On some strategies for Jacobian-free preconditioning of sequences of nonsymmetric linear systems

Jurjen Duintjer Tebbens

Institute of Computer Science

Academy of Sciences of the Czech Republic

joint work with

Miroslav Tůma

Institute of Computer Science

Academy of Sciences of the Czech Republic

SIAM Conference on Optimization, Darmstadt, May 19, 2011.

1. Jacobian-free preconditioning

Consider a sequence of large scale systems of nonlinear equations

$$F^{(j)}(x) = 0, \qquad j = 0, 1, 2, \ldots$$

with continuously differentiable functions

$$F^{(j)} : \mathbb{R}^n \to \mathbb{R}^n.$$

They arise in numerous scientific and industrial applications, e.g. every time step t_j in a numerical simulation may require the solution of $F^{(j)}(x) = 0$. They arise in Navier-Stokes equations, heat conduction problems, differential-algebraic problems, many kinds of initial and boundary value problems and many others.

1. Jacobian-free preconditioning

We restrict ourselves to inexact Newton-Krylov methods (see e.g. [Kelley - 1995]) where each Newton iteration has the form

$$J(x_k)(x_{k+1} - x_k) = -F^{(j)}(x_k), \quad k = 1, 2, \ldots,$$

where $J(x_k)$ represents the Jacobian of $F^{(j)}$ evaluated at x_k.

If the linear systems with the Jacobians are solved by a transpose-free Krylov subspace method, one can use a Jacobian-free implementation:

The multiplication of a vector v with $J(x_k)$ is replaced by a finite-difference approximation, for example the first-order approximation

$$J(x_k) \cdot v \approx \frac{F^{(j)}(x_k + h\|x_k\|v) - F^{(j)}(x_k)}{h\|x_k\|},$$

for some small h.

1. Jacobian-free preconditioning

- Exploiting finite-difference approximation, the system matrix needs not be stored and not even be computed. Convergence is Newton is in general not significantly influenced.

- Such an implementation is sometimes called „matrix-free", which is a little misleading: It is often necessary to store preconditioners and smaller matrices in some Krylov subspace methods.

- A possible definition of matrix-free is „free of matrices with storage comparable to that of the Jacobian" [Knoll, Keyes - 2004].

- We will use the term „Jacobian-free".

1. Jacobian-free preconditioning

Preconditioning is often crucial for satisfactory performance of both the Krylov and the Newton iteration.

Some preconditioners that do not need the entries of the Jacobians explicitly in order to be applied are:

- Multigrid and additive Schwarz preconditioners based on domain decomposition and grid coarsening,
- Fast Poisson solvers and and other simplifications of operators as preconditioners in e.g. convection-diffusion-reaction problems,
- Krylov subspace methods as preconditioners for the inner iteration of an outer Krylov subspace, e.g. flexible GMRES [Saad - 1993].

1. Jacobian-free preconditioning

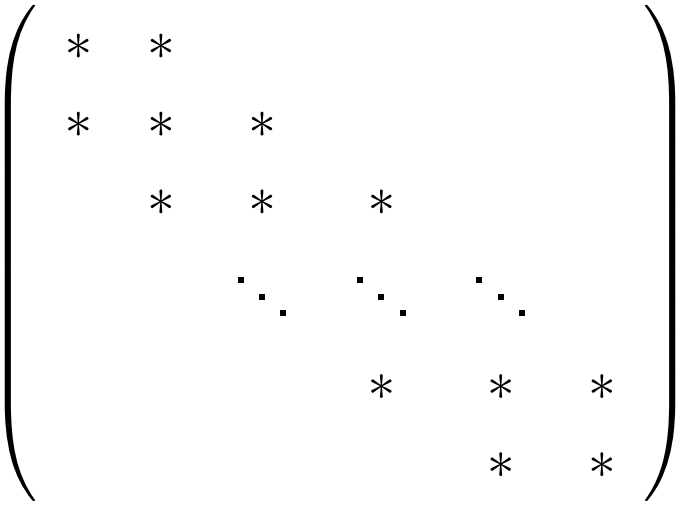
- For particular problems these preconditioners can be superior to incomplete factorizations, for some references see [Knoll, Keyes - 2004].
- In other cases one needs the universality and robustness of incomplete LU or Cholesky decomposition.
- These factorizations need the entries of the Jacobian explicitly!
- The only way to estimate these entries is through simulating matrix-vector products (matvecs) with selected test vectors v_t, e.g.

$$J(x_k) \cdot v_t \approx \frac{F^{(j)}(x_k + h\|x_k\|v_t) - F^{(j)}(x_k)}{h\|x_k\|}.$$

In [Curtis, Powell, Reid - 1978] it was first shown this may be done with a small number of matvecs if the sparsity pattern of the Jacobian is known.

1. Jacobian-free preconditioning

For example a tridiagonal Jacobian of arbitrary size



can be estimated through only three matvecs, namely with

$$(1,0,0,1,0,0,\dots)^T,$$
$$(0,1,0,0,1,0,\dots)^T,$$
$$(0,0,1,0,0,1,\dots)^T.$$

1. Jacobian-free preconditioning

In general, one uses a graph coloring algorithm to estimate the entries of the Jacobian [Coleman, More - 1984].

- The algorithms work on the *intersection graph*, that is on $G(J^T J)$
- The computational costs of graph coloring algorithms are linear in the number of nonzeros
- Every color defines a test vector whose matvec simulated with a finite difference approximation gives some entries of the Jacobian
- Heuristics are used to minimize the number of colors

Note that to construct good preconditioners it need not be necessary to have *all* nonzeros of the Jacobian. Partial estimation with modified graph coloring may be cheaper [Cullum, Tůma - 2006].

1. Jacobian-free preconditioning

When solving the sequence of large scale systems of nonlinear equations

$$F^{(j)}(x) = 0, \qquad j = 0, 1, 2, \ldots$$

one will always try to share part of the computational effort throughout the sequence. An option is reusing the same preconditioner over several Newton iterations, i.e. freeze the preconditioner. Note that recomputing the preconditioner requires for every linear system:

- A number of additional matvec simulations (i.e. function evaluations) to estimate the current matrix,
- When the sparsity pattern changes during the sequence: Rerunning the graph coloring algorithm,
- Rerunning the incomplete factorization.

Unfortunately, a frozen preconditioner can deteriorate when the system matrix changes too much. A good compromise may be approximate preconditioner *updates*.

2. Approximate preconditioner updates

Some proposed preconditioner updates include:

- In [Meurant - 2001] and [Bellavia, de Simone, di Serafino, Morini - 2011] we find approximate preconditioner updates of incomplete Cholesky factorizations for shifted SPD matrices.

- In Quasi-Newton methods the difference between system matrices is of small rank and preconditioners may be efficiently adapted with approximate small-rank preconditioner updates; this has been done in the symmetric positive definite case, see e.g. [Bergamaschi, Bru, Martínez, Putti - 2006, Nocedal, Morales - 2000].

- The preconditioner update can consist of adding recycled (spectral) information from previously generated Krylov subspaces. This can be beneficial in many applications, see e.g. [Parks, de Sturler, Mackey, Johnson, Maiti - 2006], [Giraud, Gratton, Martin - 2007], [Frank, Vuik - 2001].

2. Approximate preconditioner updates

The previous factorization updates are more or less problem specific. We now describe a class of more black-box approximate preconditioner updates based on an idea in [Benzi, Bertaccini - 2003].

Notation: Consider two linear systems in the sequence,

$$Jx = b, \qquad \text{and} \qquad J^+x^+ = b^+$$

and let

$$\Delta \equiv J - J^+.$$

Further, let

$$J \approx LDU$$

be a reference (seed) ILU factorization for the reference Jacobian J.

2. Approximate preconditioner updates

Then

$$J - LDU = J^+ - LDU + \Delta = J^+ - L(D + L^{-1}\Delta U^{-1})U,$$

i.e. the preconditioner update

$$L(D + L^{-1}\Delta U^{-1})U$$

is of the same accuracy as LDU.

This update is not useful as preconditioner because of the middle factor.

Assuming *fast decay* when moving away from the main diagonal in L^{-1} and U^{-1}, two ideas to modify the middle factor were proposed. Note that fast decay is often given in diagonally dominant problems, or it can be stimulated by reorderings that move large entries close to the main diagonal.

2. The considered preconditioner updates

- Consider

$$L(D + \text{band}(L^{-1}\Delta U^{-1}))U$$

for a small bandwidth $\Rightarrow$ additional inexpensive forming of $\text{band}(L^{-1}\Delta U^{-1})$ to compute the preconditioner and additional inexpensive solving of a banded system to apply the preconditioner. This update was used in [Bertaccini - 2004].

- Consider

$$L(D + \Delta U^{-1})U = L(DU + \Delta) \approx L(DU + \text{triu}\Delta)$$

or

$$L(D + L^{-1}\Delta)U = (LD + \Delta)U \approx (LD + \text{tril}\Delta)U$$

$\Rightarrow$ no additional costs to compute or apply the preconditioner, but based on rather greedy simplification, see [DT, Tůma - 2007].

2. Approximate preconditioner updates

Analogously, if

$$J^{-1} \approx W \tilde{D}^{-1} Z^T, \qquad W, Z \quad \text{upper triangular},$$

is a reference approximate inverse factorization for J, we can use the banded factorization update

$$W(\tilde{D} + \text{band}(Z^T \Delta W))^{-1} Z^T,$$

see [Benzi, Bertaccini - 2003] (symmetric case) and [Bellavia, Bertaccini, Morini - 2011], or we can use the triangular updates

$$W(\tilde{D} Z^T + \text{tril}(\Delta))^{-1}, \qquad (\tilde{W} D + \text{triu}(\Delta))^{-1} Z^T.$$

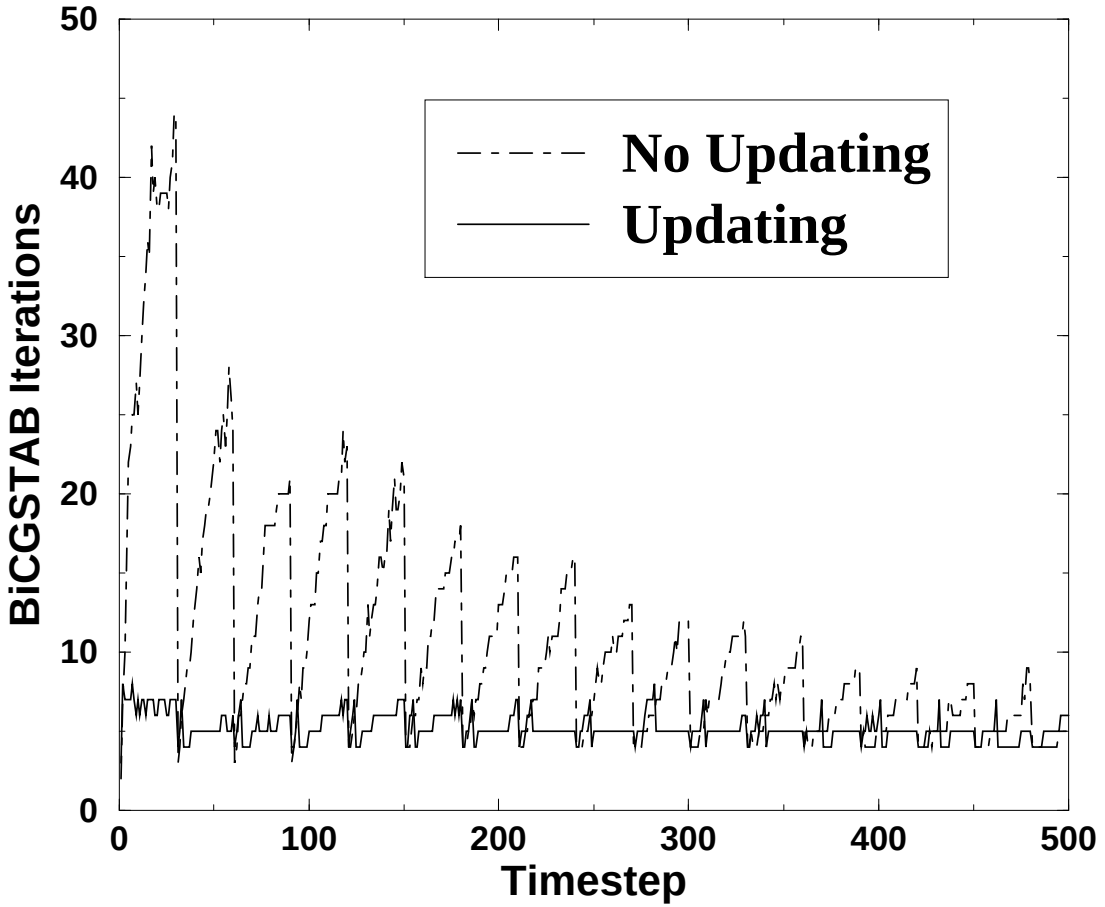
Note that in general approximate *inverse* factorizations need more nonzeros than ILU or Cholesky to be efficient and thus are less suited for Jacobian-free environment.

2. Approximate preconditioner updates

Consider the following CFD problem (compressible supersonic flow):

- Frontal flow with Mach-number 10 around a cylinder, which leads to a steady state.
- 500 time steps of the implicit Euler method are performed.
- The grid consists of 20994 points, we use Finite Volume discretization and system matrices are of dimension 83976. The number of nonzeroes is about $1.33 \cdot 10^6$ for all matrices of the sequence.
- In the beginning, a strong shock detaches from the cylinder, which then slowly moves backward through the domain until reaching the steady state position.
- The iterative solver is BiCGSTAB with stopping criterion 10^{-7}, the implementation is in C++.
- The ILU preconditioner is recomputed for every 30th linear system.

2. Approximate preconditioner updates



BiCGSTAB iterations for the first 500 systems in the cylinder problem.

2. Approximate preconditioner updates

Back to Jacobian-free implementation:

The described class of updates works with the difference matrix, that is with

$$\text{band}(Z^T \Delta W), \text{tril}(\Delta), \text{triu}(\Delta), \qquad \text{where} \qquad \Delta = J - J^+.$$

The reference Jacobian J has usually been estimated in order to obtain the reference factorization, but J^+ has not.

Instead of direct estimation of the current Jacobian J^+, two directions for improvement were proposed:

- Adapted graph coloring algorithms to estimate only the necessary entries of J^+,
- Cheap estimation of selected entries with function component evaluations.

First consider e.g. estimating $\text{triu}(\Delta)$ with adapted graph coloring.

2. Approximate preconditioner updates

Academic example:

$$J^+ = \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & \ddots & & \vdots \\ & & & \ddots & * \\ * & * & * & * & * \end{pmatrix}$$

- estimating the whole matrix asks for n matvecs with all unit vectors (i.e. there are n colors in the intersection graph);
- estimating the upper triangular part requires only 2 matvecs (i.e. there are 2 colors in the intersection graph),

$$(1, \ldots, 1, 0)^T \qquad \text{and} \qquad (0, \ldots, 0, 1)^T.$$

2. Approximate preconditioner updates

Recall the graph coloring algorithm for a matrix C works on the *intersection graph*

$$G(C^T C).$$

We can prove [DT, Tůma - 2010]: The graph coloring algorithm for $triu(C)$ works on

$$G(triu(C)^T triu(C)) \cup G_K, \qquad \text{where}$$

$$G_K = \cup_{i=1}^{n} G_i, \quad G_i = (V_i, E_i) = (V, \{\{k, j\} |\ c_{ik} \neq 0 \wedge c_{ij} \neq 0 \wedge k \leq i < j\}).$$

- This graph may have less edges than $G(C^T C)$ and there may be needed significantly less matvecs to estimate $triu(C)$ than to estimate C
- Similar modified coloring may be used to estimate the diagonals of J^+ needed to form band$(Z^T \Delta W)$
- These are instances of partial graph coloring problems [Pothen et al. - 2007].

2. Approximate preconditioner updates

As for the second strategy, assume the components $F_i^+ : \mathbb{R}^n \to \mathbb{R}$ of the current function

$$F^+ = [F_1^+, \ldots, F_n^+]^T : \mathbb{R}^n \to \mathbb{R}^n$$

can be easily *separated*. By this we mean that a function component evaluation $F_i^+(x)$ has the cost of about one nth of the full function evaluation $F^+(x)$. (Note that in some Finite Element or Volume implementations different components F_i^+ of F^+ contain the same or simultaneously generated expressions and the evaluation of $F^+(x)$ may be cheaper than n function component evaluations $F_i^+(x)$.)

With easily separable function component evaluations it may pay-off to compute the needed entries of J^+ individually according to

$$e_i^T J^+(x_k) e_j \approx \frac{F_i^+(x_k + h\|x_k\|e_j) - F_i^+(x_k)}{h\|x_k\|}.$$

2. Approximate preconditioner updates

By individual estimation with function component evaluations, we are able to avoid running a graph coloring algorithm.

Easily separable function components also give cheap estimations of *inner products with rows of the Jacobian*. For an arbitrary vector v, the inner product with the ith row of J^+ can be computed as

$$e_i^T J^+(x_k) v \approx \frac{F_i^+(x_k + h\|x_k\|v) - F_i^+(x_k)}{h\|x_k\|}$$

$\Rightarrow$ the entries of J^+W needed to form

$$\text{band}(Z^T \Delta W) = \text{band}(Z^T (J - J^+) W)$$

can be obtained from inner products with the rows of J^+ [Bellavia, Bertaccini, Morini - 2011]. This is in general faster than estimating the needed entries of J^+ individually. For extracting bands of Hessians in truncated Newton with function component evaluations, see [Lukšan, Matonoha, Vlček - submitted].

2. Approximate preconditioner updates

For the preconditioner update $L(DU - \text{triu}\Delta)$ (or $(LD - \text{tril}\Delta)U$) we can avoid any estimation of entries of J^+ (except for the main diagonal). We can apply $L(DU - \text{triu}\Delta)$ as follows:

- The forward solves with L are done with the stored entries of L.
- For the backward solves, use a mixed explicit-implicit solves: Split

$$DU - \text{triu}\Delta = DU - triu(J) + triu(J^+) \equiv X + triu(J^+)$$

  in the explicitly given $X \equiv DU - triu(J)$ and the implicit $triu(J^+)$.
- We then have to solve the upper triangular systems

$$(DU - \text{triu}\Delta)\, z = \left(X + triu(J^+)\right) z = y,$$

  yielding the standard backward substitution cycle

$$z_i = \frac{y_i - \sum_{j>i} X_{ij} z_j - \sum_{j>i} J^+_{ij} z_j}{X_{ii} + J^+_{ii}}, \qquad i = n, n-1, \ldots, 1.$$

2. Approximate preconditioner updates

In

$$z_i = \frac{y_i - \sum_{j>i} X_{ij} z_j - \sum_{j>i} J^+_{ij} z_j}{X_{ii} + J^+_{ii}}, \qquad i = n, n-1, \ldots, 1.$$

the inner product $\sum_{j>i} J^+_{ij} z_j$ with the ith row of J^+ can be computed by the function component evaluation

$$\begin{aligned} \sum_{j>i} J^+_{ij} z_j &= e_i^T J^+ (0, \ldots, 0, z_{i+1}, \ldots, z_n)^T \\ &\approx \frac{F_i^+(x_k + h\|x_k\|(0, \ldots, 0, z_{i+1}, \ldots, z_n)^T) - F_i^+(x_k)}{h\|x_k\|}. \end{aligned}$$

The diagonal $\{J^+_{11}, \ldots, J^+_{nn}\}$ can be found by individual estimation

$$J^+_{ii} = e_i^T J^+(x_k) e_i \approx \frac{F_i^+(x_k + h\|x_k\| e_i) - F_i^+(x_k)}{h\|x_k\|}, \qquad 1 \le i \le n.$$

2. Approximate preconditioner updates

- This last technique enables to do backward or forward solves with J^+ without storage of off-diagonal entries.

- The same can be done for the reference Jacobian, hence we do not need to form or store off-diagonal entries of $\Delta = J - J^+$

- This gain in storage costs is paid be slightly higher computational costs: Every forward or backward solve requires n function component evaluations, i.e. roughly one function evaluation, instead of n inner products.

2. Approximate preconditioner updates

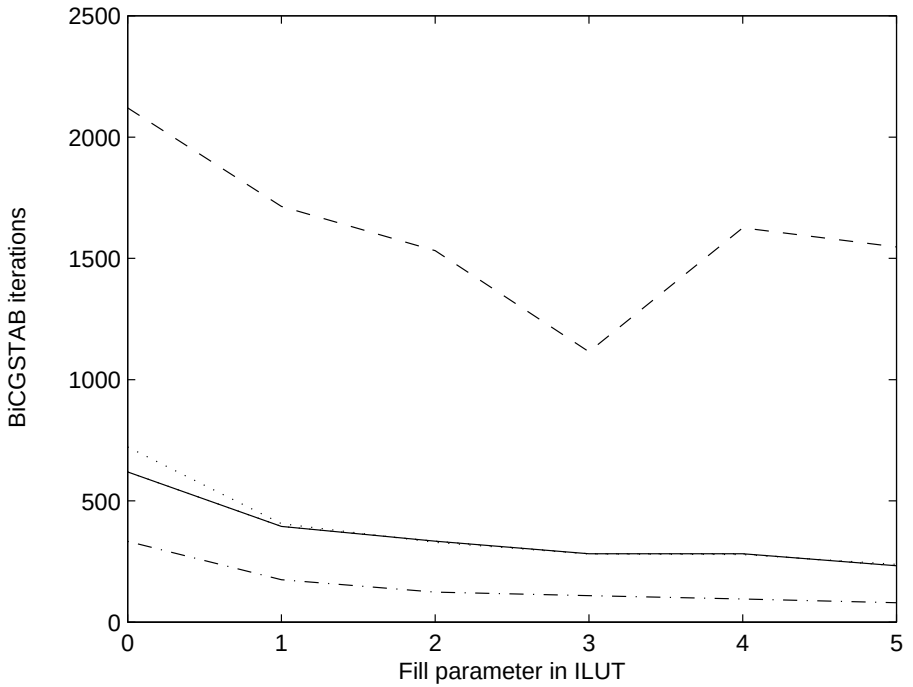
As an example consider a two-dimensional nonlinear convection-diffusion model problem: It has the form

$$-\Delta u + Ru\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) = 2000x(1-x)y(1-y), \tag{1}$$

on the unit square, discretized by 5-point finite differences on a uniform grid.

- The initial approximation is the discretization of $u_0(x, y) = 0$.
- We use here $R = 500$.
- We use the Newton method with a line search and solve the resulting linear systems with BiCGSTAB with right preconditioning.
- We use a flexible stopping criterion (see e.g. [Eisenstat, Walker - 1996]).
- Fortran implementation (embedded in the UFO - software for testing nonlinear solvers [Lukšan et al. - 2008]).

2. Approximate preconditioner updates



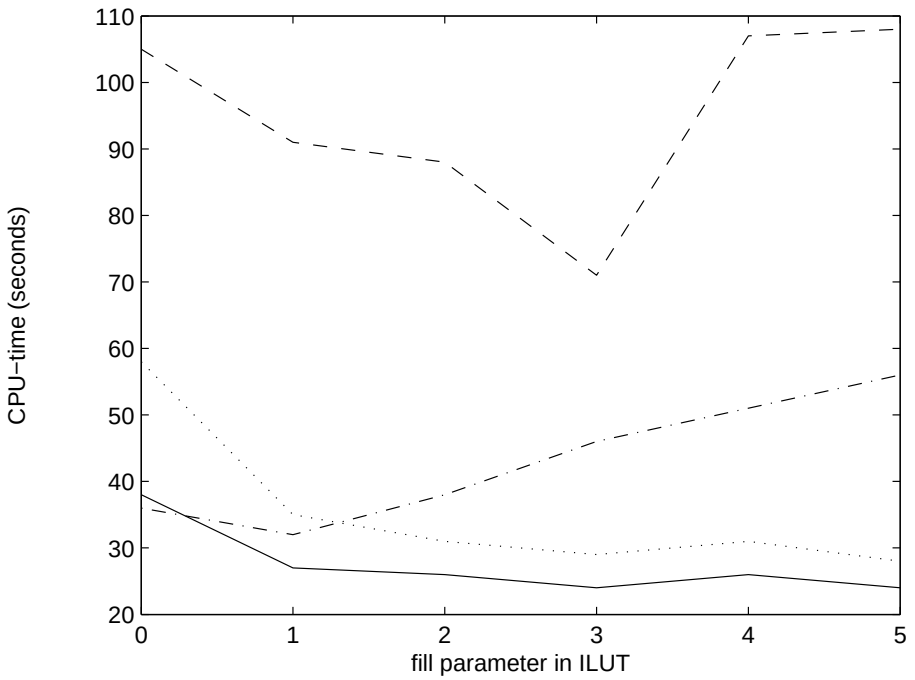
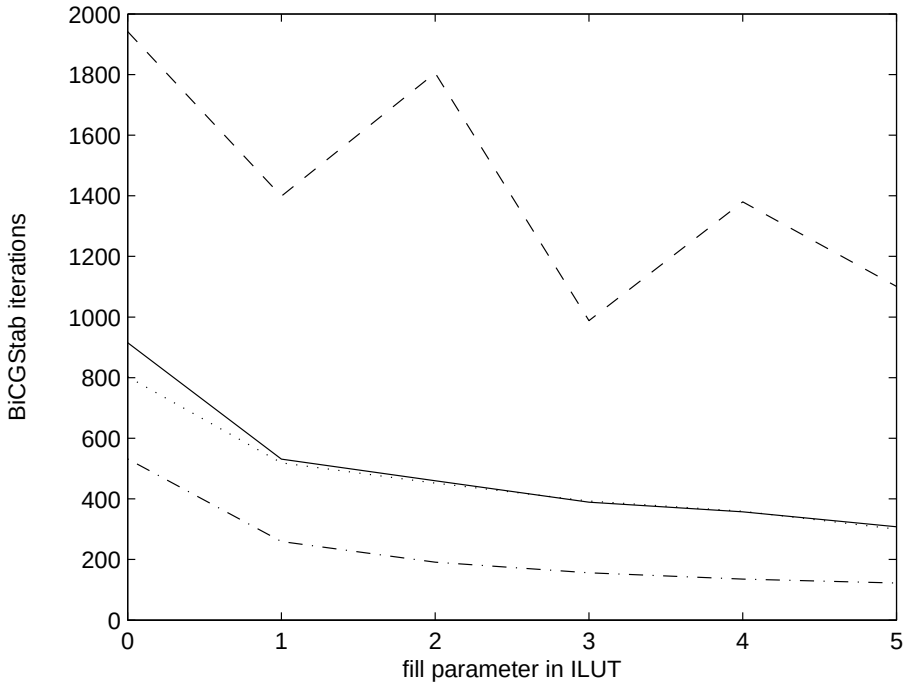


Figure 1: BiCGStab iterations and CPU-times for a 250×250 grid (dimension 62 500) with varying sizes of ILUT-factorizations (depending on the fill parameter) for freezing (dashed lines), recomputing (dash-dotted lines), triangular updating with modified graph coloring (solid lines) and triangular updating with mixed explicit-implicit solves (dotted lines).

2. Approximate preconditioner updates



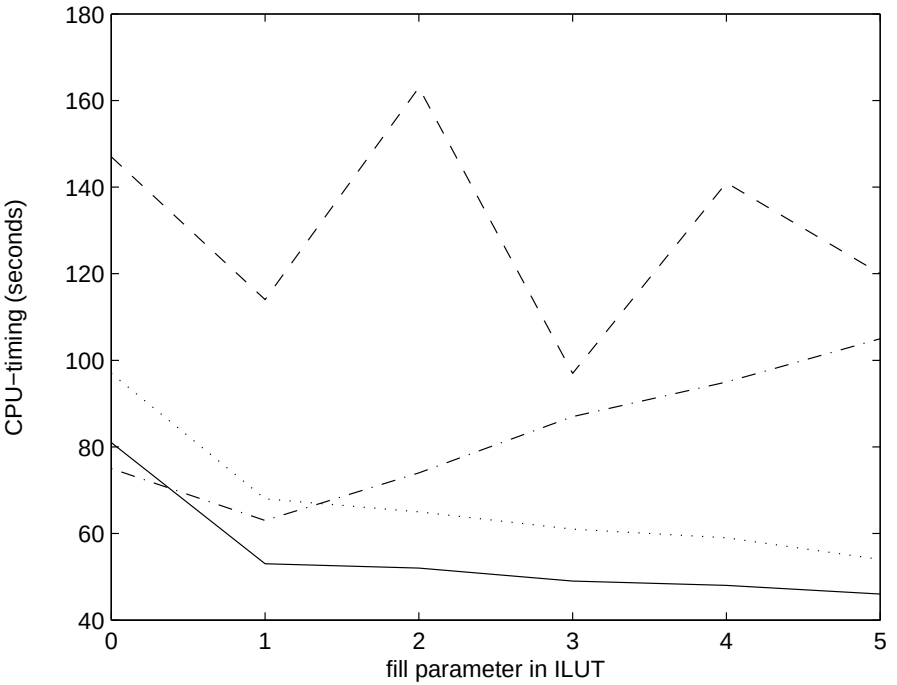
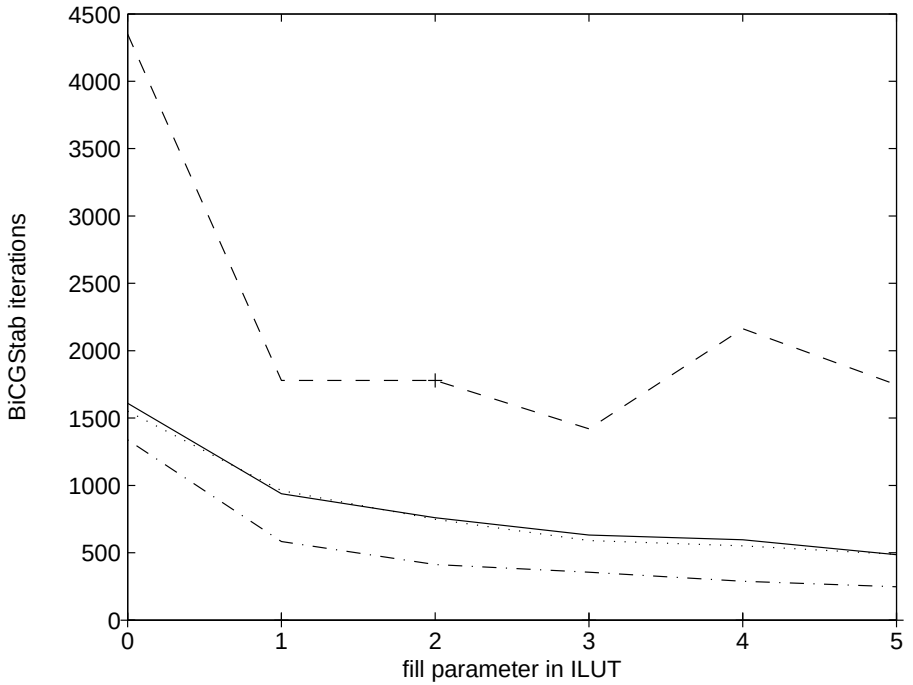


Figure 2: BiCGStab iterations and CPU-times for a 310×310 grid (dimension 96 100) with varying sizes of ILUT-factorizations (depending on the fill parameter) for freezing (dashed lines), recomputing (dash-dotted lines), triangular updating with modified graph coloring (solid lines) and triangular updating with mixed explicit-implicit solves (dotted lines).

2. Approximate preconditioner updates



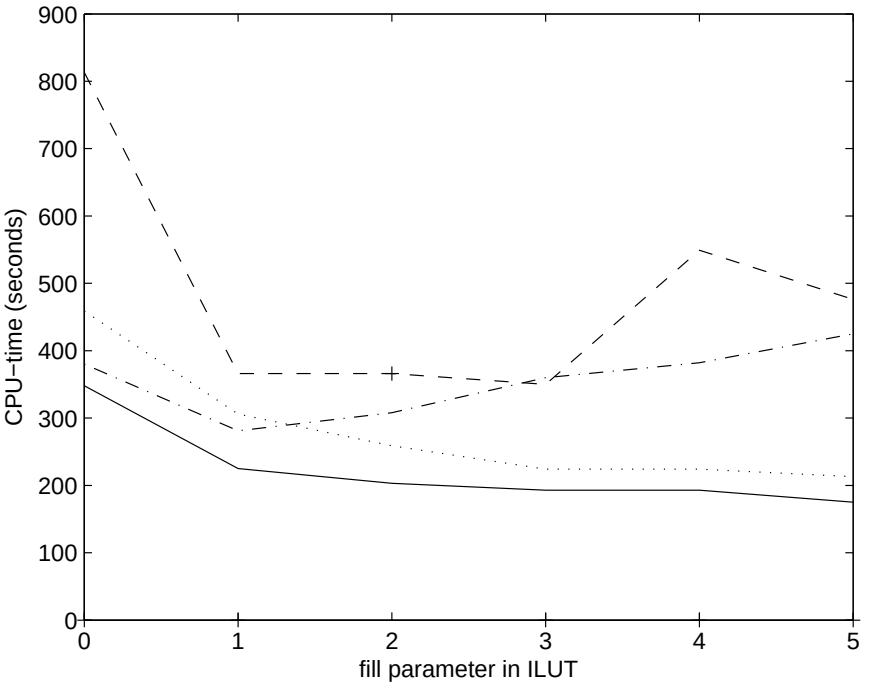


Figure 3: BiCGStab iterations and CPU-times for a 490×490 grid (dimension 240 100) with varying sizes of ILUT-factorizations (depending on the fill parameter) for freezing (dashed lines), recomputing (dash-dotted lines), triangular updating with modified graph coloring (solid lines) and triangular updating with mixed explicit-implicit solves (dotted lines).

Conclusion

Conclusions and future work:

- The described preconditioner updates are suitable for Jacobian-free sequences of linear systems
- We showed that triangular solves with the Jacobians can be done while storing only their diagonal $\rightarrow$ all Gauss-Seidel type preconditioners can be implemented in this way
- For the far future: work towards ILU preconditioning without storing the L and U factors (or only their main diagonal) ??

For more details see:

- DUINTJER TEBBENS J, TŮMA M: *Preconditioner Updates for Solving Sequences of Linear Systems in Matrix-Free Environment*, Num. Lin. Alg. Appl. vol. 17, pp. 997–1019, 2010.
- BIRKEN PH, DUINTJER TEBBENS J, MEISTER A, TŮMA M: *Preconditioner Updates Applied to CFD Model Problems*, Applied Numerical Mathematics vol. 58, no. 11, pp. 1628–1641, 2008.
- DUINTJER TEBBENS J, TŮMA M: *Efficient Preconditioning of Sequences of Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., vol. 29, no. 5, pp. 1918–1941, 2007.

Thank you for your attention!

Supported by project number IAA100300802 of the grant agency of the Academy of Sciences of the Czech Republic.