Lobachevsky State University of Nizhni Novgorod

Faculty of Computational mathematics and cybernetics

Iterative Methods for Solving Linear Systems

Krylov Subspace Iterative Methods

Supported by Intel

K. A. Barkalov
Software Department

Contents

Problem Statement

- Let us consider a system of n linear equations like

$$\begin{aligned} a_{11}x_1 &+ a_{12}x_2 &&+ \ldots + a_{1n}x_n &&= b_1 \\ a_{21}x_1 &+ a_{22}x_2 &&+ \ldots + a_{2n}x_n &&= b_2 \\ &\ldots \\ a_{n1}x_1 &+ a_{n2}x_2 &&+ \ldots + a_{nn}x_n &&= b_n \end{aligned}$$

- As a matrix, the system may be represented as follows

$$Ax=b$$

- $A=(a_{ij})$ is a $n \times n$ real matrix; b and x are vectors consisting of n elements; let the exact system solution be x^*.
- *An iterative method* generates a sequence of vectors $x^{(s)} \in R^m$, $s=0,1,2,\ldots$, where $x^{(s)}$ is an approximate system solution.

Properties of Iterative Methods

- Iterative method is convergent if

$$\forall\ x^{(0)} \in R^m \quad \lim_{s \to \infty} \left\| x^{(s)} - x^* \right\| = 0$$

- Iterative method stop criteria: accuracy and number of iterations
  - Stop, if $\|x^{(s)} - x^{(s-1)}\| \le \varepsilon_1$. In this case, $\varepsilon_2 = \|x^{(s)} - x^{(s-1)}\|$ is the attainable method accuracy.
  - Stop, if $\|r^{(s)}\| \le \varepsilon_1$. In this case, $\varepsilon_2 = \|r^{(s)}\|$ is the attainable method accuracy.
  - Stop, if $s = N$. $x^{(N)}$ is understood as an obtained solution. The maximum number of iterations N is predefined.
- From this on, let us suppose that A is a real square matrix.

Idea of Projective Methods

1. The problem of linear system solution projection to any subspace K orthogonally to $L=AK$ is equivalent to minimization of the functional $\Phi(x) = \|r(x)\|^2$ of the space K.
2. If $A=A^T$, $A>0$, the problem of linear system solution projection to any subspace K orthogonally to K is equivalent to minimization of the functional $\Phi(x) = \|x - x^*\|_A^2$ of the space K.

Here, $\|x\|_A = \sqrt{(Ax, x)}$

Which subspace K to select?

Krylov Subspace

- The Krylov subspace of dimension m generated by the vector v and matrix A is the linear space

$$K_m = K_m(A, v) = span\left\{v, Av, A^2v, ..., A^{m-1}v\right\}$$

- For the methods $v = r_0 / \| r_0 \|$ is usually selected, where $r_0 = b - Ax_0$ is the residual of the first approximation x_0.
- Having constructed the Krylov subspace, we can adapt the initial system to it; the solution of the new system will consist in approximation to the initial system solution.
- The system must be easy to project and, as a result, to solve.
- The problem is to construct the subspace basis $K_m(A, v)$

Arnoldi Orthogonalization

- The objective is to find the orthogonal basis K_m, i. e. vectors $\{v_1, v_2, \ldots, v_m\}$
- Parameters: first vector v_1 ($\|v_1\|=1$) and dimension m.

Algorithm

for $j = 1, ..., m$ do

$w = Av_j$ //next vector

for $i = 1, \ldots, j$ do

$h_{ij} = (w, v_i)$

$w = w - h_{ij} v_i$

end i

$h_{j+1,j} = \|w\|$; $v_{j+1} = w / h_{j+1,j}$

end j

Arnoldi Orthogonalization

- As a result of the Arnoldi algorithm, we obtain:

  $V_m=[v_1,v_2,\ldots, v_m]$ is the orthonormal basis of the subspace K_m.

  $\overline{V}_m = [v_1, v_2, ..., , v_m, v_{m+1}]$ – matrix expanded by the last computed vector v_{m+1}.

  H_m – orthogonalization coefficient matrix.

$$H_m = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & \cdots & h_{1m} \\ h_{21} & h_{22} & h_{23} & h_{24} & \cdots & h_{2m} \\ 0 & h_{32} & h_{33} & h_{34} & \cdots & h_{3m} \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & h_{m-1,m-2} & h_{m-1,m-1} & h_{m-1,m} \\ 0 & \cdots & 0 & 0 & h_{m,m-1} & h_{m,m} \end{bmatrix}$$

H_m – Hessenberg (upper) matrix

Arnoldi Orthogonalization

$\overline{H}_m$ – orthogonalization coefficient matrix expanded by a zero string with the coefficient $h_{m+1,m}$.

$$\overline{H}_m = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & \cdots & h_{1m} \\ h_{21} & h_{22} & h_{23} & h_{24} & \cdots & h_{2m} \\ 0 & h_{32} & h_{33} & h_{34} & \cdots & h_{3m} \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & h_{m-1,m-2} & h_{m-1,m-1} & h_{m-1,m} \\ 0 & \cdots & 0 & 0 & h_{m,m-1} & h_{m,m} \\ 0 & \cdots & & & 0 & h_{m+1,m} \end{bmatrix}$$

Generalized Minimal Residual Method

1. Select as e_1 (first normal basic vector)
   $v_1 = r_0 / \beta \quad (\beta = \| r_0 \| \ , \ r_0 = b - Ax_0)$.
2. Execute m steps of the Arnoldi algoritm (to obtain V_m, H_m)

for $j = 1, ..., m$ do

$w = Av_j$ //next vector

for $i = 1, \ldots, j$ do

$h_{ij} = (w, v_i)$

$w = w - h_{ij} v_i$

end i

$h_{j+1,j} = \|w\|; \ v_{j+1} = w / h_{j+1,j}$

end j

3. Compute $x_m = x_0 + V_m y_m$, where $y_m = \arg \min_y \left\| \beta e_1 - \overline{H}_m y \right\|$

GMRes – Implementation

- How to solve $y_m = \arg\min_y \left\| \beta e_1 - \overline{H}_m y \right\|$?
- Equivalent formulation $\overline{H}_m y = \beta e_1$
  redefined linear system, matrix size $(m+1)\times m$.
  May be solved as a least squares problem.
- $\overline{H}_m$ is a Hessenberg matrix; simplify it to a triangle using givens rotations.
- At the same time, r_i can be computed without computing x_i explicitly

GMRes – Implementation

- Define the rotation matrix Ω_i
  rotation coefficients c_i, s_i
  are selected so that
  $\eta_{i+1,i}$ is zeroed.
  At the same time, $c_i^2 + s_i^2 = 1$
- Multiply the system
  $$\overline{H}_m y = \beta e_1$$
  by the matrix Ω_i on the left.

$$\Omega_i = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix}$$

GMRes – Implementation

- Example:

$$\overline{H}_4 = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ & h_{32} & h_{33} & h_{34} \\ & & h_{43} & h_{44} \\ & & & h_{54} \end{bmatrix} \qquad \overline{g}_0 = \beta e_1 = \begin{bmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Multiply by Ω_1, where

$$s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}} \qquad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}}$$

$$\Omega_1 = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

GMRes – Implementation

- Thus, we obtain

$$\overline{H}_4^{(1)} = \begin{bmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} \\ & h_{32} & h_{33} & h_{34} \\ & & h_{43} & h_{44} \\ & & & h_{54} \end{bmatrix} \qquad \overline{g}_1 = \begin{bmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Similarly, multiply by $\Omega_2, \Omega_3, \ldots$, to obtain

$$\overline{H}_4^{(4)} = \begin{bmatrix} h_{11}^{(4)} & h_{12}^{(4)} & h_{13}^{(4)} & h_{14}^{(4)} \\ & h_{22}^{(4)} & h_{23}^{(4)} & h_{24}^{(4)} \\ & & h_{33}^{(4)} & h_{34}^{(4)} \\ & & & h_{44}^{(4)} \\ & & & 0 \end{bmatrix} \qquad \overline{g}_4 = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \end{bmatrix}$$

GMRes – implementation

- Set $Q_m = \Omega_m \cdot \Omega_{m-1} \cdot \ldots \cdot \Omega_1$, then

$$\overline{R}_m = \overline{H}_m^{(m)} = Q_m \overline{H}_m$$

$$\overline{g}_m = Q_m (\beta e_1) = (\gamma_1, \ldots, \gamma_m, \gamma_{m+1})^T$$

- As Q_m is a unitary matrix,

$$\min_y \left\| \beta e_1 - \overline{H}_m y \right\| = \min_y \left\| \overline{g}_m - \overline{R}_m y \right\|$$

- Delete the last line of the system and solve the triangular system

for y. $R_m y_m = g_m$

GMRes – Implementation

- We can prove that:
  - The rank AV_m is equal to the rank R_m.
  - The vector $y_m = \arg\min_y \left\| \beta e_1 - \overline{H}_m y \right\|$ can be obtained by solving the system $R_m y_m = g_m$, i. e. $y_m = (R_m)^{-1} g_m$
  - The residual at the step m satisfies the following relations

$$r_m = b - Ax_m = \overline{V}_m [\beta e_1 - \overline{H}_m y_m] = \overline{V}_m Q_m^T (\gamma_{m+1} e_{m+1})$$

  - As a result, $\left\| r_m \right\| = \left| \gamma_{m+1} \right|$
- The indicated value may be used as a method stop criterion $\left\| r_m \right\| < \varepsilon$ and ensure such a verification at each step of the algorithm.

GMRes(m) Restart

- As *m* increases, the memory requirements and complexity of the next method iteration grow dramatically.
- The solution consists in restarting the method every *m* iterations

1. Compute $v_1 = r_0 / \beta$ $(\beta = \| r_0 \|, r_0 = b - Ax_0)$.
2. Arnoldi algorithm: compute V_m, H_m, $\overline{V}_m$, $\overline{H}_m$
3. $y_m = \arg \min_y \left\| \beta e_1 - \overline{H}_m y \right\|$
4. $x_m = x_0 + V_m y_m$
5. If $\left\| r_m \right\| < \varepsilon$ then STOP
   else $x_0 = x_m$, go to step 1.

Preconditioned Method

1. Select as $e_1 = v_1$, where $v_1 = r_0 / \beta$
   ($\beta = \| r_0 \|$, $r_0 = M^{-1}(b - Ax_0)$).
2. Execute m steps of the Arnoldi algoritms (to obtain V_m, H_m)

for $j = 1, ..., m$ do

$w = M^{-1} A v_j$ //next vector

for $i = 1, \ldots, j$ do

$h_{ij} = (w, v_i)$

$w = w - h_{ij} v_i$

end i

$h_{j+1,j} = \|w\|$; $v_{j+1} = w / h_{j+1,j}$

end j

3. Compute $x_m = x_0 + V_m y_m$, where $y_m = \arg \min_y \left\| \beta e_1 - \overline{H}_m y \right\|$

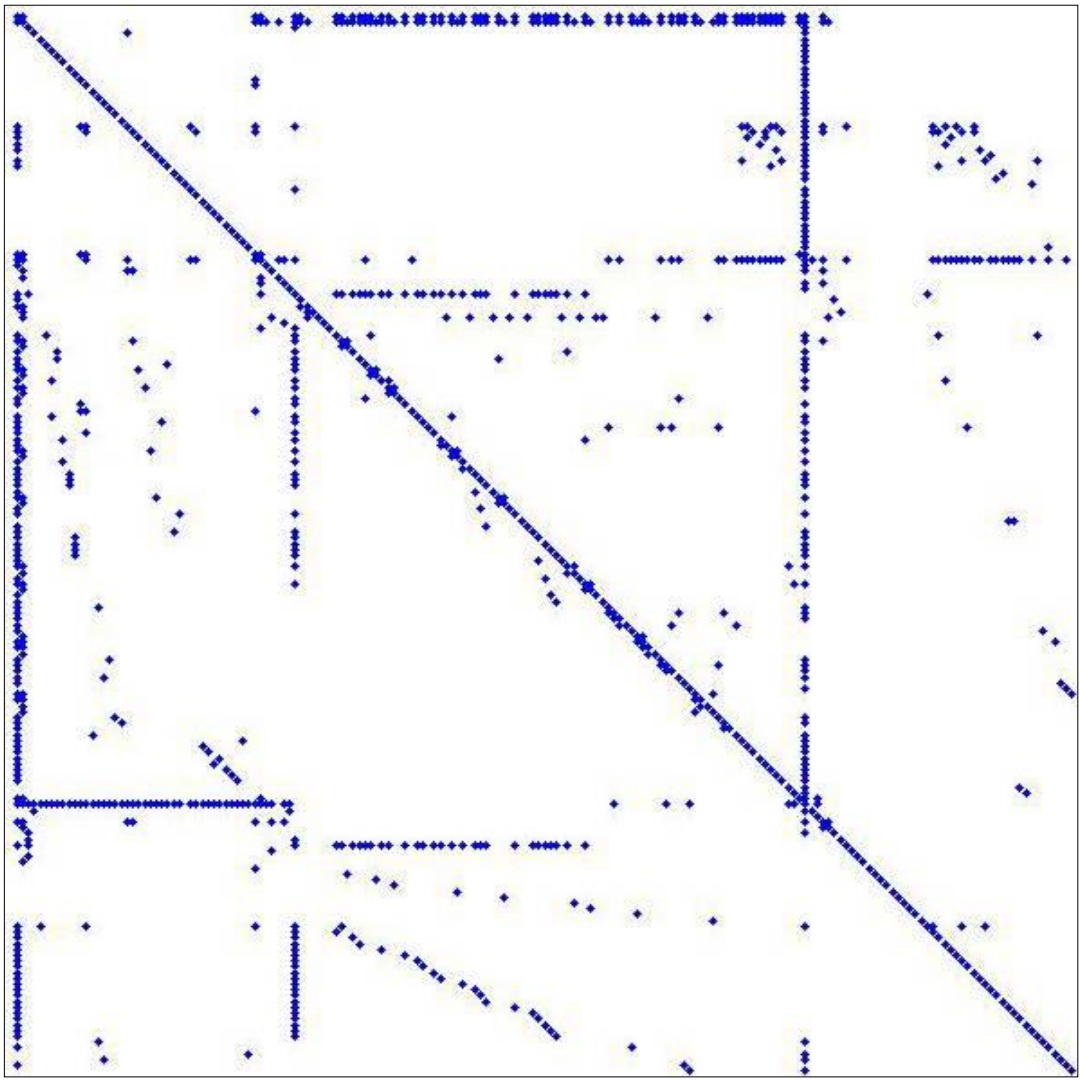
Experimental Results (Sparse)

- The University of Florida Sparse Matrix Collection
  http://www.cise.ufl.edu/research/sparse/matrices/
- Parameters of the matrices involved

Name	n	nz	μ_A
fs_183_1	183	998	$2{,}2 \cdot 10^{13}$
fs_541_1	541	4282	4467
sherman2	1080	23094	$9{,}6 \cdot 10^{11}$
watt_1	1856	11360	$4{,}4 \cdot 10^{9}$
cage10	11397	150645	11

Experimental Results (Sparse)

- Matrix pattern
  fs_183_1
- Exact solution
  $x^* = \{x_i = 1, 1 \le i \le n\}$
- Right side
  $b = Ax^*$
- System
  $Ax=b$



Experimental Results (Sparse)

- Method accuracy $\varepsilon = 10^{-8}$
- ILU(0)-preconditioner

Matrix name	n	s	
		GMRes	PGMRes
fs_183_1	183	78	16
fs_541_1	541	13	2
sherman2	1080	100	18
watt_1	1856	40	48
cage10	11397	24	7

- The use of preconditioner reduces the number of iterations (for most problems)

Lancsoz Biorthogonalization

- The vector systems $\{v_1,\ldots,v_m\}$ and $\{w_1,\ldots,w_m\}$ are biorthogonal if $(v_i,w_j)=0$ when $i\neq j$.
- The systems are biorthonormal if $(v_i,w_j)=\delta_{i\varphi}$.
- Let us set $V=[v_1,\ldots,v_m]$, $W=[w_1,\ldots,w_m]$, so the biorthonormality condition is equivalent to

$$W^T V = E.$$

- Let X be the basis of K and Y the basis of L, so, similar to the orthogonalization procedure, the biorthogonalization algorithm may be proposed to construct V and W, i. e.
the biorthonormal bases of K and L.

Lancsoz Bi-orthogonalization– Algorithm

- Let $\delta_1=\beta_1=0$, $v_0=w_0=0$,
- Select vectors v_1, w_1 so that $(v_1, w_1) = 1$

for $j=1,\ldots,m$ do

$$\alpha_j = (Av_j, w_j)$$

$$\overline{v}_{j+1} = Av_j - \alpha_j v_j - \delta_j v_{j-1} \qquad \overline{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$$

$$\delta_{j+1} = \sqrt{|(\overline{v}_{j+1}, \overline{w}_{j+1})|} \quad \text{If} \quad \delta_{j+1} = 0 \quad \text{then STOP}$$

$$\beta_{j+1} = (\overline{v}_{j+1}, \overline{w}_{j+1}) / \delta_{j+1}$$

$$w_{j+1} = \overline{w}_{j+1} / \beta_{j+1} \qquad v_{j+1} = \overline{v}_{j+1} / \delta_{j+1}$$

end j

Lancsoz Bi-orthogonalization – Properties

- Bi-orthogonalization coefficients $\alpha_i, \beta_i, \delta_i$ form the tridiagonal matrix T_m
- Similar to the Arnoldi method:

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T$$

$$AW_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T$$

$$W_m^T A V_m = T_m$$

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & & & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{bmatrix}$$

- Iterative process:

$$x_m = x_0 + V_m T_m^{-1} (\beta e_1)$$

- Directions will be bi-conjugate - use biorthogonalization instead of explicit computation of x_m.

Bi-conjugate Gradient Method

Algorithm

- $r_0 = b - Ax_0$, select $\bar{r}_0$ so that $(r_0, \bar{r}_0) \neq 0$ (e.g. $\bar{r}_0 = r_0$)
- Let $p_0 = r_0 \quad \bar{p}_0 = \bar{r}_0$

for j=0,1,…, n do

$$\alpha_j = (r_j, \bar{r}_j) / (Ap_j, \bar{p}_j) \;,\; x_{j+1} = x_j + \alpha_j p_j$$

$$r_{j+1} = r_j - \alpha_j A p_j \;,\; \bar{r}_{j+1} = \bar{r}_j - \alpha_j A^T \bar{p}_j$$

$$\beta_j = (r_{j+1}, \bar{r}_{j+1}) / (r_j, \bar{r}_j)$$

if β_j=0 or $\| r_{j+1} \| < \varepsilon$ then STOP

$$p_{j+1} = r_{j+1} + \beta_j p_j \;,\; \bar{p}_{j+1} = \bar{r}_{j+1} + \beta_j \bar{p}_j$$

end j

Preconditioned Method

Algorithm with the preconditioner *M*

- $r_0 = b - Ax_0$, select $\bar{r}_0$ so that $(r_0, \bar{r}_0) \neq 0$ (e.g. $\bar{r}_0 = r_0$)
- Solve $Mz_0 = r_0$, $M^T \bar{z}_0 = \bar{r}_0$. Let $p_0 = z_0$ $\bar{p}_0 = \bar{z}_0$

  for j=0,1,…, n do

  $$\alpha_j = (z_j, \bar{r}_j) / (Ap_j, \bar{p}_j) \ , \ x_{j+1} = x_j + \alpha_j p_j$$

  $$r_{j+1} = r_j - \alpha_j Ap_j \ , \ \bar{r}_{j+1} = \bar{r}_j - \alpha_j A^T \bar{p}_j$$

  $$Mz_{j+1} = r_{j+1} \quad M^T \bar{z}_{j+1} = \bar{r}_{j+1}$$

  $$\beta_j = (z_{j+1}, \bar{r}_{j+1}) / (z_j, \bar{r}_j)$$

  if β_j=0 or $\| r_{j+1} \| < \varepsilon$ then STOP

  $$p_{j+1} = r_{j+1} + \beta_j p_j \ , \ \bar{p}_{j+1} = \bar{r}_{j+1} + \beta_j \bar{p}_j$$

  end j

Experimental Results (Sparse)

- The University of Florida Sparse Matrix Collection
  http://www.cise.ufl.edu/research/sparse/matrices/
- Parameters of the matrices involved

Name	n	nz	μ_A
fs_183_1	183	998	$2{,}2 \cdot 10^{13}$
fs_541_1	541	4282	4467
sherman2	1080	23094	$9{,}6 \cdot 10^{11}$
watt_1	1856	11360	$4{,}4 \cdot 10^{9}$
cage10	11397	150645	11

Experimental Results (Sparse)

- Method accuracy $\varepsilon = 10^{-4}$
- ILU(0)-preconditioner

Matrix name	n	s	
		GMRes	PGMRes
fs_183_1	183	183	10
fs_541_1	541	6	1
sherman2	1080	1080	20
watt_1	1856	839	36
cage10	11397	10	3

- More iterations than in GMRes
- Less memory requirements

Lancsoz Symmetric Method

- The matrix A is symmetric. Then, for the Arnoldi method, $H_m = V_m^T A V_m$ must be symmetric
- H_m: Hessenberg+symmetric=tridiagonal
  of the vector v in the course of orthogonalization satisfies the following

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1} .$$

- Orthogonalization process is simplified

$$H_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & & \ddots & & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix}$$

Lancsoz Symmetric Method - Algorithm

- Let $\beta_1=0$, $v_0=0$,
- Select the vector v_1, $\|v_1\| = 1$

for $j=1,\ldots,m$ do

$$w_j = Av_j - \beta_j v_{j-1}$$

$$\alpha_j = (v_j, w_j)$$

$$w_j = w_j - \alpha_j v_j$$

$$\beta_{j+1} = \|w_j\| \quad \text{If} \quad \beta_{j+1} = 0 \quad \text{then STOP}$$

$$v_{j+1} = w_j / \beta_{j+1}$$

end j

Conjugate gradient method (CG)

Algorithm

- Preliminary step $r_0 = p_0 = b - Ax_0$
- Main steps

for i=0,…, n do

$$\alpha_i = \frac{(r_i, r_i)}{(Ap_i, p_i)} \qquad x_{i+1} = x_i + \alpha_i p_i$$

$$r_{i+1} = r_i - \alpha_i A p_i$$

$$\beta_i = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)} \qquad p_{i+1} = r_{i+1} + \beta_i p_i$$

end j

Preconditioned Method (PCG)

Algorithm

- Preliminary step $r_0 = b - Ax_0 \quad z_0 = M^{-1} r_0 \qquad p_0 = z_0$
- Main steps

for i=0,…, n do

$$\alpha_i = \frac{(r_i, z_i)}{(Ap_i, p_i)} \qquad x_{i+1} = x_i + \alpha_i p_i$$

$$r_{i+1} = r_i - \alpha_i A p_i \qquad z_{i+1} = M^{-1} r_{i+1}$$

$$\beta_i = \frac{(r_{i+1}, z_{i+1})}{(r_i, z_i)} \qquad p_{i+1} = z_{i+1} + \beta_i p_i$$

end j

Experimental Results (Sparse)

- The University of Florida Sparse Matrix Collection
  http://www.cise.ufl.edu/research/sparse/matrices/
- Parameters of the matrices involved

Name	n	nz	μ_A
bcsstk01	48	400	882 336
bcsstk05	153	2 423	14 281.1
bcsstk10	1086	22 070	524 225
bcsstk13	2003	83 883	1.06e+10
parabolic_fem	525 825	3 674 625	2.11e+10
tmt_sym	726 713	5 080 961	---

Experimental Results (Sparse)

- Matrix pattern bcsstk13
- Exact solution

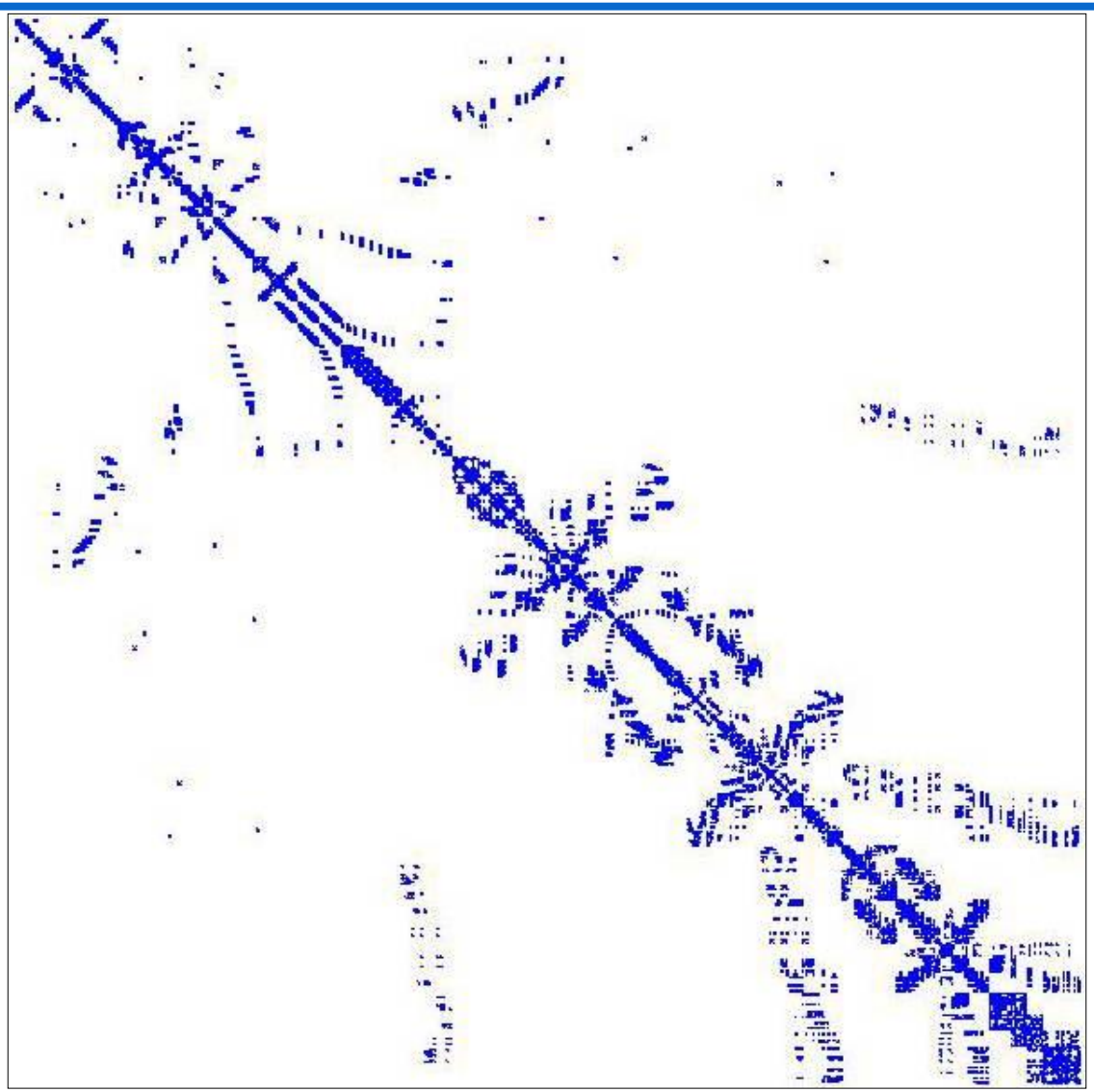
$$x^* = \{x_i = 1, 1 \le i \le n\}$$

- Right side

$$b = Ax^*$$

- System

$$Ax = b$$



Experimental Results (Sparse)

- Method accuracy $\varepsilon = 10^{-7}$
- ILU(0)-preconditioner

Matrix name	n	Accuracy	
		CG	PCG
bcsstk01	48	125	15
bcsstk05	153	272	35
bcsstk10	1086	2335	149
bcsstk13	2003	20031	10016
parabolic_fem	525 825	1690	1045
tmt_sym	726 713	5356	1210

- Convergence is ensured by more than n iterations
- The use of preconditioner reduces the number of iterations

CG – Parallel Algorithm (Dense)

- Iterations are accomplished in a sequence, so it makes sense to parallelize computations performed as part of individual iterations,
  - Basic computations according to the selected method, consist in multiplication of the matrix *A* by the vector *h*,
  - Additional computations (scalar multiplication and addition of vectors) that are less complex include various vector processing operations (scalar product, addition, subtraction, scalar multiplication).
- Known parallel algorithms of matrix multiplication by a vector are used

CG – Parallel Algorithm (Dense)

- Complexity estimation for the parallel $Ah^{(s)}$ operation in case of horizontal band division of the matrix A is

$$2n^2/p + \delta$$

n is the vector length, *p* is the number of flows, δ – contingency

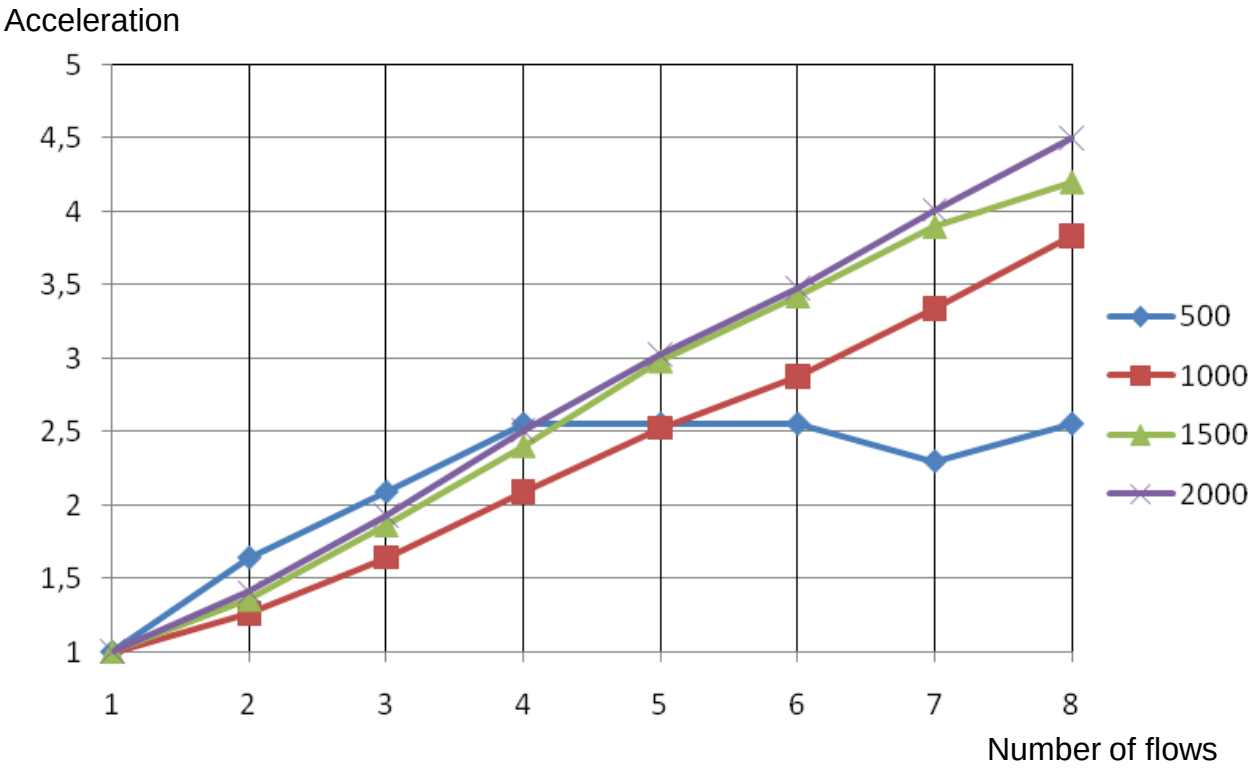
- Less complex computations are subject to single threading.
- Total complexity estimation of the parallel CG method

$$T_p = L\left(\frac{2n^2}{p} + 13n + \delta\right)$$

where *L* is the number of method iterations.

Experimental Results (Dense)

- Acceleration in relation to the single-flow version



- Effect of insufficient computation load on the flow if N=500.

Conclusion

The lecture gives a review of the following:

- Idea of projective methods
- Krylov subspace, subspace basis
  - Arnoldi orthogonalization
  - Lancsoz biorthogonalization
  - Symmetric Lancsoz algorithm
- Krylov subspace iterative methods
  - Generalized minimal residual method
  - Bi-conjugate gradient method
  - Conjugate gradient method
  - Preconditioning in the method pattern
- Experimental results

References

1. Gene H. Golub, Charles F. Van Loan. Matrix Computations. The John Hopkins University Press, 1996.
2. James W. Demmel. Applied Numerical Linear Algebra. SIAM, 1997.
3. J. Dongarra et al. Templates for the solution of linear systems: building blocks for iterative methods. SIAM, 1994.
4. Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

Web Resources

5. The University of Florida Sparse Matrix Collection – [http://www.cise.ufl.edu/research/sparse/matrices/]
6. Intel Math Kernel Library Reference Manual.

   [http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf].

Authors

- Konstantin Barkalov
  Candidate of Physical and Mathematical Sciences,
  Associate Professor, Software Department, CMC Faculty,
  Nizhny Novgorod State University
  barkalov@vmk.unn.ru

- Anna Pirova and Evgeni Kozinov (education software codes)