = e^T A$

- **Another twist … proposed for MIC**

  **Maintain** $LUx = Ax + \Lambda D x$ **for any x, using diagonal perturbations**
  - **Dupont-Kendall, Axelsson-Gustafsson, Notay (DRIC)**
  - **Reduce condition number of elliptic discretization matrices by order of magnitude (i.e., from $O(h^{-2})$ to $O(h^{-1})$ )**

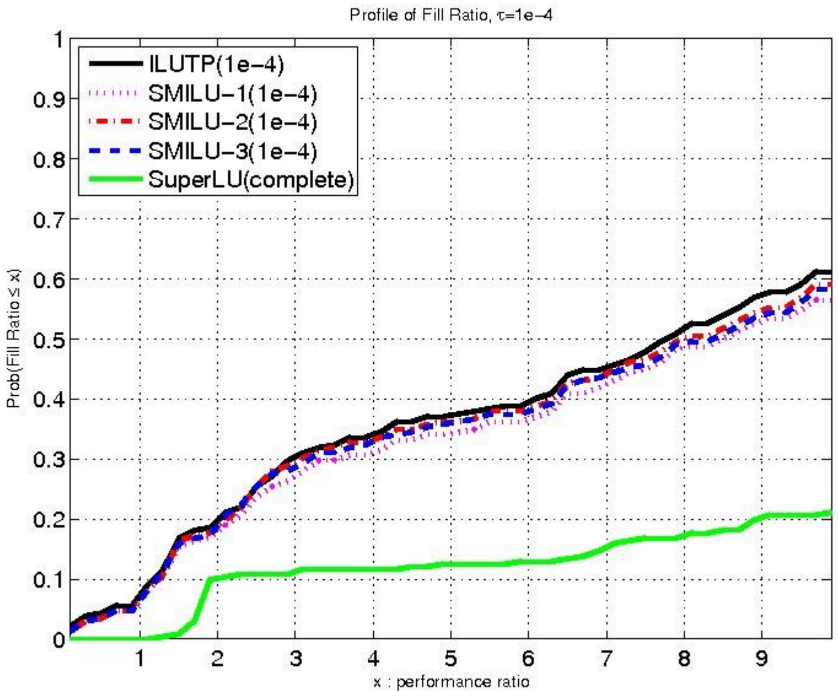
# MILU algorithm

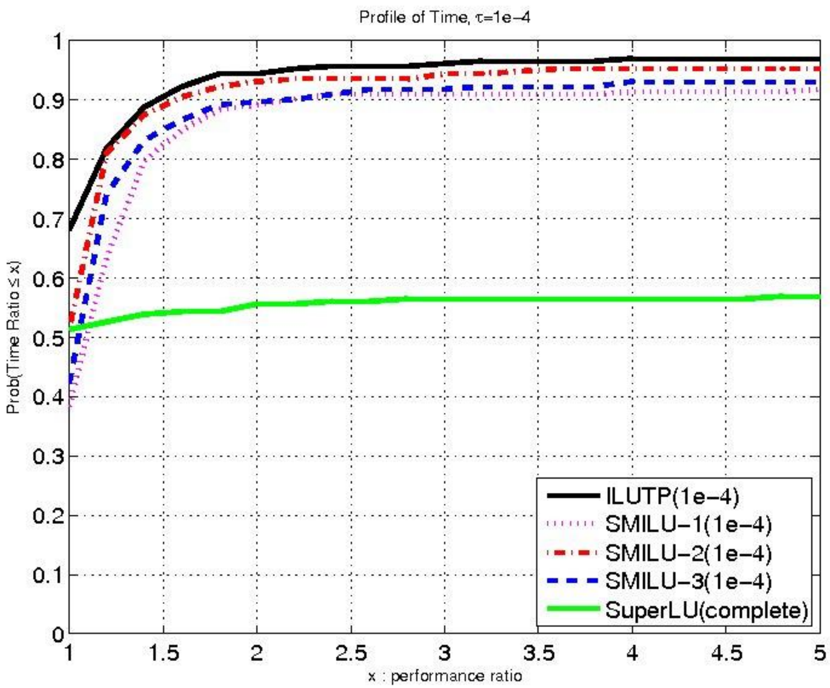
- **C-MILU:**
  1) Obtain filled column F(:, j), drop from F(:, j)
  2) Add up the dropped entries: $s = \sum_{dropped} f_{ij}$ ; Set $f_{ij} := f_{ij} + s$
  3) Set U(1:j, j) := F(1:j, j); L(j+1:n, j) := F(j+1: n, j) / F(j, j)
- **S-MILU:**
  1) First drop from U, $s = \sum_{dropped} U(:,j)$
     Set $u_{jj} := f_{jj} + s$;
  2) When a supernode is formed in L, drop more rows in L, add the dropped entries to diagonal of U




- **Our variants:**
  - S-MILU-1: $s = \sum_{dropped} U(:,j)$
  - S-MILU-2: $s = | \sum_{dropped} U(:,j) |$, $u_{jj} := f_{ij} + \text{sign}(f_{jj})*s$
  - S-MILU-3: $s = \sum_{dropped} |U(:,j)|$, $u_{jj} := f_{ij} + \text{sign}(f_{jj})*s$

# Modified ILU (MILU)



# Another look at MILU – 232 matrices

| | Converge | Slow | Diverge | Zero pivots | Average iterations |
|---|---|---|---|---|---|
| S-ILU | 133 | 51 | 46 | 1737 | 35 |
| S-MILU-1 | 125 | 72 | 33 | 1058 | 34 |
| S-MILU-2 | 127 | 71 | 31 | 296 | 30 |
| S-MILU-3 | 129 | 73 | 28 | 289 | 33 |

# Compare with the other preconditioners

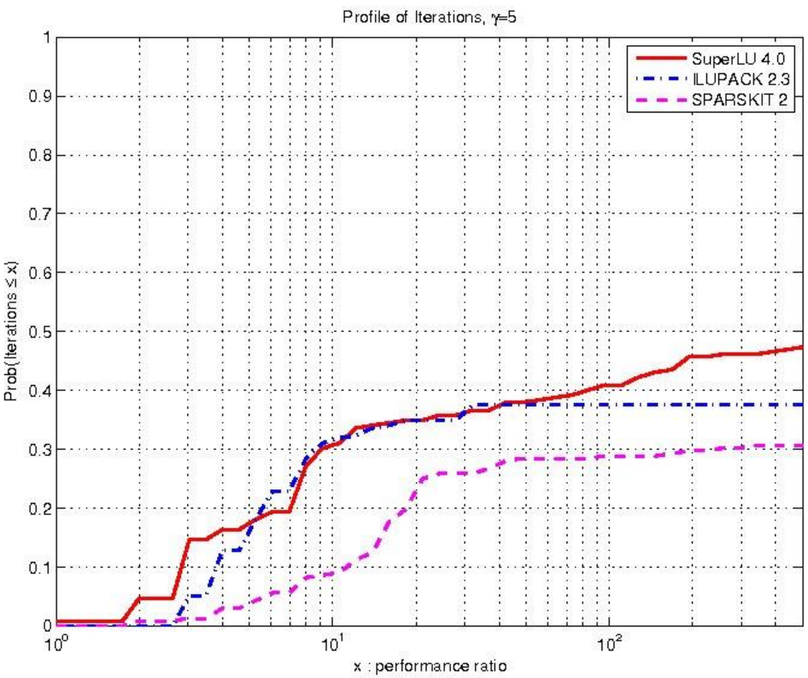
- **SPARSKIT** [saad] **: ILUTP, closest to ours**
  - **Row-wise algorithm, no supernode**
  - **Secondary dropping uses a fixed p for each row**
- **ILUPACK** [Bolhoefer et al.] : **very different**
  - **Inverse-based approach: monitor the norm of the k-th row of $L^{-1}$, if too large, delay pivot to next level**
  - **Multilevel: restart the delayed pivots in a new level**
- **ParaSails** [Chow]: **very different**
  - **Sparse approximate inverse: $M \sim A^{-1}$**
  - **Pattern of powers of sparsified A as the pattern of M**

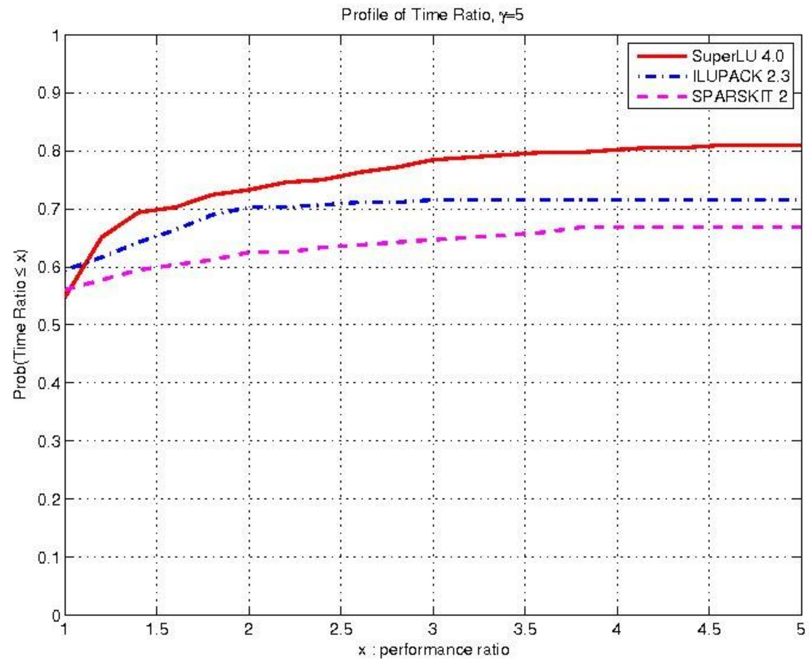
    **"thresh" to sparsify A, "nlevels" to keep level of neighbors**
  - **Default setting: thresh = 0.1, nlevels = 1**

    **Only 39 matrices converge, 62 hours to construct M, 63 hours after GMRES**
  - **Smaller thresh and larger nlevels help, but too expensive**

# Compare with SPARSKIT, ILUPACK

- **S-ILU:** $\tau = 10^{-4}, \gamma = 5,$ diag_thresh $\eta = 0.1$
- **ILUPACK :** $\tau = 10^{-4},\ \gamma = 5,\ \nu = 5$
- **SPARSKIT :** $\tau = 10^{-4},\ \gamma = 5,\ p = \gamma \cdot \frac{nnz}{n}$



# Comparison (cont) … a closer look …

- **S-ILU and ILUPACK are comparable: S-ILU is slightly faster, ILUPACK has slightly lower fill**
- **None of the preconditioners works for all problems … unlike direct methods**
- **They do not solve the same set of problems**
  - **S-ILU succeeds with 142**
  - **ILUPACK succeeds with 130**
  - **Both succeed with 100 problems**

- **Remark**

  **Two methods complimentary to one another, both have their place in practice**

# Summary of contributions

- **Supernode**
  - **Useful, but to less extend compared with complete LU**
- **Secondary dropping: area-based, adaptive-p, adaptive-tau**
  - **More reliable**
- **Empirical study of MILU**
  - **Limited success, disappointing in general**

# Final remarks

- **60-70% success with S-ILUTP for 232 matrices.**
  **When it works,  much more efficient than direct solver.**

- **Software**
  - **Available in serial SuperLU  V4.0,  June 2009**
  - **Same can be done for SuperLU_MT (left-looking, multicore)**

- **Scalable parallel  ILUTP?**
  - **How to do this with right-looking, multifrontal algorithms?**
    **e.g., SuperLU_DIST, MUMPS**