Solving sequences of linear systems by preconditioned iterative methods

Miroslav Tůma

Institute of Computer Science

Academy of Sciences of the Czech Republic

and Technical University in Liberec

joint work with

Jane Cullum and Jurjen Duintjer Tebbens

Supported by the project “Information Society”

of the Academy of Sciences of the Czech Republic under No. 1ET400300415

Householder Symposium on Numerical Linear Algebra XVI, 2005.

May 23-27, 2005 Seven Springs Mountain Resort, Champion, Pennsylvania, USA

Outline

1. Motivation
2. Our goal: Reuse of matrix approximations / preconditioners
3. Reuse of structural information (preconditioner patterns)
4. Reuse of structural $+$ numerical information (preconditioners)
5. Conclusions

1. Motivation / Newton's method

1. Solving systems of nonlinear equations

$$F(x) = 0$$

$$\Downarrow$$

Sequences of linear systems of the form

$$J(x_k)\Delta x = -F(x_k),\ J(x_k) \approx F'(x_k)$$

solved until for some $k, k = 1, 2, \ldots$

$$\|F(x_k)\| < tol$$

$J(x_k)$ may change at points influenced by nonlinearities

1. Motivation / Nonlinear convection-diffusion

2. Solving nonlinear convection-diffusion problems

$$-\Delta u + u\nabla u = f$$

$$\Downarrow$$

E.g., from the upwind discretization in 2D, with $u \geq 0$ we get for grid internal nodes (i, j)

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij} + hu_{ij}(2u_{ij} - u_{i-1,j} - u_{i,j-1}) = h^2 f_{ij}$$

It is a matrix with five diagonals

Entries in its three diagonals may change in subsequent linear systems

2. Solving nonlinear convection-diffusion problems (continued)

$$-\Delta u + u \nabla u = f$$

$$\begin{pmatrix}
* & * & \dots & * & & & & \\
* & * & * & \dots & * & & & \\
\vdots & * & * & * & \dots & * & & \\
* & \vdots & * & * & * & \dots & * & \\
 & * & \vdots & * & * & * & \dots & * \\
 & & * & \vdots & * & * & * & \dots \\
 & & & * & \vdots & * & * & * \\
 & & & & * & \vdots & * & *
\end{pmatrix}$$

2. Solving nonlinear convection-diffusion problems (continued)

$$-\Delta u + u \nabla u = f$$

$$\begin{pmatrix}
\spadesuit & \spadesuit & \dots & \spadesuit & & & & \\
* & \spadesuit & \spadesuit & \dots & \spadesuit & & & \\
\vdots & * & \spadesuit & \spadesuit & \dots & \spadesuit & & \\
* & \vdots & * & \spadesuit & \spadesuit & \dots & \spadesuit & \\
 & * & \vdots & * & \spadesuit & \spadesuit & \dots & \spadesuit \\
 & & * & \vdots & * & \spadesuit & \spadesuit & \dots \\
 & & & * & \vdots & * & \spadesuit & \spadesuit \\
 & & & & * & \vdots & * & \spadesuit
\end{pmatrix}$$

3. Solving equations with a parabolic term

$$\frac{\partial u}{\partial t} - \Delta u = f$$

$$\Downarrow$$

E.g., 2D problem with 2^{nd} order centered differences in space and backward Euler time discretization for grid internal nodes (i,j) and time step $t+1$

$$h^2(u_{ij}^{t+1} - u_{ij}^{t}) + \tau(u_{i+1,j}^{t+1} + u_{i-1,j}^{t+1} + u_{i,j+1}^{t+1} + u_{i,j-1}^{t+1} - 4u_{ij}^{t+1}) = h^2\tau f_{ij}^{t+1}$$

Again, we get a matrix with five diagonals

Diagonal entries may change with time steps

2. Our goal / Reuse of approximations: I.

Reuse of approximations of matrices in sequences of linear systems

Solving sequences of systems of linear equations

$$A^0 x = b^0,\ A^1 x = b^1,\ \ldots$$

by preconditioned iterative methods with preconditioners

$$M^0, M^1, \ldots$$

Goal: computing M^{i+k} from A^{i+k} for some $k \geq 1$,
and possibly reuse additional information from M^i and A^i.

2. Our goal / Reuse of approximations: II.

Two basic strategies for the information reuse

2. Our goal / Reuse of approximations: II.

Two basic strategies for the information reuse

1. Reuse of matrix patterns
   - Using pattern of M^i.
   - Using pattern of $\widehat{A}_i$ (A_i, or a part of A_i).

2. Our goal / Reuse of approximations: II.

Two basic strategies for the information reuse

1. Reuse of matrix patterns
   - Using pattern of M^i.
   - Using pattern of $\widehat{A}_i$ (A_i, or a part of A_i).
2. Reuse of both patterns and values
   - Using entries of M^i.
   - Using entries of $\widehat{A}_i$ (A_i, or a part of A_i).

2. Our goal / Related work: I.

Some related work

- Preconditioners from quasi-Newton updates (Morales, Nocedal, 2000)
- Freezing approximate Jacobians over a couple of subsequent systems (MNK: Shamanskii, 1967; Brent, 1973).
- Freezing the preconditioner over a couple of subsequent systems (MFNK: Knoll, McHugh, 1998).
- Some simple preconditioners (e.g., Jacobi, ILU(0) for PDEs) may be readily available even in parallel and/or matrix-free environment
- Preconditioners from a related matrix, operator (e.g., based on orthogonal grid, Truchas code, LANL, 2003; cf. Knoll, Keyes, 2004) a lot of approaches)
- Solving systems in adaptive filtering by incomplete factorizations + iterative refinement (Comon, Trystram, 1987)
- Approximate diagonal updates (Benzi, Bertaccini, 2003; Bertaccini, 2004)

2. Our goal / Related work: II.

Some related work (continued)

- World of updates of Krylov subspaces (e.g., Morgan 1995-2002); Baglama, Calvetti, Golub, Reichel, 1999; Carpentieri et. al., 2003; de Sturler, 1996; Erhel, Burrage, Pohl, 1996; Duff, Giraud, Langou, Martin, 2005; Giraud et. al. 2004–2005; Parks et al. 2004)
- Dense updates of decompositions (Bartels, Golub, Saunders, 1970; Gill, Golub, Murray, Saunders, 1974)
- Sparse updates of decompositions (Hager, Davis, 1999–2004).

2. Our goal / Related work: III.

Some related work (continued)

- Use of cheap matrix estimations based on graph coloring techniques in matrix free-environment if we know the matrix structure. This is a classical field; a (very restricted) selection of references: Curtis, Powell; Reid,1974; Coleman, Moré, 1983; Coleman, Moré, 1984; Coleman, Verma, 1998;

  The procedure

  - ⁕ Estimate the matrix A_i by a few matvecs
  - ⁕ Get the preconditioner M_i directly from A_i

- extensions to SPD (Hessian) approximations; extensions to use both A and A^T in automatic differentiation; more sophisticated estimation of resulting entries (substitution methods)
- to get only a part of the matrix which changes in the outer iterations: partial graph coloring, Gebremedhin, Manne, Pothen, 2004.

2. Our goal: An example

The 2D nonlinear convection-diffusion problem (Kelley, 1995); 5-point finite diferences; uniform grid 70×70; first 8 systems; ILUT(0.1,5)

$$\Delta u - Ru\nabla u = 2000x(1-x)y(1-y), \;\; R = 500$$

A-matrix	M-matrix	$CG - its$
A^1	M^1	25
A^2	M^1	98
A^3	M^1	90
A^4	M^1	135
A^5	M^1	179
A^6	M^1	229
A^7	M^1	275
A^8	M^1	345
A^{1-8}	M^{1-8}	25 $\pm$ 10

Freezing the preconditioner may not be enough

3. Reuse of matrix patterns

What do we gain if we use a sparsified pattern for preconditioner computation?

1. Traditionally: Sparsified pattern $\longrightarrow$ cheaper preconditioner computation
2. Proposal 1: Sparsified pattern $+$ matrix-free environment $\longrightarrow$ cheaper but approximate matrix estimation via matvecs (Cullum, T., 2004)
3. Proposal 2: Freezing the matrix pattern in a matrix-free environment: use of the sparsified pattern of a different matrix from a sequence.
4. Proposal 3: Freezing the preconditioner pattern (from a different preconditioner)

3. Reuse of matrix patterns: II.

Traditionally: Sparsified pattern leads to cheaper preconditioner computation.

3. Reuse of matrix patterns: II.

Traditionally: Sparsified pattern leads to cheaper preconditioner computation.

original matrix $\longrightarrow$ large ($\spadesuit$) & small ($\clubsuit$) entries

$$\begin{pmatrix} \spadesuit & \spadesuit & & \spadesuit & & \\ \spadesuit & \spadesuit & & & \spadesuit & \\ & \spadesuit & \spadesuit & & \spadesuit & \\ \spadesuit & & & \spadesuit & \spadesuit & \\ & & & \spadesuit & \spadesuit & \spadesuit \\ & \spadesuit & & & \spadesuit & \spadesuit \end{pmatrix} \begin{pmatrix} \spadesuit & \spadesuit & & \clubsuit & & \\ \spadesuit & \clubsuit & & & \spadesuit & \\ & \clubsuit & \spadesuit & & \spadesuit & \\ \spadesuit & & & \spadesuit & \clubsuit & \\ & & & \clubsuit & \spadesuit & \spadesuit \\ & \spadesuit & & & \clubsuit & \spadesuit \end{pmatrix}$$

3. Reuse of matrix patterns: II.

Traditionally: Sparsified pattern leads to cheaper preconditioner computation.

original matrix $\longrightarrow$ large ($\spadesuit$) & small ($\clubsuit$) entries

$$\begin{pmatrix} \spadesuit & \spadesuit & & \spadesuit & & \\ \spadesuit & \spadesuit & & & \spadesuit & \\ & \spadesuit & \spadesuit & & \spadesuit & \\ \spadesuit & & & \spadesuit & \spadesuit & \\ & & & \spadesuit & \spadesuit & \spadesuit \\ & \spadesuit & & & \spadesuit & \spadesuit \end{pmatrix} \begin{pmatrix} \spadesuit & \spadesuit & & & & \\ \spadesuit & & & & \spadesuit & \\ & & \spadesuit & & \spadesuit & \\ \spadesuit & & & \spadesuit & & \\ & & & & \spadesuit & \spadesuit \\ & \spadesuit & & & & \spadesuit \end{pmatrix}$$

3. Reuse of matrix patterns: Proposal 1

Proposal 1: Sparsified pattern $+$ matrix-free environment $\rightarrow$ cheaper but approximate matrix estimation via matvecs (Cullum, T., 2004)

3. Reuse of matrix patterns: Proposal 1

Proposal 1: Sparsified pattern $+$ matrix-free environment $\rightarrow$ cheaper but approximate matrix estimation via matvecs (Cullum, T., 2004)

First, a gentle introduction into matrix estimation

3. Reuse of matrix patterns: Proposal 1

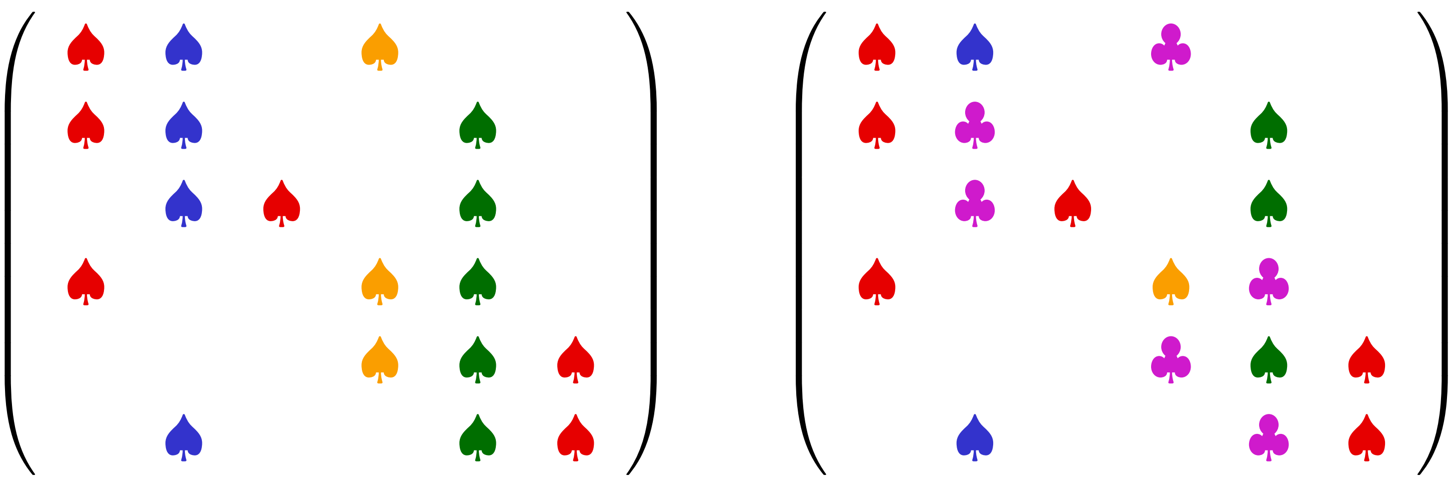
$$
\begin{pmatrix}
\spadesuit & * & & & & & & \\
\spadesuit & * & * & & & & & \\
 & * & * & \spadesuit & & & & \\
 & & * & \spadesuit & * & & & \\
 & & & \spadesuit & * & * & & \\
 & & & & * & * & \spadesuit & \\
 & & & & & * & \spadesuit & * \\
 & & & & & & \spadesuit & *
\end{pmatrix}
$$

Columns with “red spades” can be computed at the same time in one matvec since sparsity patterns of their rows do not overlap.

Namely, $A(e_1 + e_4 + e_7)$ computes entries in the columns 1, 4 and 7.

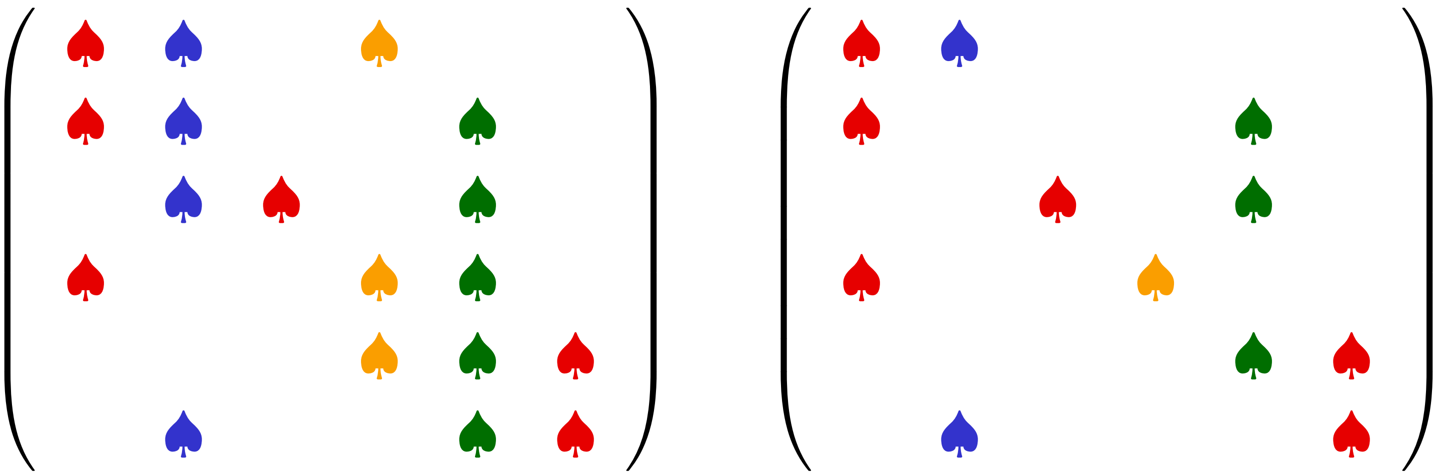
3. Reuse of matrix patterns: Proposal 1

four matvecs needed → large (coloured ♠) & small (♣) entries



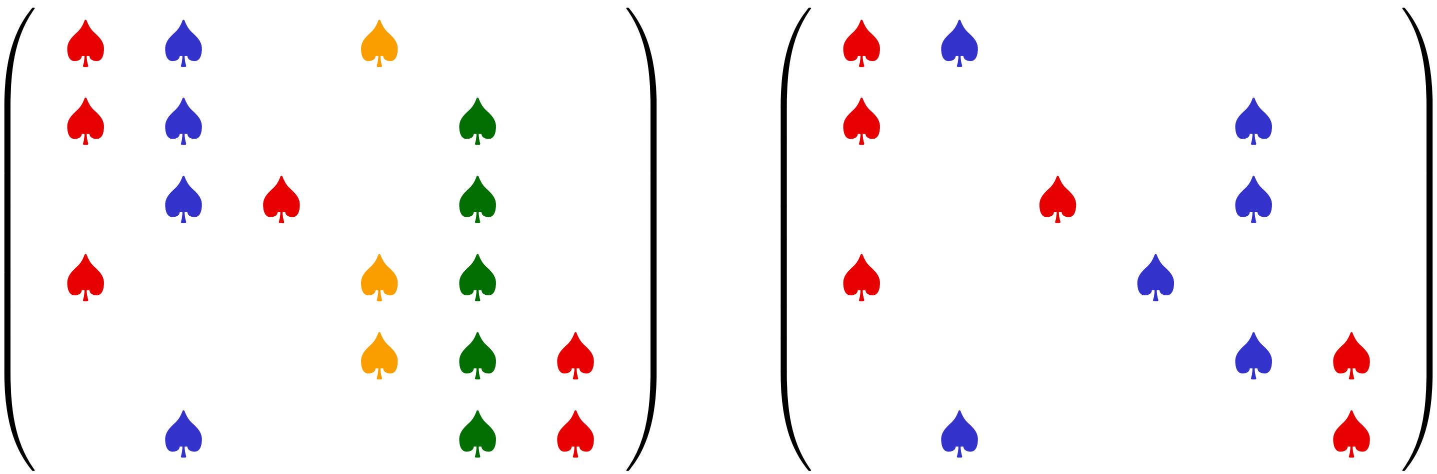
3. Reuse of matrix patterns: Proposal 1

large (coloured $\spadesuit$) $\&$ small ($\clubsuit$) entries $\quad\longrightarrow\quad$ only two matvecs needed



3. Reuse of matrix patterns: Proposal 1

large (coloured ♠) $\&$ small (♣) entries $\longrightarrow$ only two matvecs needed



3. Reuse of matrix patterns: Proposal 2: Results

Proposal 2: Freezing the matrix pattern in a matrix-free environment: use of the sparsified pattern of a different matrix from a sequence.

Driven cavity flow, $R = 500$, ILUT($1.0 * 10^{-6}, 25$); Newton's method

No. A-matrix	No. P-matrix	$CG - its$
A^1	M^1 via A^1 and pattern of $\widehat{A}^1$	44
A^2	M^2 via A^2 and pattern of $\widehat{A}^1$	38
A^3	M^3 via A^3 and pattern of $\widehat{A}^1$	41
A^4	M^4 via A^4 and pattern of $\widehat{A}^1$	42
A^5	M^5 via A^5 and pattern of $\widehat{A}^1$	43
A^2	M^2 via A^2 and pattern of $\widehat{A}^2$	42
A^3	M^3 via A^3 and pattern of $\widehat{A}^3$	38
A^4	M^4 via A^4 and pattern of $\widehat{A}^4$	43
A^5	M^5 via A^5 and pattern of $\widehat{A}^5$	42

Number of matvecs due to the use of sparsified patterns: $51 \longrightarrow 26$

3. Reuse of matrix patterns: Proposal 3

Proposal 3: Freezing the preconditioner pattern (for preconditioners with pattern input)

- Determination of a suitable pattern of preconditioner may be sometimes rather difficult

Our approach:

- Find a pattern for A_0 by a sophisticated / time-consuming method using both symbolic/numeric information
- Use the pattern to get preconditioners $M_0, M_1, \ldots$
- Restart if necessary

3. Reuse of matrix patterns: Proposal 3

old SPAI versus old LS based on old SPAI pattern

The 2D nonlinear convection-diffusion problem (Kelley, 1995); 5-point finite diferences; uniform grid 70×70; first 8 systems; SPAI(0.0,5,5); preconditioner size = 87300

$$\Delta u - Ru\nabla u = 2000x(1-x)y(1-y), \ R = 500$$

NO UPDATES		
$Matrix$	old_SPAI_its	$old\_LS/SPAI\_its$
A^1 / M^1	50	49
A^2 / M^1	76	71
A^3 / M^1	89	79
A^4 / M^1	113	134
A^5 / M^1	138	147
A^6 / M^1	157	161
A^7 / M^1	201	173
A^8 / M^1	238	179

3. Reuse of matrix patterns: Proposal 3

SPAI versus LS with old SPAI pattern

The same 2D nonlinear convection-diffusion problem (Kelley, 1995);
SPAI(0.0,5,5)

new SPAI versus new LS with old SPAI pattern

Number of iterations in SPAI:	50 ± 8
Number of iterations in LS with old SPAI pattern:	50 ± 8
Time for computing SPAI preconditioner	=0.62 – 0.69 s
Time for computing LS preconditioner with old SPAI pattern	=0.22 s
Sequential time for 50 iterations	$\approx$ 0.1 s

4. Reuse of matrix: Entrywise updates

Exploiting both pattern and numerical values

$$A_0 \rightarrow A_1 = A_0 + B$$

4. Reuse of matrix: Entrywise updates

Exploiting both pattern and numerical values

$$A_0 \rightarrow A_1 = A_0 + B$$

Case I: Simple one-entry off-diagonal updates

$$B = uv^T = \alpha e_i e_j^T$$

4. Reuse of matrix: Entrywise updates

Exploiting both pattern and numerical values

$$A_0 \rightarrow A_1 = A_0 + B$$

Case I: Simple one-entry off-diagonal updates

$$B = uv^T = \alpha e_i e_j^T$$

$$A_1^{-1} = (A_0 + B)^{-1} = (A_0 + xy^T)^{-1} = (LDU + xy^T)^{-1} =$$
$$(L(D + L^{-1}xy^TU^{-1})U)^{-1} \approx U^{-1}(I + \alpha D^{-1}e_ie_j^T)^{-1}D^{-1}L^{-1} =$$

$$\frac{U^{-1}(I - \alpha D^{-1}e_ie_j^T)D^{-1}L^{-1}}{1 + \alpha e_j^T D^{-1} e_i}$$

4. Reuse of matrix: Entrywise updates

Exploiting both pattern and numerical values

$$A_0 \to A_1 = A_0 + B$$

Case I: Simple one-entry off-diagonal updates

$$B = uv^T = \alpha e_i e_j^T$$

$$A_1^{-1} = (A_0 + B)^{-1} = (A_0 + xy^T)^{-1} = (LDU + xy^T)^{-1} =$$
$$(L(D + L^{-1}xy^T U^{-1})U)^{-1} \approx U^{-1}(I + \alpha D^{-1} e_i e_j^T)^{-1} D^{-1} L^{-1} =$$

$$\frac{U^{-1}(I - \alpha D^{-1} e_i e_j^T) D^{-1} L^{-1}}{1 + \alpha e_j^T D^{-1} e_i}$$

- Approximation good if there is a strong matrix diagonal – it can be forced by a weighted matching applied to the matrix graph
- Inversion of a Gauss-Jordan transform
- More Gauss-Jordan transforms can be accumulated in one sweep (using pattern-based conditions) . . . next slide

4. Reuse of matrix: Entrywise updates II.

More Gauss-Jordan transforms

- Entries for a sweep with Gauss-Jordan transforms can be found from a weighted spanning forest T of the graph G_B of a (sparsified) difference matrix $B = A_1 - A_0$.

4. Reuse of matrix: Entrywise updates II.

More Gauss-Jordan transforms

- Entries for a sweep with Gauss-Jordan transforms can be found from a weighted spanning forest T of the graph G_B of a (sparsified) difference matrix $B = A_1 - A_0$.

The procedure

1. Find the weighted forest T of G_B
2. Find the order of Gauss-Jordan transforms corresponding to the edges of T (It can be proved that it is feasible)
3. Add to the product of the Gauss-Jordan transforms other nonzeros entries of B (It can be shown which entries, based on simple structural conditions)

4. Reuse of matrix: Entrywise updates III.

Updates of diagonal $+$ subdiagonal representing convection changes in a 2D convection-diffusion problem, $n = 10000$

2D CD problems with a 1D - convection shift		
$Shift$	$NO\_updates\_its$	$CONV\_updates\_its$
0.1	52	54
0.2	58	60
0.3	67	60
0.4	68	61
0.5	75	65
0.6	81	73
0.7	100	76
0.8	121	78
0.9	146	81
1.0	186	82

New preconditioner: 34 iterations

4. Reuse of matrix: Entrywise updates IV.

Updates of diagonal $+$ subdiagonal representing convection changes in a 2D convection-diffusion problem (different from the previous one), $n = 10000$

2D CD problems with a 1D - convection shift		
$Shift$	$NO\_updates\_its$	$CONV\_updates\_its$
0.1	34	33
0.2	36	35
0.3	39	34
0.4	38	33
0.5	44	34
0.6	56	38
0.7	53	34
0.8	63	34
0.9	69	30

New preconditioner: 35 iterations

4. Reuse of matrix: Triangular updates

Exploiting both pattern and numerical values

$$A_0 \rightarrow A_1 = A_0 + B$$

4. Reuse of matrix: Triangular updates

Exploiting both pattern and numerical values

$$A_0 \rightarrow A_1 = A_0 + B$$

Case II: Triangular updates

$$B = L_B + D_B + U_B$$

4. Reuse of matrix: Triangular updates

Exploiting both pattern and numerical values

$$A_0 \rightarrow A_1 = A_0 + B$$

Case II: Triangular updates

$$B = L_B + D_B + U_B$$

$$A_1^{-1} = (A_0 + B)^{-1} = (A_0 + L_B + D_B + U_B)^{-1} =$$
$$(LDU + L_B + D_B + U_B)^{-1} = (L(D + L^{-1}(L_B + D_B + U_B)U^{-1})U)^{-1} \approx$$
$$U^{-1}(I + D(D_B + U_B)U^{-1})^{-1}D^{-1}L^{-1} = (DU + D_B + U_B)^{-1}L^{-1}$$

4. Reuse of matrix: Triangular updates

Exploiting both pattern and numerical values

$$A_0 \longrightarrow A_1 = A_0 + B$$

Case II: Triangular updates

$$B = L_B + D_B + U_B$$

$$A_1^{-1} = (A_0 + B)^{-1} = (A_0 + L_B + D_B + U_B)^{-1} =$$
$$(LDU + L_B + D_B + U_B)^{-1} = (L(D + L^{-1}(L_B + D_B + U_B)U^{-1})U)^{-1} \approx$$
$$U^{-1}(I + D(D_B + U_B)U^{-1})^{-1}D^{-1}L^{-1} = (DU + D_B + U_B)^{-1}L^{-1}$$

- Approximation good in many practical situations
- It can be shown that it behaves theoretically well if $||L - I||$ is small
- . . . but not only in this case, as the subsequent experiments show

4. Reuse of matrix: Triangular updates II.

Triangular update versus no update

The 2D nonlinear convection-diffusion problem (Kelley, 1995); 5-point finite diferences; uniform grid 70×70; first 8 systems; ILUT(0.1,5)

$$\Delta u - Ru\nabla u = 2000x(1-x)y(1-y), \; R = 500$$

matrix / precond	CG its / no update	CG its / triangular update
A^1 / M^1	25	25
A^2 / M^1	98	30
A^3 / M^1	90	27
A^4 / M^1	135	30
A^5 / M^1	179	35
A^6 / M^1	229	36
A^7 / M^1	275	36
A^8 / M^1	345	53
A^{1-8} / M^{1-8}	25 ± 10	25 ± 10

5. Conclusions

- Nonsymmetric preconditioners in the form of decompositions can be successfully updated by algebraic techniques.
- Shown to be efficient in solving nonlinear problems
- A lot of possibilities for improvements under investigation
  - e.g., combination of rank-1 and rank-n updates
  - e.g., sparse updates of incomplete factorizations
- Last but not least: even pattern recycling can help
- A lot of possible combinations with matrix-free environment