Matrix-free Preconditioning of Sequences of 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

and Technical University in Liberec

3rd Joint Workshop on Krylov Subspace Methods,

Institute of Computer Science, Prague, May 15, 2009.

Outline

1. Brief overview of useful preconditioning strategies in matrix-free environment

2. Approximate preconditioner updates for sequences of linear systems in matrix-free environment

1. Matrix-free preconditioning

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

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

Implementations of Krylov subspace methods where the system matrix is not stored are called *matrix-free*.

Standard example: Solving a *nonlinear* system of equations with a Newton-type method.

Solving the nonlinear system amounts to minimizing

$$\min_x \|F(x)\|, \quad F : \mathbb{R}^n \to \mathbb{R}^n.$$

1. Matrix-free preconditioning

In every Newton iteration one solves a linear system 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 (approximate) 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.

Preconditioning is often crucial for satisfactory convergence of Krylov subspace methods.

Question: What preconditioners can be used in matrix-free environment and how can they be used?

1. Matrix-free preconditioning

Some preconditioners lend themselves very well for matrix-free environment:

- Inner-outer iterations: We solve the preconditioned system

$$PAx = Pb,$$

  where the product Pv approximates $A^{-1}v$ by performing a small number of the Krylov subspace method itself. If the method is matrix-free, then so is the preconditioning.

- Toeplitz, circulant or other structured preconditioners need not be stored fully, their generators suffice. For instance, the product Pv can consist of multiplication with a Toeplitz matrix done with the help of discrete FFT.

- The product Pv can also be performed as a Multigrid or Additive Schwarz sweep

1. Matrix-free preconditioning

- Deflation-based preconditioners are matrix-free when the Krylov subspace method is. For instance, for GMRES the Arnoldi decomposition

$$AV_k = V_k H_k + h_{k,k+1} v_{k+1} e_k^T$$

is generated matrix-free. Ritz values and vectors are obtained from the eigenpairs of H_k:

$$H_k y_j = \sigma_j y_j \quad \Rightarrow \quad \{\sigma_j, V_k y_j\}$$

Deflation-based preconditioners are defined through projecting away unwanted eigenspaces and have the compact form

$$P \equiv I - W_k W_k^T,$$

for some approximate invariant subspace W_k of dimension k (see e.g. [Burrage, Erhel, Pohl - 1996], [Frank, Vuik - 2000], [Loghin, Ruiz, Touhami - 2004]).

1. Matrix-free preconditioning

Well-known preconditioners like ILU, IC, AINV based on incomplete decomposition need the system matrix *explicitly* $\Rightarrow$ 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

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

1. Matrix-free preconditioning

In general, one uses a graph coloring algorithm to estimate a matrix A whose sparsity pattern is given:

- Consider the so-called *intersection graph* $G(A^T A)$ with vertices $\{1, \ldots, n\}$ and edges $\{\{i, j\} | (A^T A)_{ij} \neq 0\}$
- The graph coloring algorithm colors each edge such that no two adjacent vertices have the same color
- Each group of vertices of the same color yields a matvec to find the entries of the corresponding columns

Graph coloring algorithms try to minimize the number of colors, i.e. the number of matvecs needed estimation. This is an NP-hard problem.

Note: Often a good approximation of A obtained with few matvecs suffices to construct a good preconditioner [Cullum, Tůma - 2006].

1. Matrix-free preconditioning

The estimation of the system matrix and the *explicit* storage of an incomplete decomposition are somehow in contradiction with the philosophy of matrix-free environment. Nevertheless this is done in many applications where incomplete decompositions are the most robust preconditioners available.

How about estimation of only a small part of A ? A simple example is a Jacobi preconditioner which needs only the main diagonal, e.g. in

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

Can the main diagonal be estimated more cheaply than the whole matrix ?

1. Matrix-free preconditioning

Notation: Let the matvec Av be performed with a function evaluation,

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

e.g. in Newton's method

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

and let the individual components $G_i : \mathbb{R}^n \rightarrow \mathbb{R}$ of G be defined through

$$G_i(v) = e_i^T G(v).$$

Then a straightforward idea is to compute the main diagonal $\{a_{11}^+, \ldots, a_{nn}^+\}$ by the function *component* evaluations

$$a_{ii} = G_i(e_i), \qquad 1 \leq i \leq n.$$

This makes sense if function *component* evaluations are inexpensive.

1. Matrix-free preconditioning

Definition: We speak of separable function components if it is possible to compute the function components $G_i(v)$ at the cost of about one nth of the full function evaluation $G(v)$.

Function components may not be separable in complicated Finite Element or Finite Volume computations where the contributions for each element or volume are computed simultaneously.

Clearly, with separable function components the Jacobi preconditioners can be computed with

$$a_{ii} = G_i(e_i), \qquad 1 \leq i \leq n.$$

at the cost of only *one* full function evaluation.

1. Matrix-free preconditioning

With separable function components it is also possible to apply Gauss-Seidel preconditioners in a matrix-free fashion. We propose the following, to our knowledge new strategy. Consider first the simple case

$$P = triu(A).$$

- During the initialization, estimate the main diagonal through

$$a_{ii} = G_i(e_i), \qquad 1 \le i \le n.$$

- Solve the upper triangular system $triu(A)z = y$ with the standard cycle

$$z_i = \frac{y_i - \sum_{j>i} a_{ij} z_j}{a_{ii}}, \qquad i = n, n-1, \ldots, 1,$$

  where the implicit sum $\sum_{j>i} a^{+}_{ij} z_j$ can be computed as

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

1. Matrix-free preconditioning

In this way we can solve any triangular system

$$triu(A)z = y$$

without explicit knowledge of the entries of A or $triu(A)$. The costs are

1. n initial function component evaluations for estimation of the main diagonal $\Rightarrow$ about one full function evaluation
2. n function component evaluations for the backward substitution cycle $\Rightarrow$ about one other full function evaluation

Clearly, a system

$$tril(A)z = y$$

can be solved analogously and hence more complex SOR-type preconditioners like

$$P = tril(A)diag(A)^{-1}triu(A),$$

possibly with relaxation parameters, can applied in this matrix-free way.

2. Approximate preconditioner updates

Now consider a general sequence of linear systems

$$A^{(i)}x = b^{(i)}, \quad i = 1, 2, \ldots, \qquad A^{(i)} \in \mathbb{R}^{n\times n},\, b \in \mathbb{R}^n$$

Such sequences arise in numerous applications like CFD problems, operation research problems, Helmholtz equations, . . .

The central question for efficient solution of *sequences* of linear systems will always be:

How can we share part of the computational effort throughout the sequence ?

We consider here only strategies related to preconditioning of Krylov subspace methods.

2. Approximate preconditioner updates

A very simple idea is to freeze the preconditioner [Knoll, Keyes - 2004], e.g. use

$$M^{(1)}A^{(i)}x = M^{(1)}b^{(i)}, \quad i = 1, 2, \ldots$$

for a reference preconditioner $M^{(1)}$.

Naturally, a frozen preconditioner will deteriorate when the system matrix changes too much. One often recomputes a preconditioner periodically for every mth linear system or uses some simple heuristic to determine the moment for recomputing, like

- when the distance between the current matrix and the matrix corresponding to the frozen preconditioner is larger than a tolerance;
- when the number of linear solver iterations grows larger than a tolerance.

2. Approximate preconditioner updates

To enhance the power of a frozen preconditioner one may also use *approximate preconditioner updates*. A few approximate preconditioner updates have been proposed:

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

2. A nonsymmetric preconditioner update

The banded updates in [Benzi, Bertaccini - 2003, 2004] were extended recently [DT, Tůma - 2007] yielding a preconditioner update which is:

- 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.
- Inspired by the banded preconditioner updates in [Benzi, Bertaccini - 2003, 2004]

In the following we focus on the preconditioner updates in [Benzi, Bertaccini - 2003, 2004] and [DT, Tůma - 2007].

2. Approximate preconditioner updates

Notation:

Consider two systems

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

preconditioned by M and M^+ respectively and let

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

In [Benzi, Bertaccini - 2003, 2004] it is assumed that the difference matrix B is diagonal and that we have a triangular factorization

$$M = LDL^H \approx A \qquad \text{or} \quad M = ZD^{-1}Z^T \approx A^{-1}.$$

Then the approximate preconditioner update is required to have the form

$$M^+ = L(D - C)L^H \approx A^+ \qquad \text{or} \quad M^+ = Z(D - C)^{-1}Z^T \approx (A^+)^{-1}$$

for a C such that linear systems with the factor $D - C$ are easily solvable.

2. Approximate preconditioner updates

Consider $C = L^{-1}BL^{-H}$, then

$$A - M = A - LDL^H = A - L(D - L^{-1}BL^{-H})L^H - B = A^+ - M^+,$$

i.e. we have equality of preconditioner accuracies

$$||A - M|| = ||A^+ - M^+||$$

and we therefore hope M^+ to be an update of quality comparable to M.

The factor $D - C = D - L^{-1}BL^{-H}$ can be very dense; *banded* approximations are used instead:

$$M_k^+ = L(D - C_k)L^H, \quad \text{where} \quad C_k = L_k^{-1}BL_k^{-H}$$

and L_k^{-1} is obtained by extracting the $k-1$ lower diagonals of L^{-1}.
Similarly Benzi and Bertaccini define the updates

$$M_k^+ = Z(D - C_k)^{-1}Z^H.$$

2. Approximate preconditioner updates

Now let M be factorized as

$$M = LDU \approx A,$$

then the choice

$$M^+ = LDU - B$$

yields again

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

We will approximate this ideal update $LDU - B$ in two steps. 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. Approximate 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	$A^+ \approx L^+D^+U^+$	solves with L^+, D^+U^+	A^+, L^+, D^+U^+
Update	—	solves with $L, DU, triu(B)$	$A^+, triu(A), L, DU$

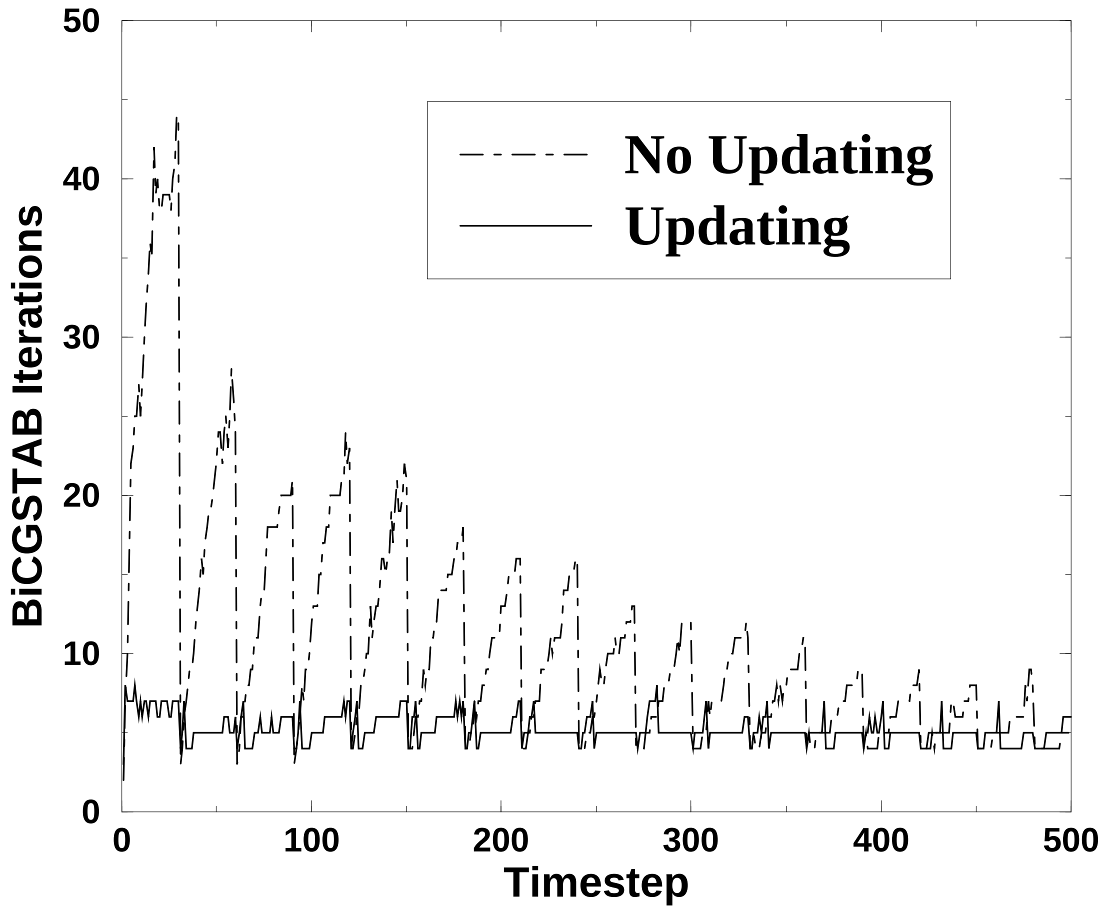
This is the basic idea; more sophisticated improvements are possible.

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 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. Approximate preconditioner updates



BiCGSTAB iterations for the first 500 systems in the cylinder problem

2. Approximate preconditioner updates

Now we return to matrix-free environment.

Recomputing a factorized preconditioner in matrix-free environment 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
- the actual incomplete factorization

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

How about the usage of the preconditioner updates in matrix-free environment?

2. Approximate preconditioner updates

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 to estimate only the upper triangular part
- Can there be taken any advantage of the fact we estimate only the upper triangular part?

2. Approximate preconditioner updates

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 is an example of a *partial graph coloring problem* [Pothen et al. - 2007].

2. Approximate preconditioner updates

As we saw before, 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	$est(A^+)$, $A^+ \approx L^+ D^+ U^+$	solves with L^+, D^+U^+	L^+, D^+U^+
Update	$est(triu(A^+))$	solves with $L, DU, triu(B)$	$L, DU, triu(B)$

2. Approximate preconditioner updates

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	

2. Approximate preconditioner updates

An alternative strategy assumes separable function components.

Notation: Let the matvec with the current matrix be replaced with a function evaluation

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

and let the corresponding components $F_i^+ : \mathbb{R}^n \longrightarrow \mathbb{R}$ satisfy

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

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

Then the following strategy can be beneficial:

- The forward solves with L in $M^+ = L(DU - D_B - U_B)$ are trivial.

2. Approximate preconditioner updates

- For the backward solves with $DU - D_B - U_B$, 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

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

2. Approximate preconditioner updates

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	$est(A^+)$, $A^+ \approx L^+ D^+ U^+$	solves with L^+, D^+U^+	L^+, D^+U^+
Update	$est(diag(A^+))$	solves with L, DU, $triu(A)$, $\mathcal{F}^+$	$L, DU, triu(A)$

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 $R = 50$ 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).

2. Approximate preconditioner updates

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.