On a Black-Box Preconditioner Update for Solving Sequences of Nonsymmetric Linear Systems in Matrix-free Environment

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

and Technical University in Liberec

Winter school "New trends in scientific computing", Luminy, February 11, 2009.

1. Introduction: Efficient solution of sequences

The solution of *sequences* of linear systems arises in numerous applications, e.g. CFD-problems with implicit Euler leads to a number of linear systems in every time-step.

The central question for efficient solution will always be: How can we share part of the computational effort throughout the sequence ?

Below we list some known strategies.

- Very simple trick: Hot starts, i.e. use the solution of the previous system as initial guess. A little more sophisticated is extrapolation of the current solution.

- If the linear solver is a Krylov subspace method, strategies to recycle information gained from previously generated Krylov subspaces can be beneficial in many applications [Parks, de Sturler, Mackey, Johnson, Maiti - 2006], [Giraud, Gratton, Martin - 2007], [Frank, Vuik - 2001].

1. Introduction: Efficient solution of sequences

- Sometimes exact updating of the LU decompositions for large problems is feasible: Rank-one updates of LU factorizations have been used since decades in the simplex method where the change of one system matrix to another is restricted to one column [Bartels, Golub, Saunders - 1970; Forrest, Tomlin - 1972 ; Suhl, Suhl - 1993].

- In nonlinear systems solved with a Newton-type method one can skip evaluations of the (approximate) Jacobian during some iterations, leading to Shamanskii's combination of the chord and Newton method [Brent - 1973].

- Another option: Allow changing the system matrices but freeze the preconditioner [Knoll, Keyes - 2004]. Naturally, a frozen preconditioner will deteriorate when the system matrix changes too much.

1. Introduction: Efficient solution of sequences

To enhance the power of a frozen preconditioner one may also use approximate preconditioner updates:

- In [Meurant - 2001] we find approximate preconditioner updates of incomplete Cholesky factorizations and
- in [Benzi, Bertaccini - 2003, 2004] banded preconditioner updates were proposed for both symmetric positive definite approximate inverse and incomplete Cholesky preconditioners.
- 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].

2. The considered preconditioner updates

In this presentation we focus on a preconditioner update proposed recently [DT, Tůma - 2007]:

- A black-box approximate preconditioner update designed for *nonsymmetric* linear systems solved by arbitrary iterative methods.
- Its computation is much cheaper than the computation of a new preconditioner.
- Simple algebraic updates which can be easily combined with other (problem specific) strategies and can be applied in matrix-free environment.

2. The considered preconditioner updates

Notation:

Consider two systems

$$Ax = b, \qquad \text{and} \qquad A^+x^+ = b^+$$

preconditioned by M and M^+ respectively and let

$$B \equiv A - A^+.$$

We would like M^+ to be an update of M that is as powerful as M itself.

If $||A - M||$ is the accuracy of the preconditioner M for A, we will try to find an updated M^+ for A^+ with comparable accuracy,

$$||A - M|| \approx ||A^+ - M^+||.$$

2. The considered preconditioner updates

Let M be factorized as

$$M = LDU \approx A,$$

then the choice

$$M^+ = LDU - B$$

would give $||A - M|| = ||A^+ - M^+||$.

We will approximate this ideal update $LDU - B$ in two steps, similarly to the techniques in [Benzi, Bertaccini - 2003, Bertaccini - 2004]. First we use

$$LDU - B = L(DU - L^{-1}B) \approx L(DU - B) \quad \text{or}$$

$$LDU - B = (LD - BU^{-1})U \approx (LD - B)U$$

depending on whether L is closer to identity or U. Define the standard splitting

$$B = L_B + D_B + U_B.$$

2. The considered preconditioner updates

Then the second approximation step is

$$L(DU - B) \approx L(DU - D_B - U_B) \equiv M^+$$

(upper triangular update) or

$$(LD - B)U \approx (LD - L_B - D_B)U \equiv M^+$$

(lower triangular update). Then M^+ is for free and its application asks for one forward and one backward solve. Schematically,

type	initialization	solve step	memory
Recomp	$A^+ \approx L^+U^+$	solves with L^+, U^+	A^+, L^+, U^+
Update	—	solves with $L, U, triu(B)$	$A^+, triu(A), L, U$

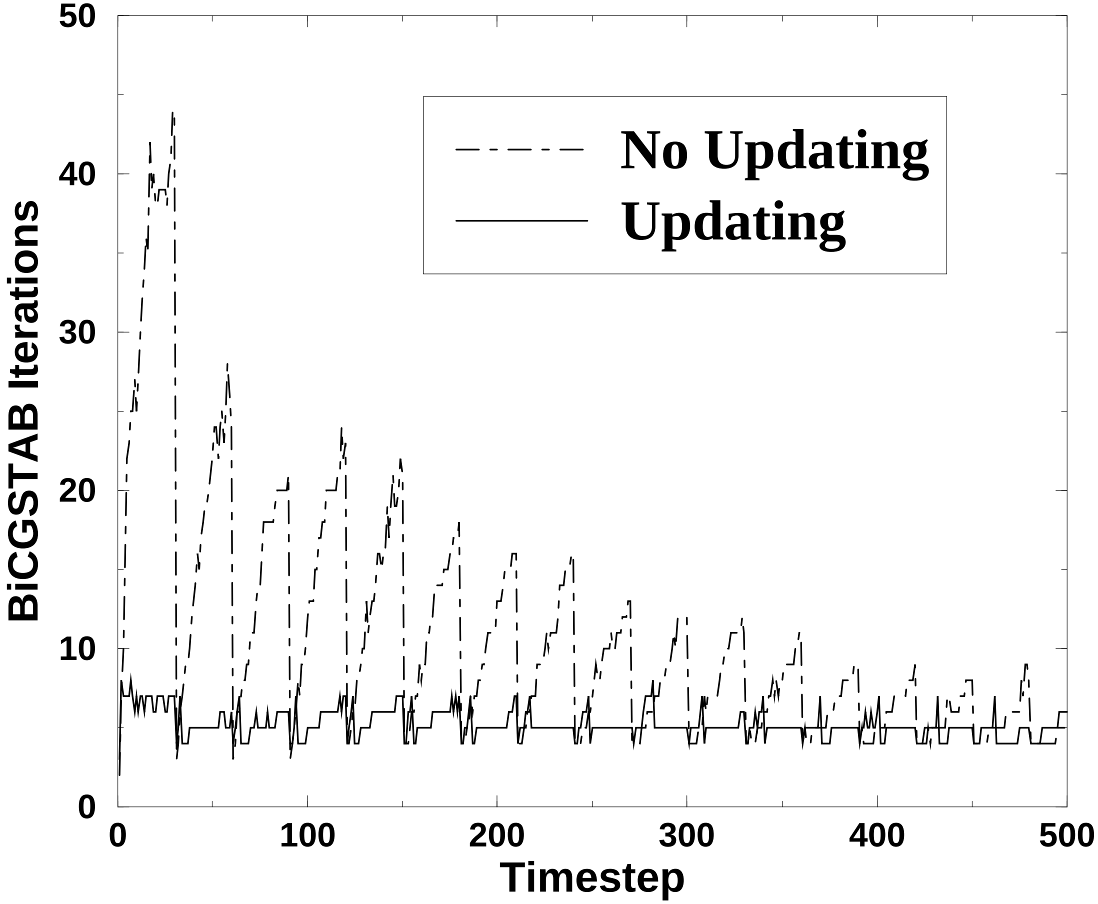
- This is the basic idea; more sophisticated improvements are possible
- Ideal for upwind/downwind modification but our experiments cover broader spectrum of problems

2. The considered 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 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++.

2. The considered preconditioner updates



BiCGSTAB iterations for the first 500 systems in the cylinder problem

3. Updates in matrix-free environment

An important advantage of Krylov subspace methods is that they do not require the system to be stored explicitly; a matrix-vector product (matvec) subroutine, based on a function evaluation, suffices.

$\Rightarrow$ important reducing of storage and computation costs.

Standard example: Newton iteration of the form

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

where $J(x_k)$ is the Jacobian of F evaluated at x_k. Then a matvec with $J(x_k)$ is replaced by the standard difference approximation,

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

for some small h.

3. Updates in matrix-free environment

First note that to compute an incomplete factorization in matrix-free environment at all, the system matrix has to be *estimated by matvecs* ; for example a tridiagonal matrix

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

can be easily obtained with matvecs with

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

3. Updates in matrix-free environment

In general, one uses a graph coloring algorithm that tries to minimize the number of matvecs for a good estimate [Cullum, Tůma - 2006].

Hence recomputing the preconditioner requires for every linear system:

- A number of additional matvecs to estimate the current matrix
- When the nonzero pattern changes during the sequence: Rerunning the graph coloring algorithm.

Preconditioner recomputation is even more expensive in matrix-free environment !

How about the preconditioner updates ?

3. Updates in matrix-free environment

Recall the upper triangular update is of the form

$$M^+ = L(DU - D_B - U_B)$$

based on the splitting

$$L_B + D_B + U_B = B = A - A^+.$$

Thus the update needs some entries of A and A^+ and repeated estimation is necessary.

However:

- A has been estimated before to obtain the reference ILU-factorization
- Of A^+ we need estimate only the upper triangular part
- Can there be taken any advantage of the fact we estimate only the upper triangular part?

3. Updates in matrix-free environment

Example:

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

- estimating the whole matrix asks for n matvecs with all unit vectors;
- estimating the upper triangular part requires only 2 matvecs,

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

The problem of estimating only the upper triangular part leads to a *partial graph coloring problem* [Pothen et al. - 2007].

3. Updates in matrix-free environment

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

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

We can prove: The graph coloring algorithm for $triu(C)$ works on

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

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

Combined with a priori sparsification, there may be needed significantly less matvecs to estimate $triu(C)$ than to estimate C. Summarizing,

type	initialization	solve step	memory
Recomp	$est(A^+)$, $A^+ \approx L^+U^+$	solves with L^+, U^+	L^+, U^+
Update	$est(triu(A^+))$	solves with $L, U, triu(B)$	$triu(A^+), triu(A), L, U$

3. Updates in matrix-free environment

Table 1: *Sequence from structural mechanics problem of dimension 4.936 solved by preconditioned GMRES(40).*

ILUT(0.001,20), Psize $\approx$ 404 000						
Matrix	Recomp		Freeze		Updated	
	its	fevals	its	fevals	its	fevals
$A^{(0)}$	187	89	187	89	187	89
$A^{(1)}$	89	89	393	0	146	25
$A^{(2)}$	126	89	448	0	182	25
$A^{(3)}$	221	89	480	0	184	25
$A^{(4)}$	234	89	513	0	190	25
$A^{(5)}$	193	89	487	0	196	25
$A^{(6)}$	178	89	521	0	196	25
$A^{(7)}$	246	89	521	0	196	25
overall fevals	2 186		3 639		1 966	

3. Updates in matrix-free environment

An alternative strategy circumvents estimation of A^+:

Let the matvec be replaced with a function evaluation

$$A^+ \cdot v \quad \rightarrow \quad F^+(v), \quad F^+ : \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

e.g. in Newton's method

$$J(x^+) \cdot v \quad \approx \quad \frac{F(x^+ + h\|x^+\|v) - F(x^+)}{h\|x^+\|} \equiv F^+(v).$$

We assume separable function components, i.e. we assume it is possible to compute the components $F_i^+ : \mathbb{R}^n \rightarrow \mathbb{R}$,

$$F_i^+(v) = e_i^T F^+(v)$$

at the cost of about one nth of the full function evaluation $F^+(v)$.

Then the following strategy can be beneficial:

3. Updates in matrix-free environment

- The forward solves with L in $M^+ = L(DU - D_B - U_B)$ are trivial.
- For the backward solves, use a mixed explicit-implicit strategy: Split

$$DU - D_B - U_B = DU - triu(A) + triu(A^+)$$

in the explicitly given part

$$X \equiv DU - triu(A)$$

and the implicit part $triu(A^+)$.

We then have to solve the upper triangular systems

$$\left(X + triu(A^+)\right) z = y,$$

yielding the standard backward substitution cycle

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

3. Updates in matrix-free environment

In

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

the sum $\sum_{j>i} a^+_{ij} z_j$ can be computed by the function evaluation

$$\sum_{j>i} a^+_{ij} z_j = e_i^T A^+ (0, \ldots, 0, z_{i+1}, \ldots, z_n)^T \approx F_i^+ \left((0, \ldots, 0, z_{i+1}, \ldots, z_n)^T \right).$$

The diagonal $\{a^+_{11}, \ldots, a^+_{nn}\}$ can be found by computing

$$a^+_{ii} = F_i^+(e_i), \qquad 1 \le i \le n.$$

Summarizing, we have the cost comparison:

type	initialization	solve step	memory
Recomp	$est(A^+)$, $A^+ \approx L^+ U^+$	solves with L^+, U^+	L^+, U^+
Update	$est(diag(A^+))$	solves with L, U, $triu(B)$, eval($\mathcal{F}^+$)	L, U

3. Updates in matrix-free environment

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

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

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 $R = 50$ and a 91×91 grid.
- We use a Newton-type method and solve the resulting linear systems with BiCGSTAB with right preconditioning.
- We use a flexible stopping criterion.
- Fortran implementation (embedded in the UFO - software for testing nonlinear solvers).

3. Updates in matrix-free environment

Table 2: *Sequence from nonlinear convection-diffusion problem of dimension 8 281 with Reynolds number 50 solved with preconditioned BiCGStab with flexible stopping criterion. The reference preconditioner is ILU(0).*

	Freeze	Recomp.	Lower tr. update	Upper tr. update
linear solver iterations	410	122	153	186
Newton iterations	9	9	9	9
overall time in seconds	4.39	4.29	2.25	2.73

For more details see:

- DUINTJER TEBBENS J, TŮMA M: *Preconditioner Updates for Solving Sequences of Linear Systems in Matrix-Free Environment*, submitted to NLAA in 2008.
- 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: *Improving Triangular Preconditioner Updates for Nonsymmetric Linear Systems*, LNCS vol. 4818, pp. 737–744, 2007.
- 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 KJB100300703 of the grant agency of the Academy of Sciences of the Czech Republic.