Batched Routines in Preconditioning

The Future of Incomplete Factorization Preconditioners

Hartwig Anzt, Thomas Huckle, Jack Dongarra

Workshop on Batched, Reproducible, and Reduced Precision BLAS
http://www.netlib.org/utk/people/JackDongarra/WEB-PAGES/Batched-BLAS-2016/
Innovative Computing Laboratory
University of Tennessee
May 18th – 19th, 2016

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b$, $A \in \mathbb{R}^{n \times n}$.

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)}^{*} \end{cases}$$

Many other choices possible for incomplete factorizations!

*Saad: **"Iterative Methods for Sparse Linear Systems (2nd Edition)".** *SIAM, 2003.*

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z$, $Ux = y$.

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z$, $Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$
$$y = M_L^{-1} z + (I - M_L^{-1} L) y$$

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\, A \in \mathbb{R}^{n\times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n\times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z\,,\, Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

Standard approach: **exact triangular solves** (trsv)

- Inherently sequential, level scheduling often provides little parallelism.

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b$, $A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n\times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z$, $Ux = y$.

For matrix splitting $L = M_L - \cancel{N_L}$ we get

$$y = M_L^{-1} z + \cancel{M_L^{-1} N_L} y$$

$$y = M_L^{-1} z + \underbrace{(I - M_L^{-1} L)}_{} y$$

$= 0$ for $M_L = L$, i.e. $(M_L^{-1} L = I)_{S^*}$ for $S^* = \mathbb{R}^{n \times n}$

Standard approach: **exact triangular solves** (trsv)

- Inherently sequential, level scheduling often provides little parallelism.

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b$, $A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z$, $Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + \underbrace{(I - M_L^{-1} L)}_{=0 \text{ for } M_L = L \text{, i.e. } (M_L^{-1} L = I)_{S^*} \text{ for } S^* = \mathbb{R}^{n \times n}} y$$

Standard approach: **exact triangular solves** (trsv)

- Inherently sequential, level scheduling often provides little parallelism.
    - **exact factorization + exact solve = Sparse direct solver**

ICL THE UNIVERSITY OF TENNESSEE KNOXVILLE

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\ A \in \mathbb{R}^{n\times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n\times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z,\ Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + (\underbrace{I - M_L^{-1} L}_{=0 \text{ for } M_L = L})y$$

$= 0$ for $M_L = L$, i.e. $(M_L^{-1} L = I)_{S^*}$ for $S^* = \mathbb{R}^{n\times n}$

Standard approach: **exact triangular solves** (trsv)

- Inherently sequential, level scheduling often provides little parallelism.
- Over-engineering for incomplete factorization preconditioners with $S \subsetneq \mathbb{R}^{n\times n}$?

Preconditioner benefit limited by ILU quality

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z,\, Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + \underbrace{(I - M_L^{-1} L)}_{\approx 0 \text{ for } (M_L^{-1} L = I)_{S^*} \text{ for some sparsity pattern } S^*} y$$

Use similar idea like in the factorization step: Approximate on some sparsity pattern!

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n\times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z\,,\, Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + \underbrace{(I - M_L^{-1} L)}_{\approx 0 \text{ for } (M_L^{-1} L = I)_{S^*} \text{ with } S^* = diag(L)} y$$

➡ Jacobi Iteration $y = D_L^{-1} z + \left(I - D_L^{-1} L\right) y$

-- More Krylov solver iterations may be needed

\+ Faster triangular solves (few SpMVs) can make solution process faster[1,2]

[1]Chow and Patel. **"Fine-grained Parallel Incomplete LU Factorization".** *In: SIAM J. on Sci. Comp. (2015).*
[2]Anzt, Chow, and Dongarra. **"Iterative Sparse Triangular Solves for Preconditioning".** *In: LNCS. 2015.*

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z$, $Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$
$$y = M_L^{-1} z + (\underbrace{I - M_L^{-1} L}_{\approx 0 \text{ for } (M_L^{-1} L = I)_{S^*} \text{ with } S^* = diag_B(L)})y$$

➡ Block Jacobi Iteration

-- Need inverse of diagonal blocks

\+ Good for problems with inherent block-structure (e.g. with FEM origin)[3]

[3]Chow and Scott. **"On the use of iterative methods and blocking for solving sparse triangular systems in incomplete factorization preconditioning".** *Rutherford-Appleton Technical Report RAL-P-2016-006, 2016.*

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b, A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z$, $Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + \underbrace{(I - M_L^{-1} L)}_{\approx 0} y$$

$$\approx 0 \text{ for } (M_L^{-1} L - I) \approx 0$$

$$\Leftrightarrow (M_L^{-1} L - M_L^{-1} M_L) \approx 0$$

$$\Leftrightarrow (M_L^{-1} (L - M_L)) \approx 0$$

$$\Leftrightarrow (M_L^{-1} (\underbrace{L M_L^{-1}} M_L - M_L)) \approx 0$$

$$L M_L^{-1} \approx I \text{ i.e. } (L M_L^{-1} = I)_{S^*} \text{ for some } S^*$$

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\ A \in \mathbb{R}^{n \times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n \times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z,\ Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + (\underbrace{I - M_L^{-1} L}_{\approx 0 \text{ for } (LM_L^{-1} = I)_{S^*} \text{ with, e.g. } S^* = spy(A)})y$$

Factorizations for Sparse Matrices

Goal: Find solution to sparse linear problem $Ax = b,\ A \in \mathbb{R}^{n\times n}$.

Compute factorization

$$(A = LU)_{\mathcal{S}} \text{ for some sparsity pattern } \mathcal{S} \begin{cases} \mathcal{S} = \mathbb{R}^{n\times n} & \text{exact fact.} \\ \mathcal{S} = spy(A) & \text{ILU(0)} \end{cases}$$

Preconditioner application involves solving triangular systems $Ly = z,\ Ux = y$.

For matrix splitting $L = M_L - N_L$ we get

$$y = M_L^{-1} z + M_L^{-1} N_L y$$

$$y = M_L^{-1} z + (\underbrace{I - M_L^{-1} L}_{\approx 0})y$$

≈ 0 for $(LM_L^{-1} = I)_{S^*}$ with, e.g. $S^* = spy(A)$

SpAI aproach: choose M_L^{-1} as sparse approximate inverse for L on pattern S^*

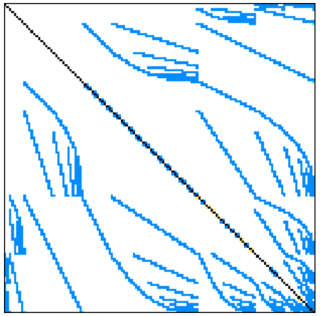
-- Need generation of SpAI matrix

+ One single SpMV for SpAI application

+ Flexibility in choice of the SpAI pattern (CA-Krylov methods...)

[4]Huckle, Anzt, Dongarra **"Parallel Preconditioning"**. *In: SIAM PP 2016*.

Example: ILU - BiCGSTAB

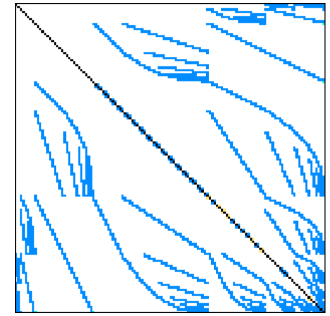


Test matrix: parabolic_fem (n=525,825 nnz=3,674,625)
Use RCM-reordering

BiCGSTAB Preconditioning	Iterations	Prec. setup time	Solver runtime	Total runtime
no	4,752			
ILU + cuSPARSE-trisolves	407			
ILU + Jacobi-trisolves (3s)	1,696			
ILU + Jacobi-trisolves (4s)	905			
ILU + Jacobi-trisolves (5s)	645			
ILU + SPAI (pattern L)	1,459			
ILU + SPAI (pattern L^2)	1,022			
ILU + SPAI (pattern L^3)	693			
ILU + SPAI (pattern L^4)	531			

NVIDIA K40 GPU, CUDA/cuSPARSE v7.0

Example: ILU - BiCGSTAB



Test matrix: parabolic_fem (n=525,825 nnz=3,674,625)
Use RCM-reordering

BiCGSTAB Preconditioning	Iterations	Prec. setup time	Solver runtime	Total runtime
no	4,752		8.21 s	
ILU + cuSPARSE-trisolves	407		43.72 s	
ILU + Jacobi-trisolves (3s)	1,696		13.01 s	
ILU + Jacobi-trisolves (4s)	905		8.64 s	
ILU + Jacobi-trisolves (5s)	645		7.41 s	
ILU + SPAI (pattern L)	1,459		4.65 s	
ILU + SPAI (pattern L^2)	1,022		4.13 s	
ILU + SPAI (pattern L^3)	693		4.38 s	
ILU + SPAI (pattern L^4)	531		3.70 s	

~ 3x

NVIDIA K40 GPU, CUDA/cuSPARSE v7.0

ICL THE UNIVERSITY OF TENNESSEE KNOXVILLE

Generating SpAI for triangular factor

Goal is to find M_L^{-1} with $(LM_L^{-1} = I)_{S^*}$ for $S^* = spy(A)$

Generating SpAI for triangular factor

Goal is to find M_L^{-1} with $(LM_L^{-1} = I)_{S^*}$ for $S^* = spy(A)$

➡ Choose $spy(M_L^{-1}) = spy(L)$ and solve $L \cdot M_L^{-1}(i,:) = e_i \; \forall \; i = 1 \ldots n$

Generating SpAI for triangular factor

Goal is to find M_L^{-1} with $(LM_L^{-1} = I)_{S^*}$ for $S^* = spy(A)$

Choose $spy(M_L^{-1}) = spy(L)$ and solve $L \cdot M_L^{-1}(i,:) = e_i \; \forall \; i = 1 \ldots n$

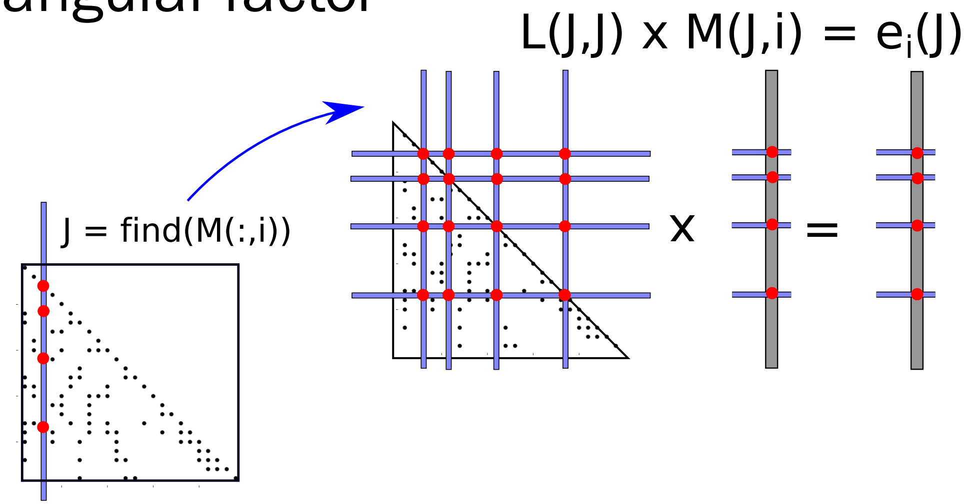
Use M for M_L^{-1}:

```
[L U] = ILU(A);
M = spones(L);                  % also possible L^2, L^3 ...
n=length(L);
for i=1:n
   J = find(M(:,i));            % determine nonzero entries
   generate L(J,J);             % small triangular system
   solve L(J,J) M(J,i) = e_i(J);  % one trsv per column of M
   insert M(J,i) into M;        % insert data into SpAI matrix M
end
```

Same strategy for SpAI matrix approximating upper triangular factor.

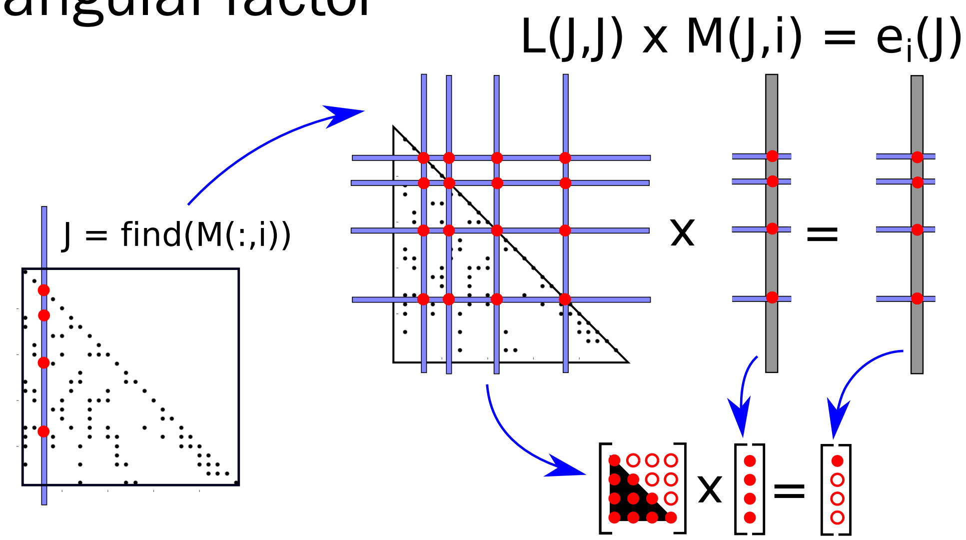
Generating SpAI for triangular factor

```
for i=1:n
   J = find(M(:,i));
   generate L(J,J);
   solve L(J,J) M(J,i) = e_i(J);
   insert M(J,i) into M;
end
```



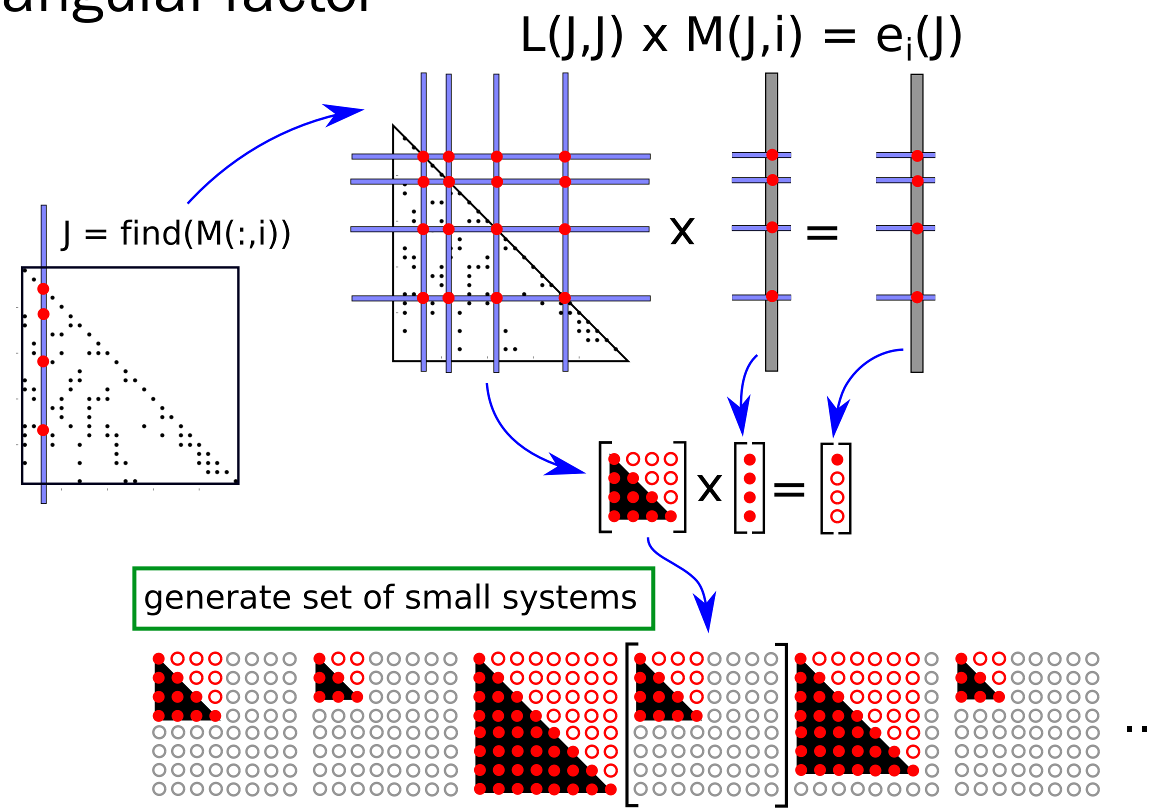
Generating SpAI for triangular factor

```
for i=1:n
   J = find(M(:,i));
   generate L(J,J);
   solve L(J,J) M(J,i) = e_i(J);
   insert M(J,i) into M;
end
```



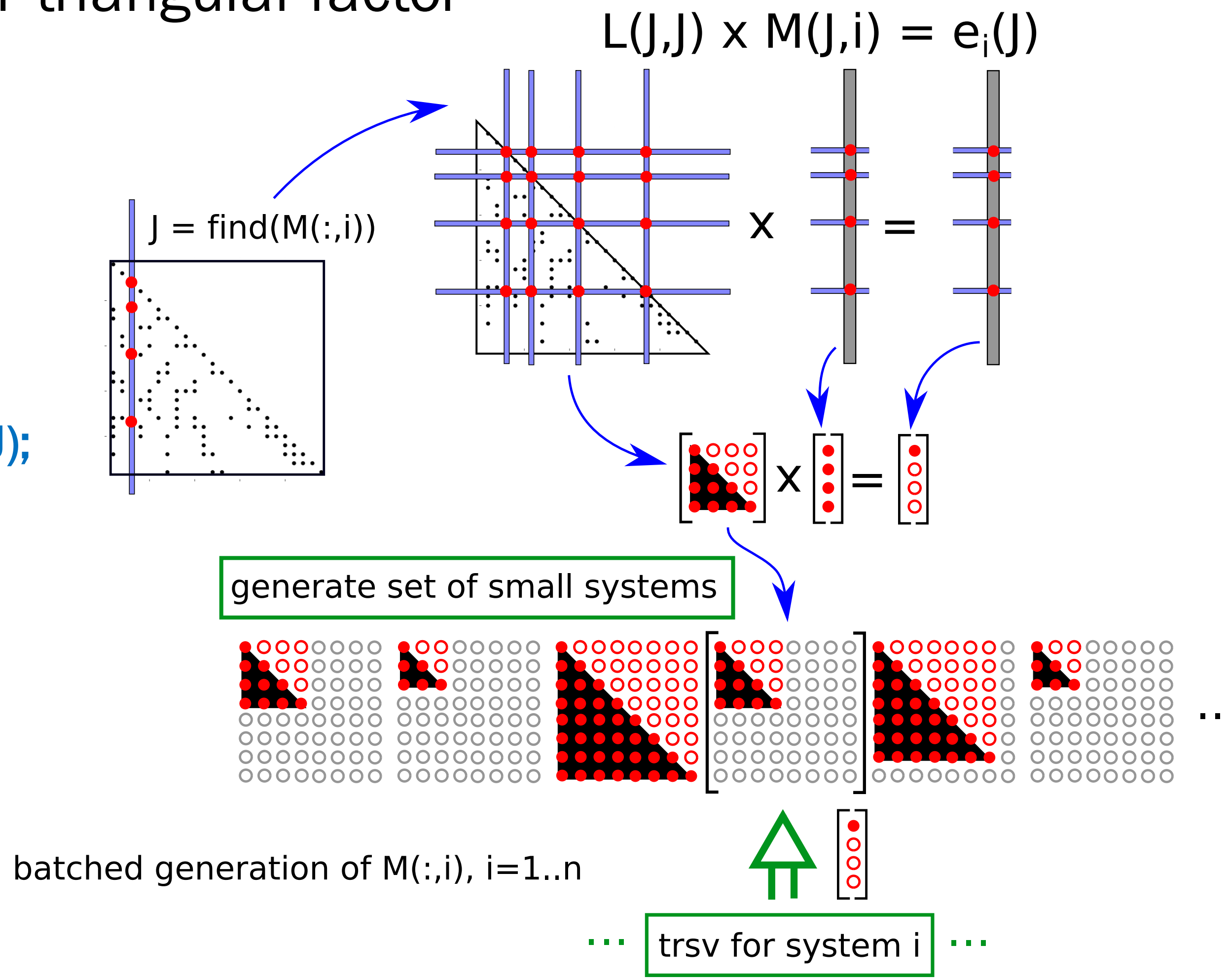
Generating SpAI for triangular factor

```
for i=1:n
   J = find(M(:,i));
   generate L(J,J);
   solve L(J,J) M(J,i) = e_i(J);
   insert M(J,i) into M;
end
```



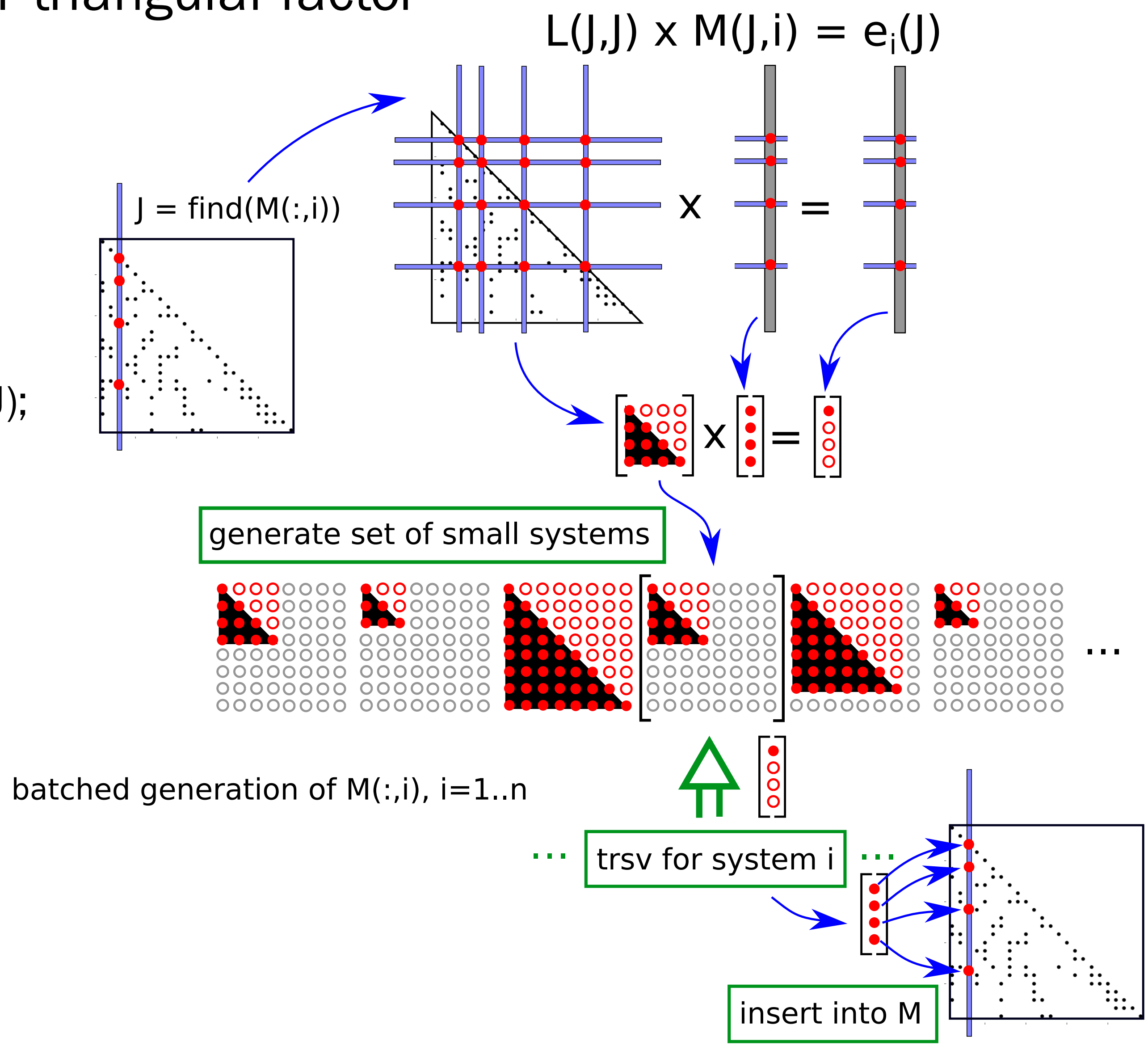
Generating SpAI for triangular factor

for i=1:n
  J = find(M(:,i));
  generate L(J,J);
  solve L(J,J) M(J,i) = e_i(J);
  insert M(J,i) into M;
end



Generating SpAI for triangular factor

```
for i=1:n
   J = find(M(:,i));
   generate L(J,J);
   solve L(J,J) M(J,i) = e_i(J);
   insert M(J,i) into M;
end
```



Generating SpAI for triangular factor

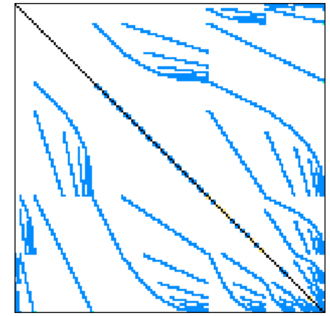
```
for i=1:n
   J = find(M(:,i));
   generate L(J,J);
   solve L(J,J) M(J,i) = e_i(J);
   insert M(J,i) into M;
end
```

Four Batched Routines:

- Find the locations in each row
  - store size information for small tri-systems
  - store nonzero-locations to find matches
- Generate batch of small triangular systems
  - different sizes in uniformly-sized blocks
- Batched trsv
  - different sizes
  - non-coalescent in memory (uniform blocks)
  - use kernel-switch for hard-coded sizes
- Batched re-insertion into sparse SpAI matrix
  - non-coalescent reads/writes

- Batched trsv will become standard building block.
- Batched routines for extracting/inserting data into sparse structures?

Example: ILU - BiCGSTAB



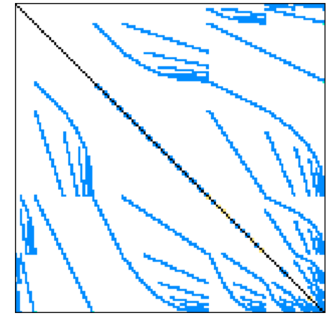
Test matrix: parabolic_fem (n=525,825 nnz=3,674,625)
Use RCM-reordering

BiCGSTAB Preconditioning	Iterations	Prec. setup time	Solver runtime	Total runtime
no	4,752	0.00 s	8.21 s	
ILU + cuSPARSE-trisolves	407	0.67 s	43.72 s	
ILU + Jacobi-trisolves (3s)	1,696	0.73 s	13.01 s	
ILU + Jacobi-trisolves (4s)	905	0.73 s	8.64 s	
ILU + Jacobi-trisolves (5s)	645	0.73 s	7.41 s	
ILU + SPAI (pattern L)	1,459	0.87 s	4.65 s	
ILU + SPAI (pattern L^2)	1,022	1.02 s	4.13 s	
ILU + SPAI (pattern L^3)	693	1.24 s	4.38 s	
ILU + SPAI (pattern L^4)	531	1.58 s	3.70 s	

NVIDIA K40 GPU, CUDA/cuSPARSE v7.0

Example: ILU - BiCGSTAB

Test matrix: parabolic_fem (n=525,825 nnz=3,674,625)
Use RCM-reordering



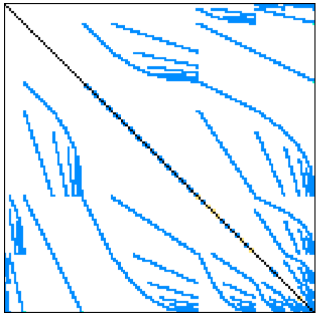
BiCGSTAB Preconditioning	Iterations	Prec. setup time	Solver runtime	Total runtime
no	4,752	0.00 s	8.21 s	
ILU + cuSPARSE-trisolves	407	0.67 s	43.72 s	
ILU + Jacobi-trisolves (3s)	1,696	0.73 s	13.01 s	
ILU + Jacobi-trisolves (4s)	905	0.73 s	8.64 s	
ILU + Jacobi-trisolves (5s)	645	0.73 s	7.41 s	
ILU + SPAI (pattern L)	1,459	0.87 s	4.65 s	
ILU + SPAI (pattern L^2)	1,022	1.02 s	4.13 s	
ILU + SPAI (pattern L^3)	693	1.24 s	4.38 s	
ILU + SPAI (pattern L^4)	531	1.58 s	3.70 s	

30%

NVIDIA K40 GPU, CUDA/cuSPARSE v7.0

Example: ILU - BiCGSTAB

Test matrix: parabolic_fem (n=525,825 nnz=3,674,625)
Use RCM-reordering



BiCGSTAB Preconditioning	Iterations	Prec. setup time	Solver runtime	Total runtime
no	4,752	0.00 s	8.21 s	8.21 s
ILU + cuSPARSE-trisolves	407	0.67 s	43.72 s	44.39 s
ILU + Jacobi-trisolves (3s)	1,696	0.73 s	13.01 s	13.74 s
ILU + Jacobi-trisolves (4s)	905	0.73 s	8.64 s	9.37 s
ILU + Jacobi-trisolves (5s)	645	0.73 s	7.41 s	8.14 s
ILU + SPAI (pattern L)	1,459	0.87 s	4.65 s	5.52 s
ILU + SPAI (pattern L^2)	1,022	1.02 s	4.13 s	5.15 s
ILU + SPAI (pattern L^3)	693	1.24 s	4.38 s	5.62 s
ILU + SPAI (pattern L^4)	531	1.58 s	3.70 s	5.28 s

NVIDIA K40 GPU, CUDA/cuSPARSE v7.0

Example: ILU - BiCGSTAB

Matrix	cuSPARSE		5 Jacobi		SPAI L		SPAI L^2	
Laplace 2D	648	58.04 s	1,263	24.89 s	1,011	4.93 s	743	6.67 s
Laplace 3D	54	4.12 s	85	5.96 s	83	3.64 s	*	*
chipcool0	63	1.94 s	61	0.16 s	102	0.18 s	*	*
apache2	631	22.52 s	-	-	-	-	-	-
ani7	1,356	76.46 s	-	-	5,873	8.94 s	3,178	5.98 s
shallow_water2	9	0.38 s	109	0.41 s	11	0.15 s	9	0.18 s
chem_master1	83	1.35 s	-	-	244	0.24 s	163	0.18 s
stomach	13	1.36 s	14	0.44 s	23	0.50 s	*	*
airfoil2d	50	2.92 s	52	0.16 s	108	0.17 s	*	*
G3_circuit	862	104.22 s	-	-	-	-	-	-
tmt_unsym	?	>10k s	2,455	71.74 s	3,070	40.04 s	2,665	54.34 s
FEM3Dthermal2	6	1.33 s	-	-	21	0.59 s	-	-
venkat01	13	0.99 s	17	0.23 s	38	0.31 s	*	*
mesh96	54	4.63 s	84	5.42 s	83	3.71 s	*	*

* High memory requirement
\- No convergence

NVIDIA K40 GPU, CUDA/cuSPARSE v7.0

Example: ILU - BiCGSTAB

Proof-of-concept study:

- Large set of general matrices from University of Florida Matrix Collection
  UFMC; https://www.cise.ufl.edu/research/sparse/matrices/
- BiCGSTAB as outer solver
- Comparison:
  - exact trsv (cuSPARSE v. 7.5),
  - Jacobi sweeps,
  - SpAI using sparsity pattern of spy(L), spy(L^2), spy(L^3)
- All computational routines from MAGMA-sparse (Nvidia K40 GPU, CUDA 7.5)

http://www.icl.utk.edu/~hanzt/precond_comparison/

Summary and Outlook

SpAI for solving **sparse triangular systems in preconditioning** very efficient:

- SpMV *much faster* than triangular exact solves
- Fast SpAI *generation via batched routines*
- *Flexibility* in choosing the *SpAI nonzero-structure*

Future work

- *Optimize* nonzero-structure (preconditioner quality vs. performance)
- Interplay with *CA-Krylov methods* (precond. communication pattern...)
- SpAI for efficient ILU preconditioning on *distributed systems*

This research is based on a cooperation between Hartwig Anzt from the University of Tennessee, and Thomas Huckle from TU Munich, and partly funded by the Department of Energy.

THE UNIVERSITY OF TENNESSEE KNOXVILLE
U.S. DEPARTMENT OF ENERGY
TECHNISCHE UNIVERSITÄT MÜNCHEN