Accelerating the spike family of algorithms by solving linear systems with multiple right-hand sides

Βασίλης Καλαντζής

Dept. of Computer Engineering & Informatics
University of Patras

Graduate Program in Computer Science & Technology
July 30, 2014

Outline

Goal and motivation I

Important problem in NLA

Efficient solution of

$$AX = F$$

where $A \in \mathbb{R}^{n \times n}$ is a sparse matrix and $F \in \mathbb{R}^{n \times s}$ is the matrix of right-hand sides.

Applications

Numerous...

- Modeling of heat propagation
- Weather forecasting
- Fluid dynamics
- Quantum chromodynamics (QCD)
- Simulation of electronic circuits
- Analysis of civil structures

Goal and motivation II

The problem is very important...

Numerous methods based on different assumptions...

In this thesis

Goal: To extend the Spike framework by using techniques for the solution of linear systems with multiple right-hand sides.

Characteristics of the problem

The coefficient matrix A can be:

- A general sparse matrix $\rightarrow$ $\mathrm{nnz}(A) = O(n)$
- Banded with semi-bandwidth m (if $m \ll n \rightarrow \mathrm{nnz}(A) = O(n)$).
- The banded coefficient matrix can be:

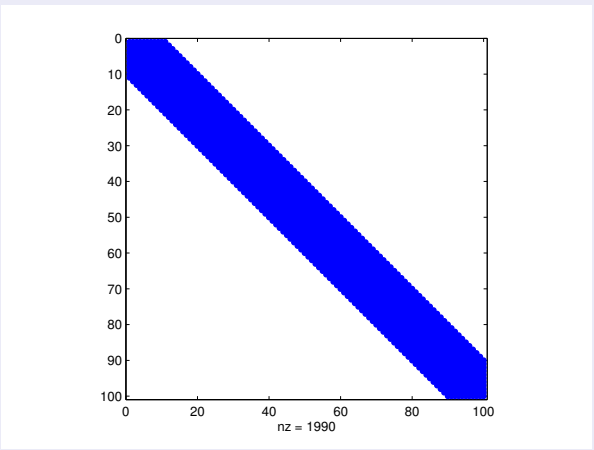


Figure : Dense within the band

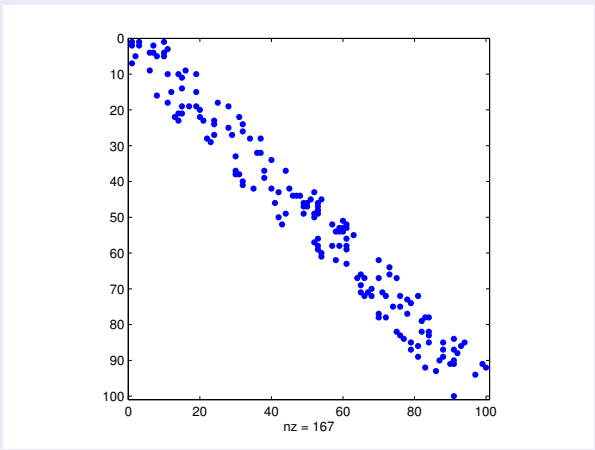
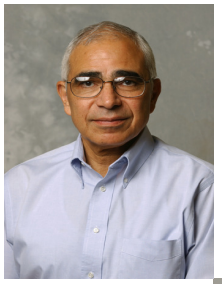


Figure : Sparse within the band

- We focus in symmetric and positive definite (SPD) matrices.

Outline



A parallel hybrid banded system solver: the SPIKE algorithm

Eric Polizzi, Ahmed H. Sameh *

SPIKE: A parallel environment for solving banded linear systems

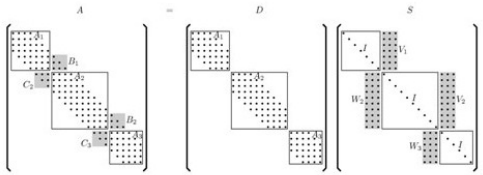
Eric Polizzi, Ahmed Sameh *

Intel® Adaptive Spike-Based Solver

By Henry Gabb (Intel), Mon, Oct 11, 2010

What If Home | Product Overview | Intel® TM ABI specification | Technical Requirements
FAQ | Primary Technology Contacts | Discussion Forum | Blog

Product Overview



SPIKE algorithm

From Wikipedia, the free encyclopedia

The **SPIKE algorithm** is a hybrid parallel solver for banded linear systems developed by Eric Polizzi and Ahmed Sameh.

Contents [hide]

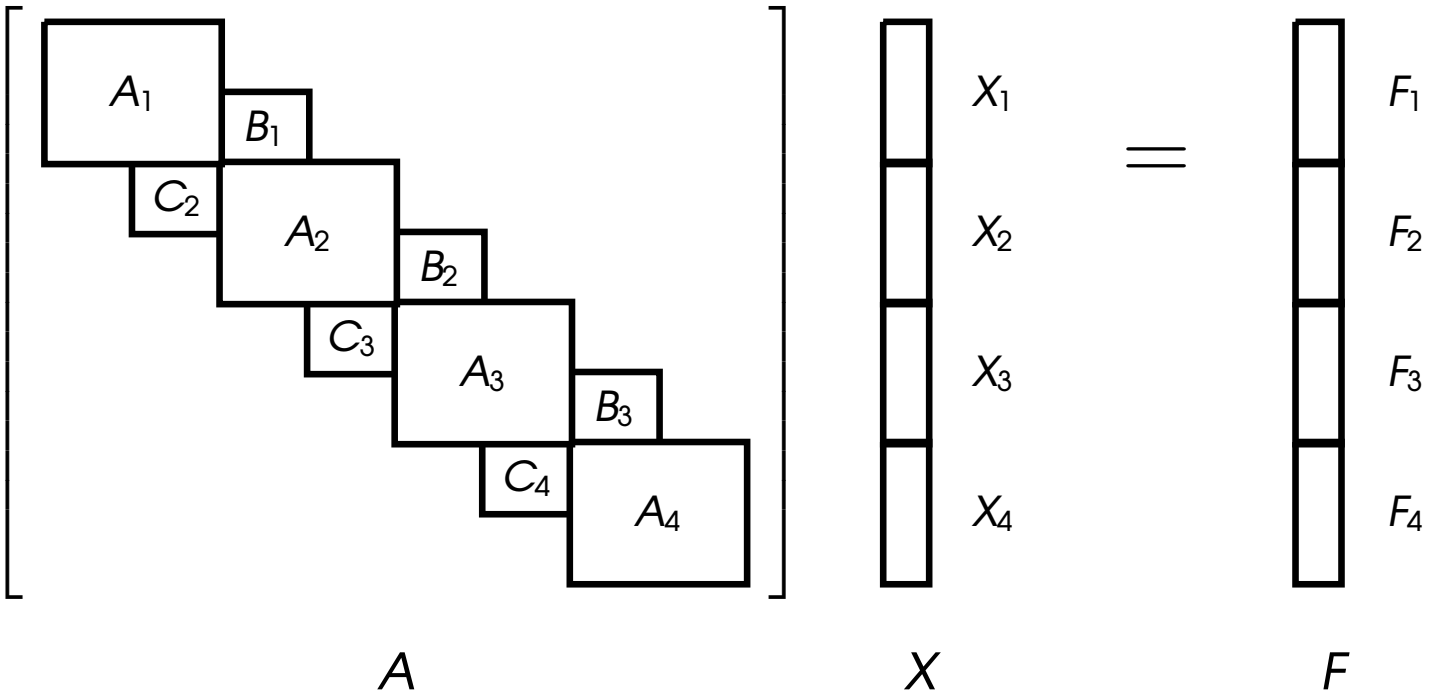
The Spike algorithm I

- The Spike (SamPol11,PolSam06) algorithm is a parallel algorithm for the solution of linear systems of the form

$$AX = F$$

  with banded A of semi-bandwidth m, $m \ll n$.
- Banded matrix $\rightarrow$ Block tridiagonal matrix
- It can be divided in two phases
  - Pre-processing
  - Post-processing

An example for $p = 4$ partitions



$$A_j \in \mathbb{R}^{n_j \times n_j},\ F_j \in \mathbb{R}^{n_j \times s},\ B_j,\ C_j \in \mathbb{R}^{m \times m}$$

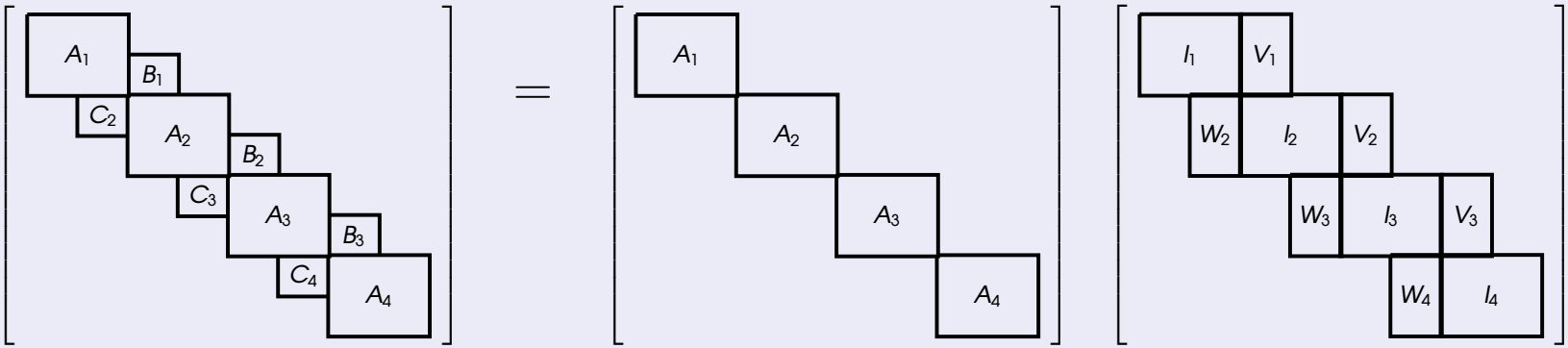
The Spike algorithm II

$A = DS$

- Matrix A is factorized as

$$A = DS \tag{1}$$

where $D \in \mathbb{R}^{n \times n}$ is block diagonal and $S \in \mathbb{R}^{n \times n}$ block tridiagonal.



A D S

It holds that $V_j,\ W_j \in \mathbb{R}^{n/p \times m}$.

The Spike algorithm III

Pre-processing stage

- Solution of the local linear systems

$$A_j[V_j, W_j] = [\hat{B}_j, \hat{C}_j], \quad j = 1, \ldots, p. \tag{2}$$

Notation: $\hat{B}_j = [0_{(n-m)\times m}; B_j]$, $\hat{C}_j = [C_j; 0_{(n-m)\times m}]$

Post-processing stage

- Solution of

$$A_j G_j = F_j, \quad j = 1, \ldots, p. \tag{3}$$

and

$$SX = G. \tag{4}$$

- Solution of (3) is performed locally in each processor.
- Solution of (4) demands communication between neighboring processors.

Solution of $SX = G$ I

Partitioning of V_j, W_j, X_j, G_j

$$V_j = \begin{bmatrix} V_j^{(1)} \\ V_j^{(')} \\ V_j^{(q)} \end{bmatrix}, \qquad W_j = \begin{bmatrix} W_j^{(1)} \\ W_j^{(')} \\ W_j^{(q)} \end{bmatrix}$$

where $V_j^{(1)}$, $V_j^{(')}$, $V_j^{(q)}$ and $W^{(1)}$, $W_j^{(')}$, $W_j^{(q)}$ represent the first m, middle $n_j - 2m$ and last m rows of V_j, W_j respectively. We assume the same partition for X_j, G_j as well.

$S_r X_r = G_r$, example for $p = 3$

$$\begin{bmatrix} I & V_1^{(q)} & 0 & 0 \\ W_2^{(1)} & I & 0 & V_2^{(1)} \\ W_2^{(q)} & 0 & I & V_2^{(q)} \\ 0 & 0 & W_3^{(1)} & I \end{bmatrix} \begin{bmatrix} X_1^{(q)} \\ X_2^{(1)} \\ X_2^{(q)} \\ X_3^{(1)} \end{bmatrix} = \begin{bmatrix} G_1^{(q)} \\ G_2^{(1)} \\ G_2^{(q)} \\ G_3^{(1)} \end{bmatrix}. \tag{5}$$

Solution of $SX = G$ II

Intermediate blocks of rows are computed locally as

$$\begin{cases} X_1^{'} = G_1^{'} - V_1^{'} X_2^{(q)} & j = 1 \\ X_j^{'} = G_j^{'} - V_j^{'} X_{j+1}^{(q)} - W_j^{'} X_{j-1}^{(1)} & 2 \leq j \leq p-1 \\ X_p^{'} = G_p^{'} - W_p^{'} X_{p-1}^{(1)} & j = p \end{cases}$$

Outline

The Spike algorithm and the solvers for mrhs I

Local linear systems in Spike algorithm

- The Spike algorithm leads to the solution of a linear system of the form

$$A_j[V_j, W_j, G_j] = [\hat{B}_j, \hat{C}_j, F_j]. \tag{6}$$

Each local linear system contains $2m + s$ right-hand sides

Obvious approach: Each right-hand side is solved on its own

- Solution of each right-hand side by direct methods (by factorizing A_j)
- Solution of each right-hand side by using iterative methods

Solution methods

Direct methods

- By *LU* decomposition

$$P_j A_j = L_j U_j. \tag{7}$$

  - CC: $O(n_j^3)$
  - Sparse structure? Pardiso, MUMPS, SuperLu...
  - Banded structure? CC: $O(n_j m^2)$

- By Cholesky decomposition

$$A_j = L_j L_j^\top. \tag{8}$$

Iterative methods

- Stationary point methods
- Krylov subspace methods

Krylov subspace iterative methods

Solution of $A_j x = b$

- Solution is sought in the Krylov subspace,

$$\mathcal{K}_\mu(A_j, b) = \text{span}\{b, A_j b, \ldots, A_j^{\mu-1} b\}.$$

- If $q(A_j) = \alpha_0 I_n + \alpha_1 A_j + \ldots + \alpha_{n_j} A_j^{n_j} = 0$, then

$$A_j^{-1} b = \frac{-1}{\alpha_0} \sum_{i=0}^{n_j-1} \alpha_{i+1} A_j^i. \tag{9}$$

- In the i-th iteration

$$x_i \in x_0 + \mathcal{K}_i, \qquad r_i = (b - A_j x_i) \perp \mathcal{C}_i.$$

Different choices for subspace $\mathcal{C}_i$: $\mathcal{C}_i \equiv \mathcal{K}_i$ or $\mathcal{C}_i \equiv A_j \mathcal{K}_i$.

The Spike algorithm and the mrhs solvers II

But...

- ... matrix A_j is the same for each right-hand side

Direct methods

- Factorization is performed only once, $P_j A_j = L_j U_j$
- More right-hand sides $\rightarrow$ better overhead amortization

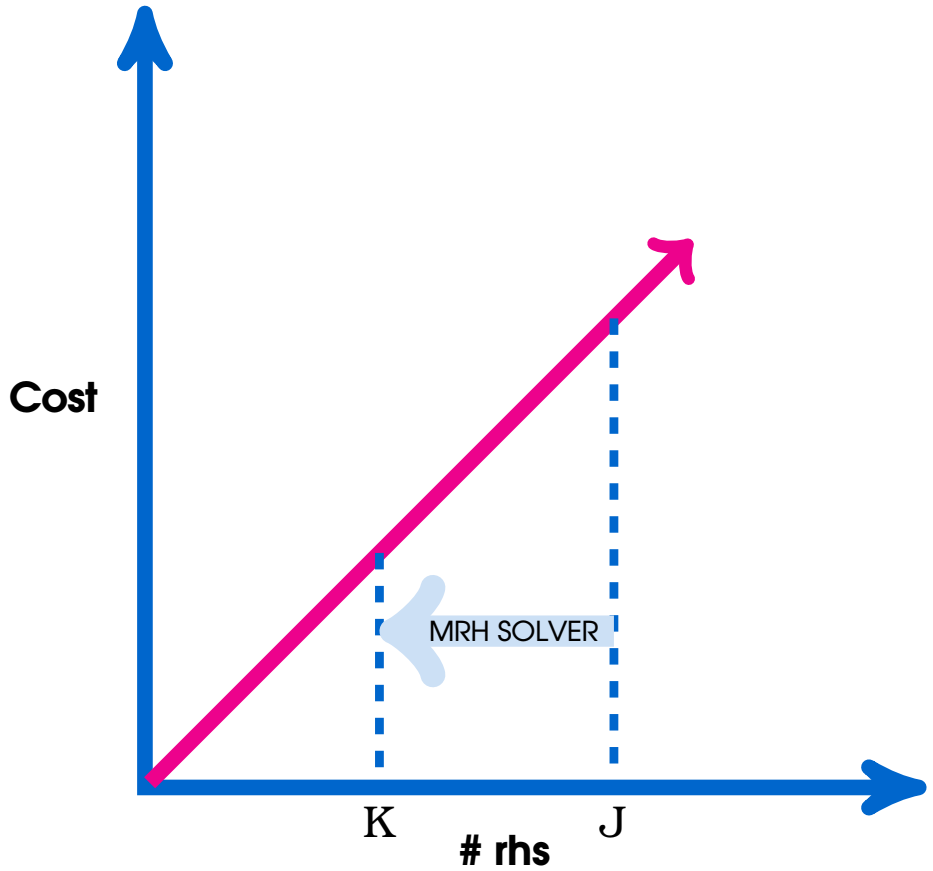
Iterative methods

- Can we employ something similar?

The Spike algorithm and the mrhs solvers III

Goal: Reduce the solution cost of J linear systems

How: Multiple Right Hand Sides Solvers



Ideally $(\mathrm{J} - \mathrm{K}) \longrightarrow \mathrm{J}$

The Spike algorithm and the mrhs solvers IV

A case example

- Suppose solving the following linear system,

$$A[x^{(1)}, x^{(2)}] = [f^{(1)}, f^{(2)}].$$

- First approach: solution of each $Ax^{(i)} = f^{(i)},\ i = 1, 2$ independently (one by one)
- A very simple concept but... if $f^{(2)} = \gamma f^{(1)}$ then $x^{(2)} = \gamma x^{(1)}$!

The Spike algorithm and the mrhs solvers IV

A case example

- Suppose solving the following linear system,

$$A[x^{(1)}, x^{(2)}] = [f^{(1)}, f^{(2)}].$$

- First approach: solution of each $Ax^{(i)} = f^{(i)},\ i = 1, 2$ independently (one by one)
- A very simple concept but... if $f^{(2)} = \gamma f^{(1)}$ then $x^{(2)} = \gamma x^{(1)}$!

MRHs Solvers: Categories

- The above example, although trivial, shows the path
- Right-hand sides can be co-linear, orthogonal...
- Two main classes, a) seed methods and b) block methods

Seed methods for the solution of mrhs problems I

Seed methods - brief description

- Based on an "artificial" augmentation of the search subspace
- Solution of the first right-hand side returns an orthonormal basis $V_i^{(1)}$ of $\mathcal{K}_i(A, r_0^{(1)})$. Then

$$x_0^{(1,j)} = x_0^{(1,j)} + V_i^{(1)} \left(T_i^{(1)}\right)^{-1} \left(V_i^{(1)}\right)^{\top} r_0^{(1,j)} \tag{10}$$

  with

$$T_i^{(1)} = \left(V_i^{(1)}\right)^{\top} A V_i^{(1)}. \tag{11}$$

- Procedure continues till all mrhs are solved
- The j-th right-hand side is projected on the subspaces Krylov $\mathcal{K}_{i_1}(A, r_0^{(1,1)}), \ldots, \mathcal{K}_{i_{j-1}}(A, r_0^{(j-2,j-1)})$

Seed methods for the solution of mrhs problems II

Algorithm 1 Generic algorithm of seed methods (SEED) (SPM89,Saad87,SimG95)

Input: $A \in \mathbb{R}^{n \times n}$, $F \in \mathbb{R}^{n \times s}$, $X_0 \in \mathbb{R}^{n \times s}$, tol
Output: $X \in \mathbb{R}^{n \times s}$

for $j = 1, \ldots, s$ **do**
  for $i = 1, \ldots$ **do**
    the j-th linear system performs the i-th iteration
    for $g = j + 1, \ldots, s$ **do**
      update solution of $x_i^{(j,g)}$
    end for
  end for
end for

Seed methods for the solution of mrhs problems III

Algorithm 2 The Single-Seed Conjugate Gradient algorithm (SSCG) (ChanWan97)

Input: $A \in \mathbb{R}^{n\times n}$, $F \in \mathbb{R}^{n\times s}$, $X_0 \in \mathbb{R}^{n\times s}$, $tol \in \mathbb{R}$
Output: $X \in \mathbb{R}^{n\times s}$

$R_0 = F - AX_0$
$P_0 = R_0$
for $j = 1, \ldots, s$ **do**
  while $\|r_i^{(j)}\|/\|f^{(j)}\| \geq tol$ **do**
    $\alpha = ((r_i^{(j)})^\top r_i^{(j)})/((p_i^{(j)})^\top A p_i^{(j)})$
    $x_{i+1}^{(j)} = x_i^{(j)} + p_i^{(j)}\alpha; \quad r_{i+1}^{(j)} = r_i^{(j)} - Ap_i^{(j)}\alpha$
    $\beta = ((r_{i+1}^{(j)})^\top r_{i+1}^{(j)})/((r_i^{(j)})^\top r_i^{(j)})$
    $p_{i+1}^{(j)} = r_{i+1}^{(j)} + p_i^{(j)}\beta$

    for $k = j+1, \ldots, s$ **do**
      $\eta_i^{(j,k)} = (p_i^{(j)})^\top r_i^{(j,k)}/(p_i^{(j)})^\top A p_i^{(j)}$
      $x_{i+1}^{(j,k)} = x_i^{(j,k)} + \eta_i^{(j,k)} p_i^{(j)}; \quad r_{i+1}^{(j,k)} = r_i^{(j,k)} - \eta_i^{(j,k)} Ap_i^{(j)}$
    end for
  end while
end for

Seed methods for the solution of mrhs problems IV

Theorem (ChanWan97)

Let $F = [f^{(1)}, \ldots, f^{(s)}]$ and suppose that $\mathrm{rank}(F) = k,\ k < s$. Then, there exists α, independent of the number of steps m_p, such that for the residual of the non-solved rhs we have

$$\|r_0^{(j,k)}\| \leq \alpha \sum_{h=1}^{j} \left|\beta_{m_p+1}\right|, \qquad k = j+1, \ldots, s,$$

where value β_{m_p+1} stems from Lanczos if we consider that the h-th rhs is solved by the Lanczos method.

Proof.

See (ChanWan97). □

Block methods for the solution of the mrhs problem

Block methods (Olea80,Gut09,SimG96a,SimG96b)

- A generalization of the classic approach.
- In step i, solution of the j-th right-hand side is sought in the following Krylov subspace,

$$\mathcal{K}_{\mu}^{B}(A, R_0) = \text{span}\{R_0, AR_0, \ldots, A^{\mu-1}R_0\},$$

and

$$x_i^{(j)} \in x_0^{(j)} + \mathcal{K}_i^{B}(A, R_0).$$

- Dimension of the search subspace is at most s times bigger than previously,

$$K_{\mu}^{B}(A, R_0) \subseteq K_{\mu}(A, r_0^{(1)}) + \ldots + K_{\mu}(A, r_0^{(s)}).$$

- Guaranteed to converge in at most n/s steps.

Numerical solution of a Laplacian PDE - discretized by finite differences

$n_x = n_y = n_z = 15$

$AX = F,\ s = 50$

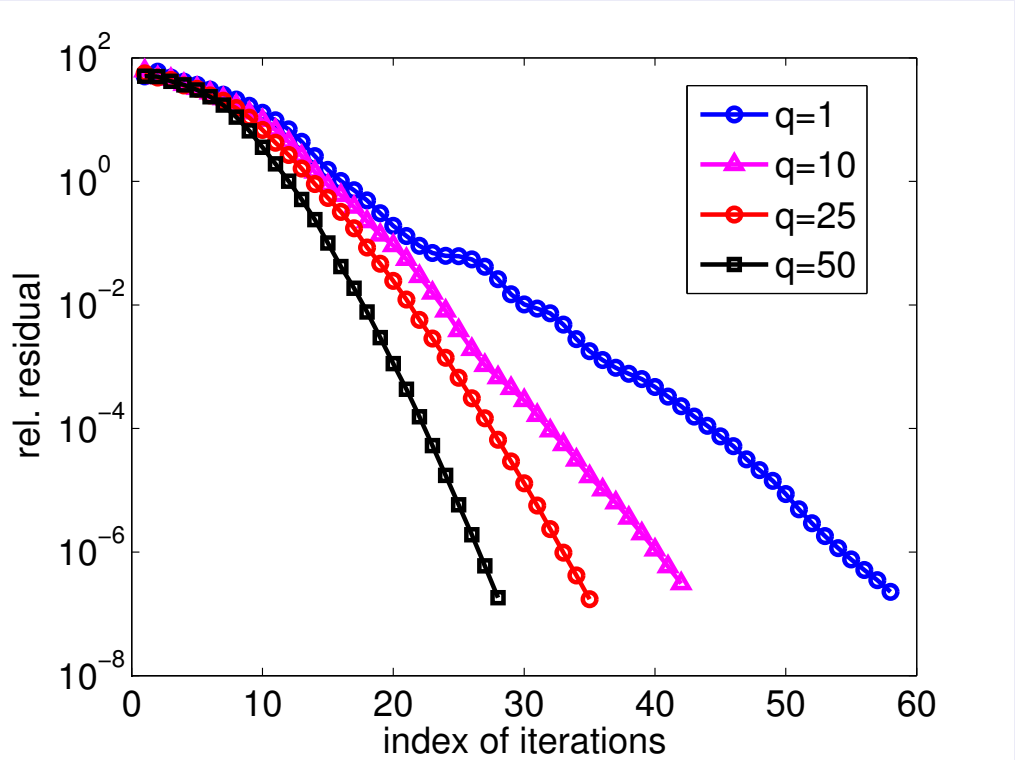


Figure : Norm of relative residual for different blocksizes

An approach based on matrix inversion I

- For an SPD matrix H we have

$$|(H^{-1})_{ij}| \leq C\lambda^{|i-j|}, \ i,j = 1,\ldots,n$$

with

$$\lambda = \left(\frac{\sqrt{\kappa(H)} - 1}{\sqrt{\kappa(H)} + 1} \right)^{2/m}.$$

- Spikes V_j, $W_j \rightarrow$ mult/ion of A_j^{-1} with B_j, C_j
- We can approximate submatrices of A_j^{-1} by lower rank matrices
- Can we take advantage of it?

An approach based on matrix inversion II

Express A_j as a 2×2 block matrix

$$A_j = \begin{bmatrix} A_j^{(1)} & A_j^{(2)} \\ A_j^{(3)} & A_j^{(4)} \end{bmatrix} \tag{12}$$

where $A_j^{(1)} \in \mathbb{R}^{(n_j - z) \times (n_j - z)}$, $A_j^{(2)} \in \mathbb{R}^{(n_j - z) \times z}$, $A_j^{(3)} \in \mathbb{R}^{z \times (n_j - z)}$, $A_j^{(4)} \in \mathbb{R}^{z \times z}$ and $z \geq m$, then

$$A_j^{-1} = \begin{bmatrix} * & -(A_j^{(1)} - A_j^{(2)}(A_j^{(4)})^{-1}A_j^{(3)})^{-1}A_j^{(2)}(A_j^{(4)})^{-1} \\ * & (A_j^{(4)})^{-1} + (A_j^{(4)})^{-1}A_j^{(3)}(A_j^{(1)} - A_j^{(2)}(A_j^{(4)})^{-1}A_j^{(3)})^{-1}A_j^{(2)}(A_j^{(4)})^{-1} \end{bmatrix}$$

An approach based on matrix inversion III

Spike V_j can be computed as

$$A_j^{(4)} Y_j^{(1)} = \bar{B}_j \tag{13}$$

$$S_{A_j^{(4)}} Y_j^{(2)} = A_j^{(2)} Y_j^{(1)} \tag{14}$$

$$A_j^{(4)} Y_j^{(3)} = A_j^{(3)} Y_j^{(2)} \tag{15}$$

where $S_{A_j^{(4)}} = (A_j^{(1)} - A_j^{(2)}(A_j^{(4)})^{-1} A_j^{(3)})$ is the Schur complement of $A_j^{(4)}$ and by $\bar{B}_j$ we denote the (upwards) extension of B_j by $z - m$ null rows.

An approach based on matrix inversion IV

In the same manner, we can express A_j as a 2×2 block matrix

$$A_j = \begin{bmatrix} A_j^{(1)} & A_j^{(2)} \\ A_j^{(3)} & A_j^{(4)} \end{bmatrix} \tag{16}$$

where $A_j^{(1)} \in \mathbb{R}^{z \times z}$, $A_j^{(2)} \in \mathbb{R}^{z \times (n_j - z)}$, $A_j^{(3)} \in \mathbb{R}^{(n_j - z) \times z}$, $A_j^{(4)} \in \mathbb{R}^{(n_j - z) \times (n_j - z)}$. Then,

$$A_j^{-1} = \begin{bmatrix} (A_j^{(1)})^{-1} + (A_j^{(1)})^{-1} A_j^{(2)} (A_j^{(4)} - A_j^{(3)} (A_j^{(1)})^{-1} A_j^{(2)})^{-1} A_j^{(3)} (A_j^{(1)})^{-1} & * \\ -(A_j^{(4)} - A_j^{(3)} (A_j^{(1)})^{-1} A_j^{(2)})^{-1} A_j^{(3)} (A_j^{(1)})^{-1} & * \end{bmatrix}$$

An approach based on matrix inversion V

Spike W_j can be computed as

$$A_j^{(1)} Y_j^{(1)} = \bar{C}_j \tag{17}$$

$$S_{A_j^{(1)}} Y_j^{(2)} = A_j^{(3)} Y_j^{(1)} \tag{18}$$

$$A_j^{(1)} Y_j^{(3)} = A_j^{(2)} Y_j^{(2)} \tag{19}$$

where $S_{A_j^{(1)}} = (A_j^{(4)} - A_j^{(3)}(A_j^{(1)})^{-1} A_j^{(2)})$ is the Schur complement of $A_j^{(1)}$ and by $\bar{C}_j$ we denote the (downwards) extension of C_j by $z - m$ null rows.

Algorithm 3 The SC-SPIKE algorithm

Input : $A_j \in \mathbb{R}^{n\times n},\ B_j \in \mathbb{R}^{m\times m},\ C_j \in \mathbb{R}^{m\times m},\ z \in \mathbb{Z}$
Output : $V_j \in \mathbb{R}^{n\times m},\ W_j \in \mathbb{R}^{n\times m}$

{Each processor j:}

{**compute** V_j **(if** $j < p$**):**}

$A_j^{(4)} Y_j^{(1)} = \hat{B}_j$

$Y_j^{(2)} = \text{PROJ}\big(A_j^{(1)} - A_j^{(2)}\big(A_j^{(4)}\big)^{-1} A_j^{(3)},\ A_j^{(2)} Y_j^{(1)}\big)$

$A_j^{(4)} Y_j^{(3)} = A_j^{(3)} Y_j^{(2)}$

{**compute** W_j **(if** $j > 1$**):**}

$A_j^{(1)} Y_j^{(1)} = \hat{C}_j$

$Y_j^{(2)} = \text{PROJ}\big(A_j^{(4)} - A_j^{(3)}\big(A_j^{(1)}\big)^{-1} A_j^{(2)},\ A_j^{(3)} Y_j^{(1)}\big)$

$A_j^{(1)} Y_j^{(3)} = A_j^{(2)} Y_j^{(2)}$

Solve $A_j G_j = F_j$ by an iterative method

Solve $SX = G$

Algorithm 4 PROJ

Input: $K \in \mathbb{R}^{n \times n}$, $L \in \mathbb{R}^{n \times s}$
Output: $Y \in \mathbb{R}^{n \times s}$

1. $[Q,\ R] = \mathrm{QR}(L)$
2. $\ell = \{i : l^{(i)}$ belongs in basis of $L\}$
3. Solve $KY_\ell = L_\ell$
4. $D = L_\ell \left(L_\ell^\top L_\ell\right)^{-1} \left(L_\ell^\top L_{\notin \ell}\right)$
5. $Y_{\notin \ell} = Y_\ell D$

Outline

Computational system and software

Computational system

- Experiments were performed in Purdue University
- 2 Inter(R) Xeon(R) processors with a total of 12 cores in 2.8 GHz
- 2 GBs RAM per core

Software

- Fortran90
- Intel(R) MPI Library for Linux, 64-bit, Version 4.0
- MPIIFORT Fortran MPI compiler

Numerical experiments with banded matrices I

About

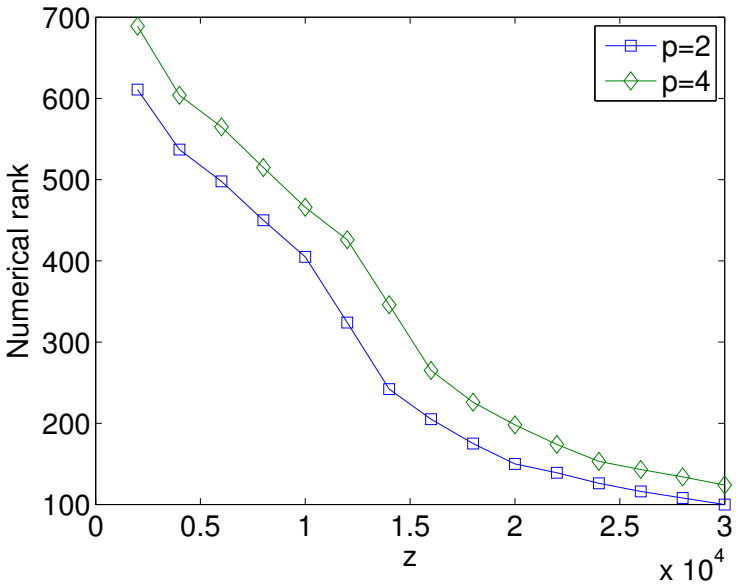
- In this set of experiments we study the SC-SPIKE algorithm
- Matrices were selected from the University of Florida Sparse Matrix Collection (UFSC) (DavHu11)
- We also present experiments with Toeplitz matrices

Numerical experiments with banded matrices II

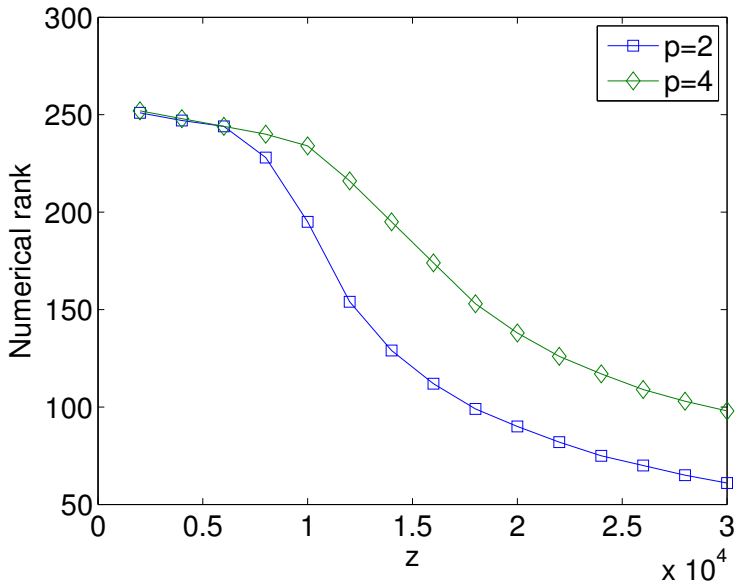
Table : Characteristics of the matrices used (UFSC)

A	n	$\mathrm{nnz}(A)$	semi-bandwidth (m)	Application
1. BenElechi1	245,874	13M	821	2D/3D
2. kim2	456,976	11.5M	1354	2D/3D
3. af_5_k101	503,625	17.5M	859	Structural problem
4. mc2depi	525,825	2.1M	513	2D/3D

Matrices from UFSC I



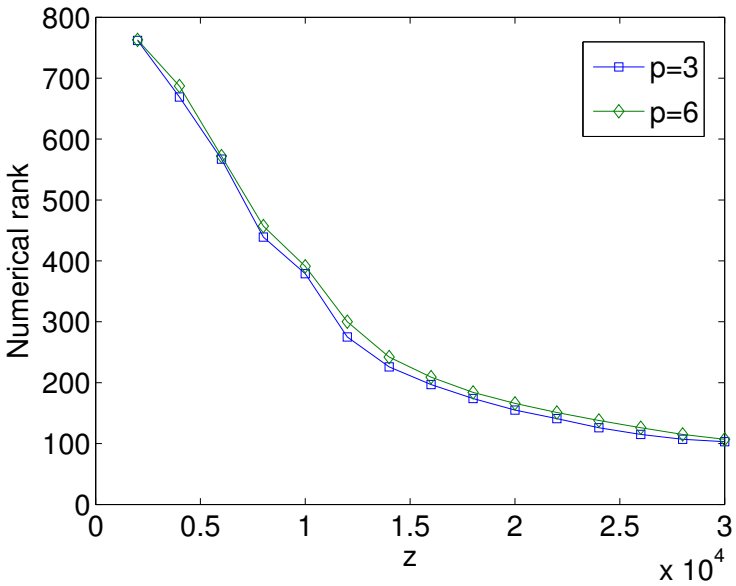
(a) BenElechi1



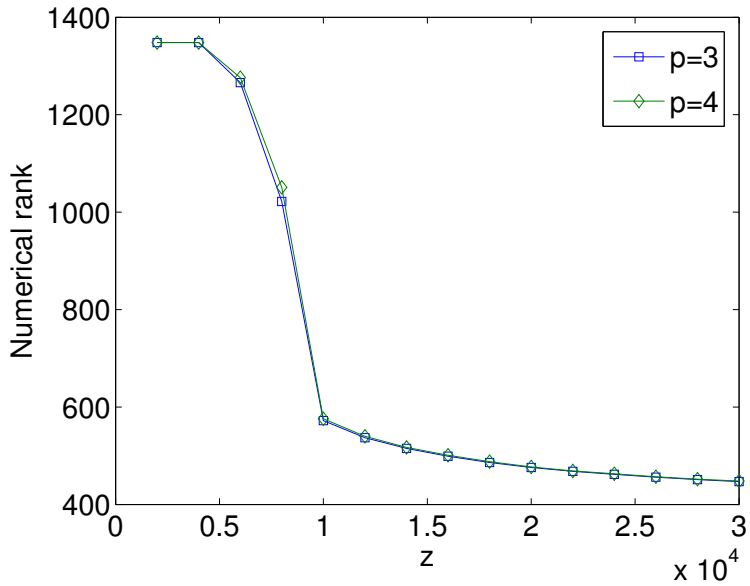
(b) mc2depi

Figure : Rank of submatrix $(V_1)_{1:n_j-z}^{1:m}$ ($p = 2, 4$) for the BenElechi1 and mc2depi matrices.

Matrices from UFSC II



(a) af_5_k101

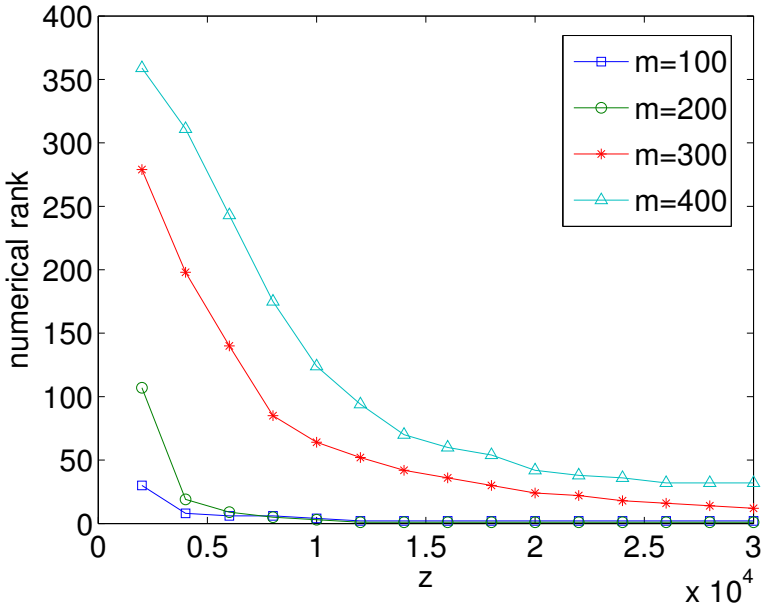


(b) kim2

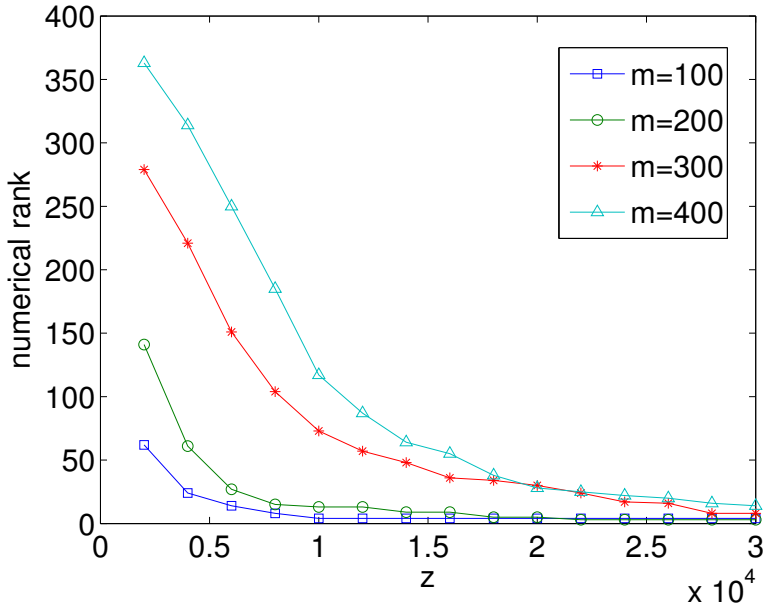
Figure : Rank of submatrix $(V_1)_{1:n_j-z}^{1:m}$ ($p = 2, 4$) for the shipsec5 and kim2 matrices.

Toeplitz I

$a_{ij} = randn(1),\ a_{ii} = \delta \sum_{i=1, i\neq j}^{n} |a_{ij}|$



(a) $\delta = 1/2$

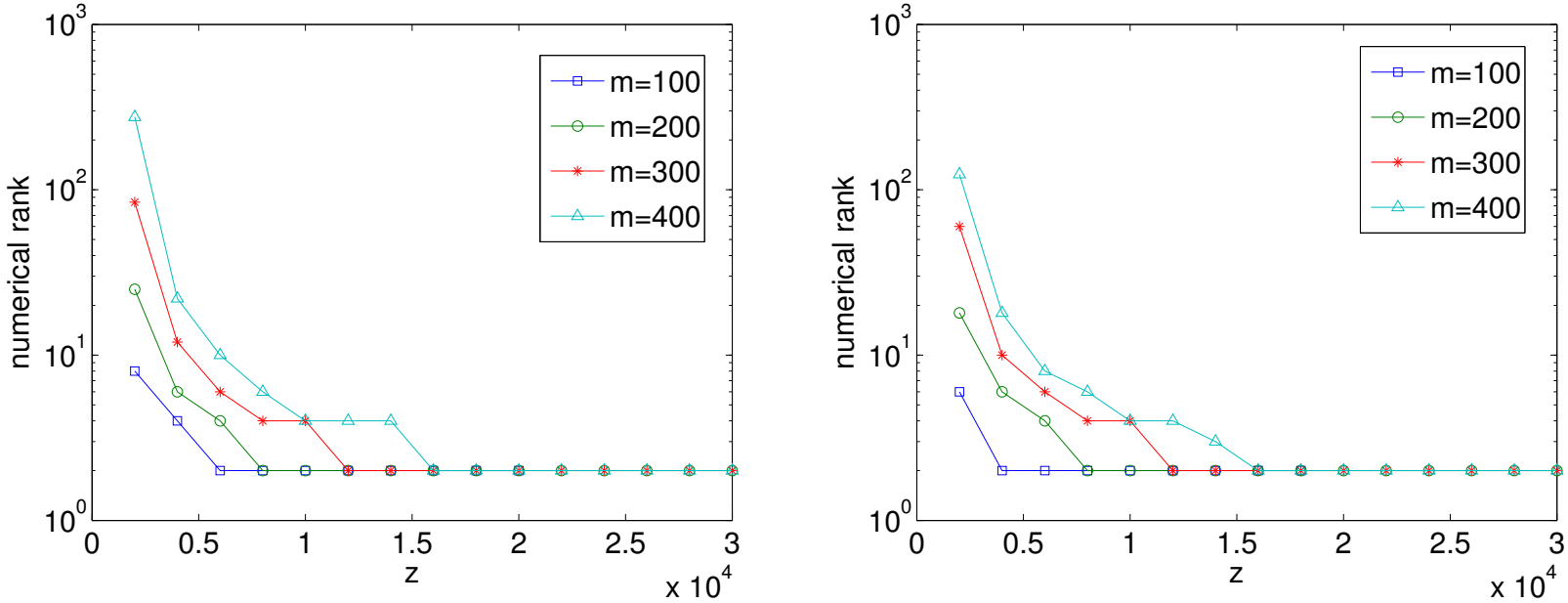


(b) $\delta = 1^{+}$

Figure : Rank of submatrix $(V_1)_{1:n_j-z}^{1:m}$ for a Toeplitz matrix with varying m and degree of diagonal dominance.

Toeplitz II

$a_{ij} = 1/|i-j|$, $a_{ii} = \delta \sum_{i=1, i\neq j}^{n} |a_{ij}|$



(a) $\delta = 1/2$ (b) $\delta = 1^{+}$

Figure : Rank of submatrix $(V_1)_{1:n_j-z}^{1:m}$ for a Toeplitz matrix with varying m and degree of diagonal dominance.

Numerical experiments with banded matrices III

About

- Matrices were selected from UFSC
- $F = rand(n, 10)$

List of algorithms

- Alg1: All multiple right-hand sides are solved by BCG
- Alg2: All multiple right-hand sides are solved by BCG, preconditioned by IC(0)
- Cholesky: All multiple right-hand sides are solved by Cholesky

Numerical experiments with banded matrices IV

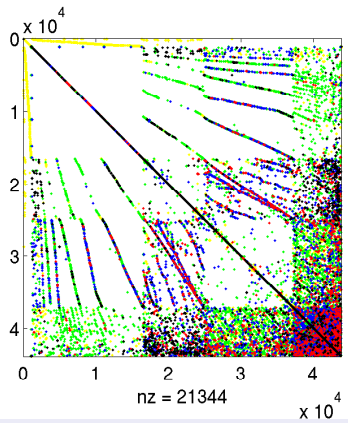
Table : Running times of SPIKE algorithm for varying number of partitions ($s = 10$).

	$p = 2$			$p = 4$			$p = 8$		
	Cholesky	Alg1	Alg2	Cholesky	Alg1	Alg2	Cholesky	Alg1	Alg2
bcsstk16	**0.9**	5.1	2.4	**0.4**	2.5	0.9	**0.2**	0.7	0.3
fv1	2.8	0.5	**0.4**	1.3	0.3	**0.2**	0.5	0.2	**0.1**
t2dah_e	2.7	7.5	**2.1**	**1.0**	3.1	1.2	**0.5**	1.9	0.6
crystm02	54.7	7.7	**0.9**	23.8	5.8	**0.5**	9.6	3.1	**0.2**
wathen100	12.7	16.7	**5.3**	6.5	7.3	**2.4**	2.9	3.7	**1.3**
wathen120	15.7	21.7	**8.2**	6.5	6.9	**2.9**	3.4	5.3	**1.7**
jnbrng1	12.8	11.0	**6.2**	5.8	7.9	**4.4**	4.2	4.6	**2.2**
apache1	**204.2**	F	644.9	**117.1**	F	278.5	**53.6**	F	105.2

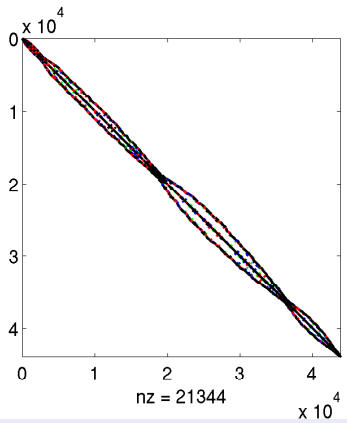
The Spike algorithm as a preconditioner

- The Spike algorithm can be used as a banded preconditioner M
- We need $||M||_F \approx ||A||_F$
- We need pre-processing of A...
- Reordering?
- RCM, Fiedler?
- Weighted Fiedler (WSO) (Mang10)?

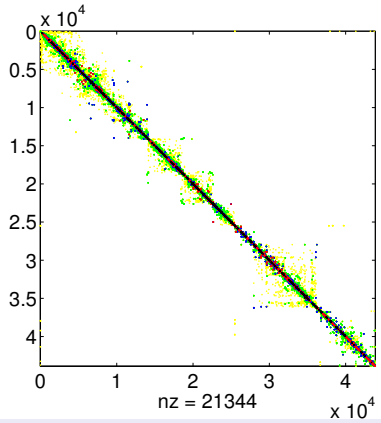
Reordering of sparse matrices I



(a) No reordering



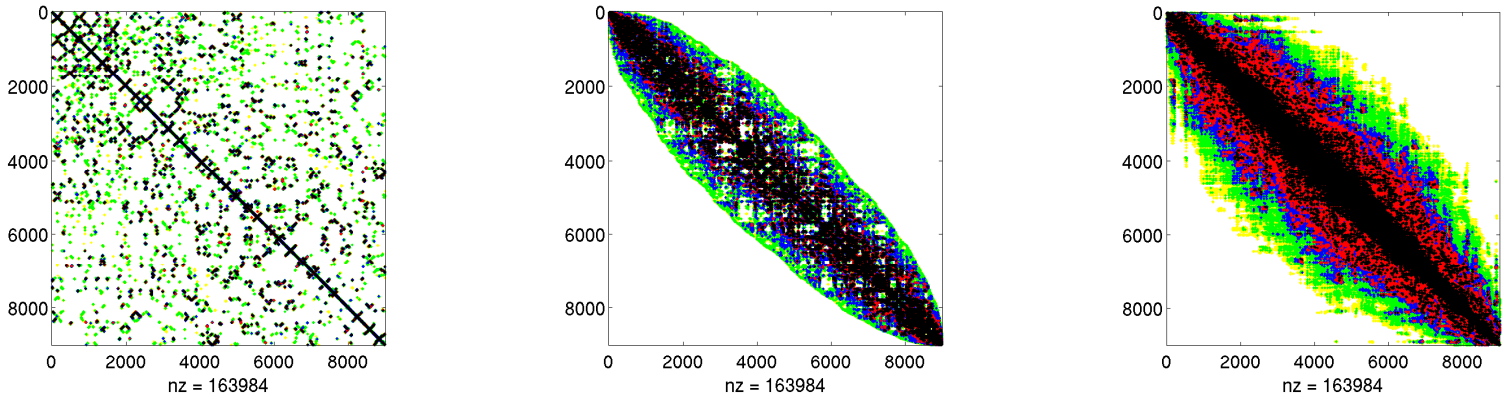
(b) RCM



(c) WSO

Figure : Non-zero pattern before and after RCM and WSO permutations for matrix opf10000

Reordering of sparse matrices II



(a) No reordering (b) RCM (c) WSO

Figure : Non-zero pattern before and after RCM and WSO permutations for matrix nd3k

The PSpike algorithm

Algorithmic sketch

1. Reordering of the coefficient matrix A
2. Extraction of a banded preconditioner M with semi-bandwidth $\hat{m}$
3. Solve
$$M^{-1}(PAP^{\top})\tilde{X} = M^{-1}F$$
by a Krylov subspacec method. Preconditioner is applied through Spike.
4. Form final solution,
$$X = P^{\top}\tilde{X}.$$

Numerical experiments with general sparse matrices

About

- Matrices were selected from UFSC
- Different reordering methods
- Extraction of a banded preconditioner M with semi-bandwidth $\hat{m}_1$ $(\hat{m}_2)$ such that $||M||_F \geq 0.95 * ||A||_F$ $(||M||_F \geq 0.99 * ||A||_F)$

Reordering methods

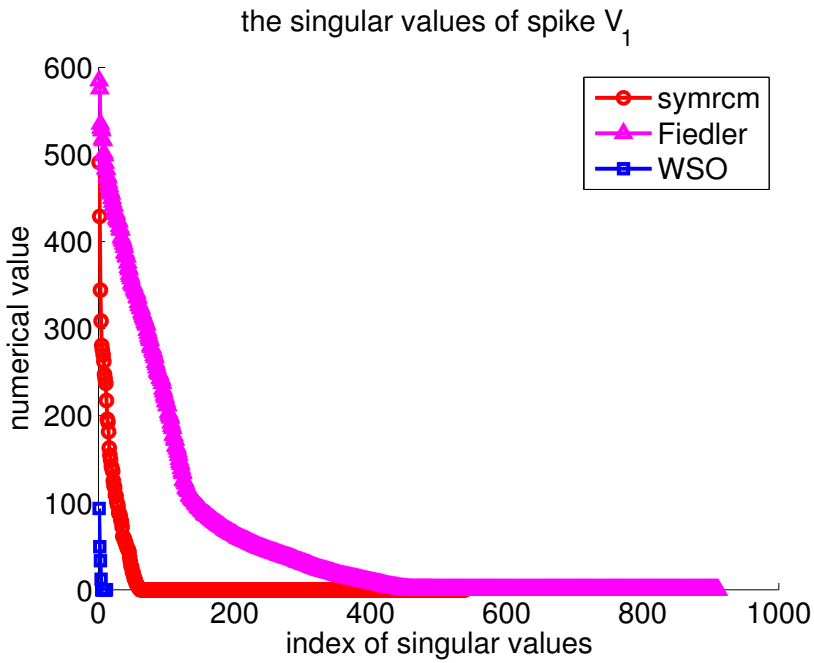
- RCM
- Fiedler
- WSO

Table : Characteristics of the matrices selected from UFSC

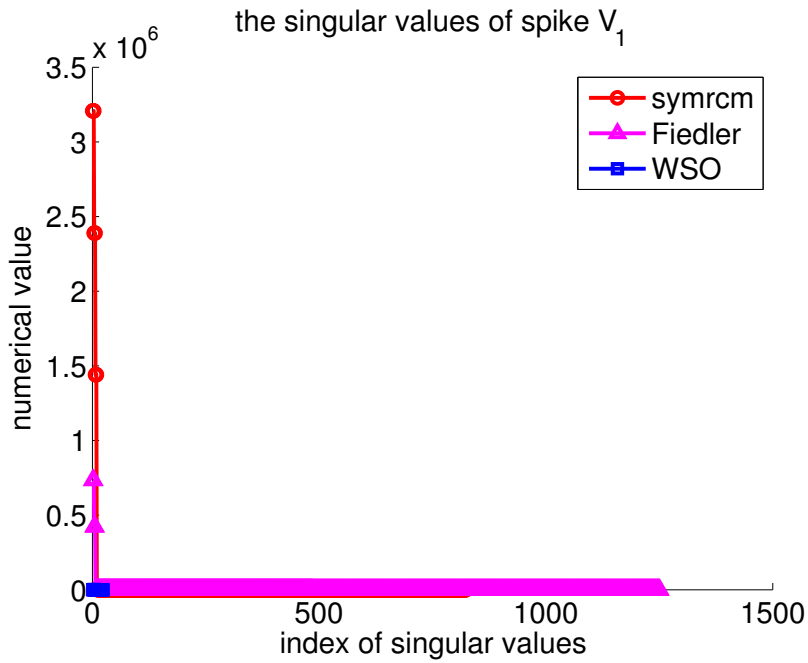
	n	Application
1. bcsstk28	4410	Structural problem
2. eurqsa	7245	Structural problem
3. OPF_3754	15735	Power electricity
4. dubcova1	16129	2D/3D
5. raefsky4	19779	Structural problem

Table : Original semi-bandwidth m and semi-bandwidths $\hat{m}_1$, $\hat{m}_2$ so that $||M||_F \geq 0.95 * ||A||_F$ and $||M||_F \geq 0.99 * ||A||_F$ after symmetric reorderings

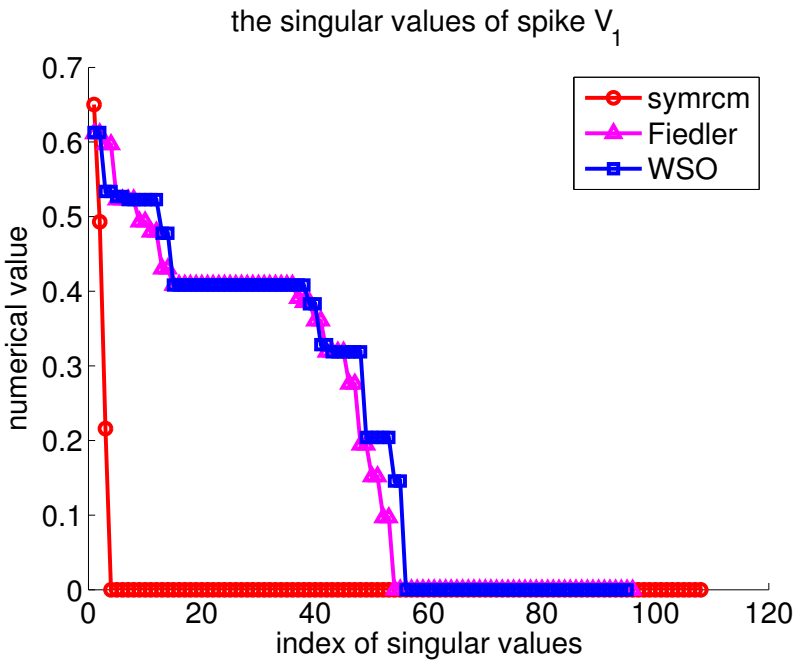
	RCM			Fiedler			WSO		
	m	$\hat{m}_1$	$\hat{m}_2$	m	$\hat{m}_1$	$\hat{m}_2$	m	$\hat{m}_1$	$\hat{m}_2$
bcsstk28	406	271	274	815	224	344	993	10	17
eurqsa	438	357	383	1568	335	978	1733	77	321
OPF_3754	800	550	729	3240	74	393	5289	7	28
dubcova1	500	108	446	653	97	245	691	95	236
raefsky4	1052	539	823	5918	911	1251	5105	12	24



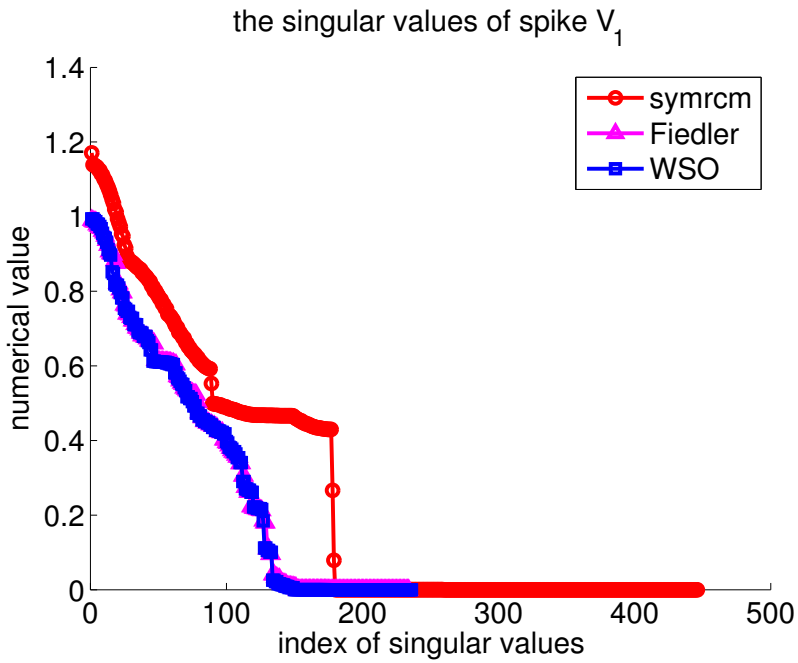
(a) raefsky4 ($\hat{m}_1$)



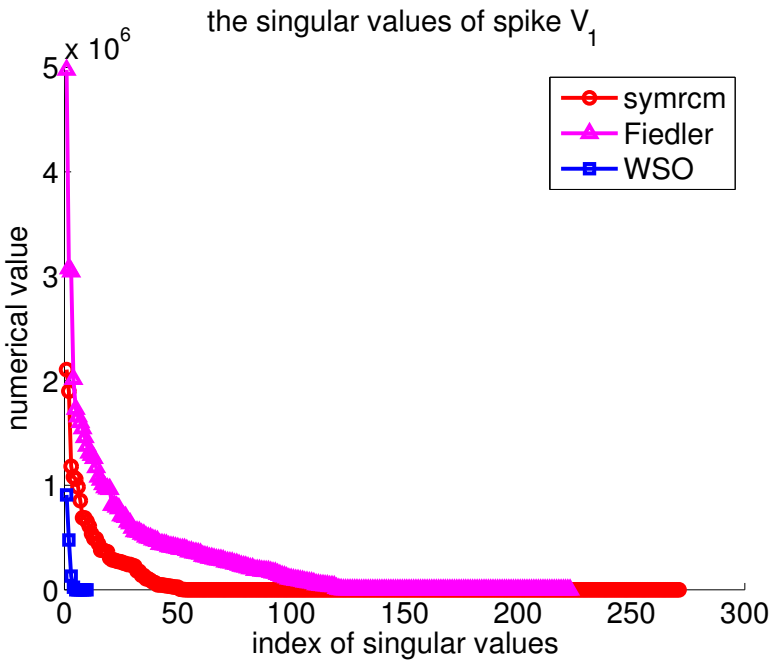
(b) raefsky4 ($\hat{m}_2$)



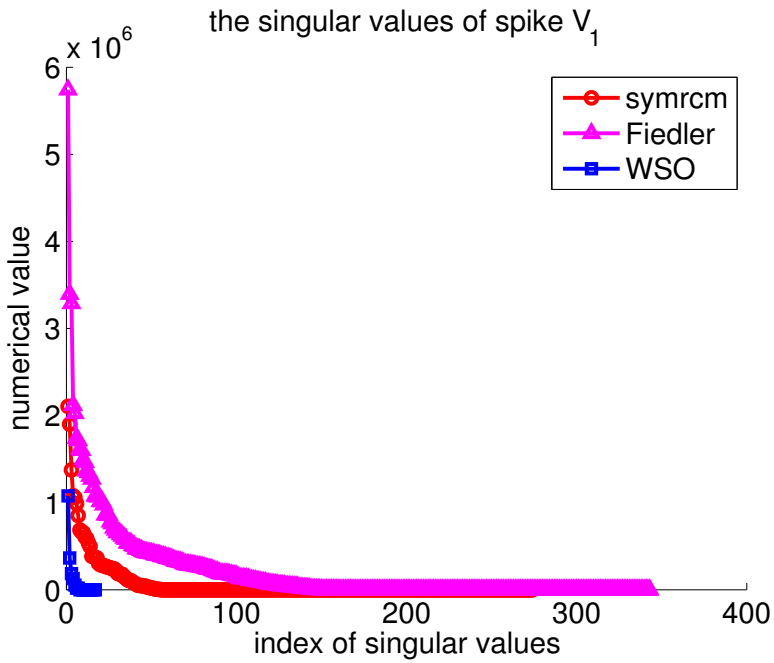
(c) dubcova1 ($\hat{m}_1$)



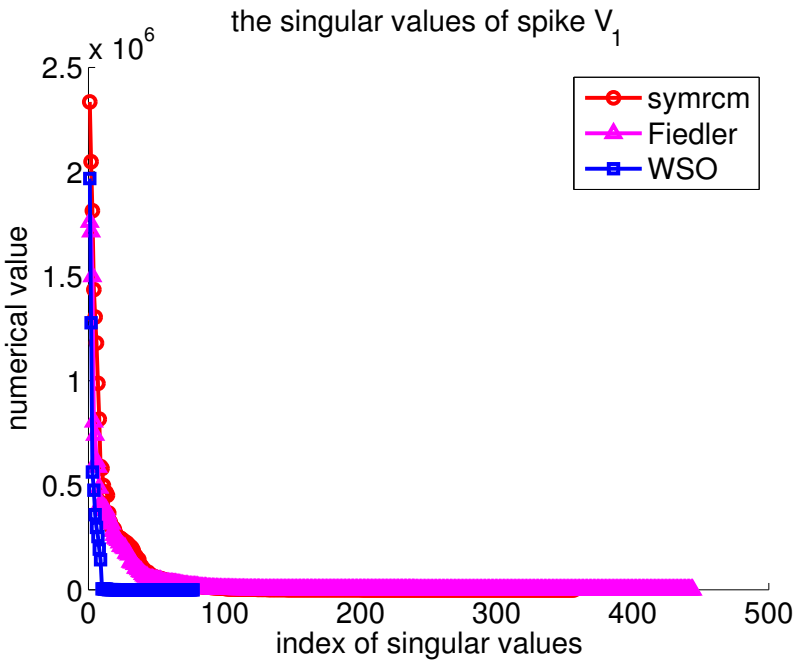
(d) dubcova1 ($\hat{m}_2$)



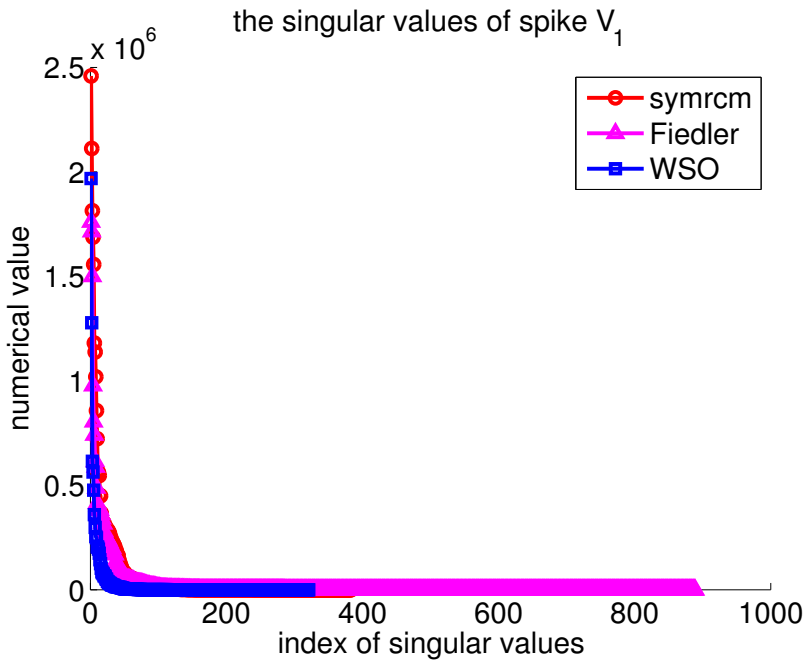
(e) bcsstk28 ($\hat{m}_1$)



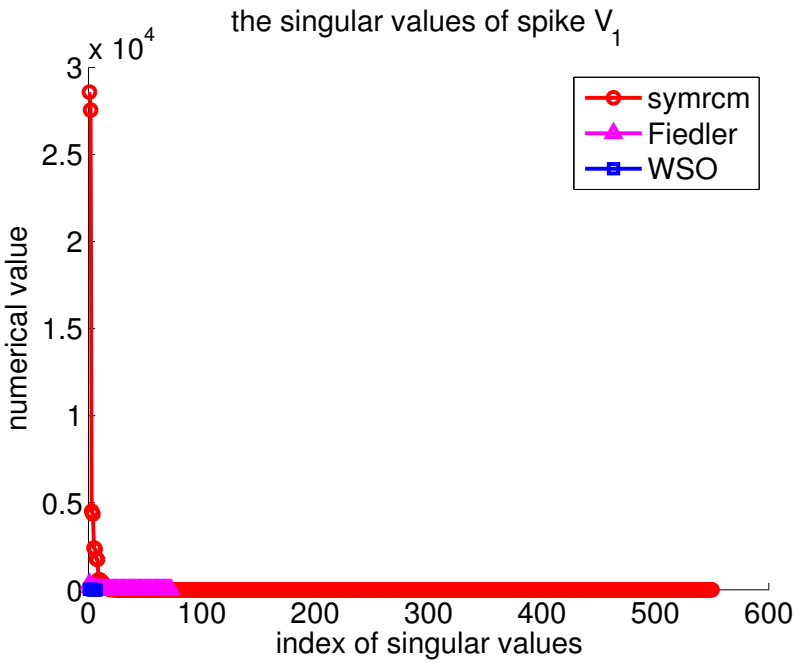
(f) bcsstk28 ($\hat{m}_2$)



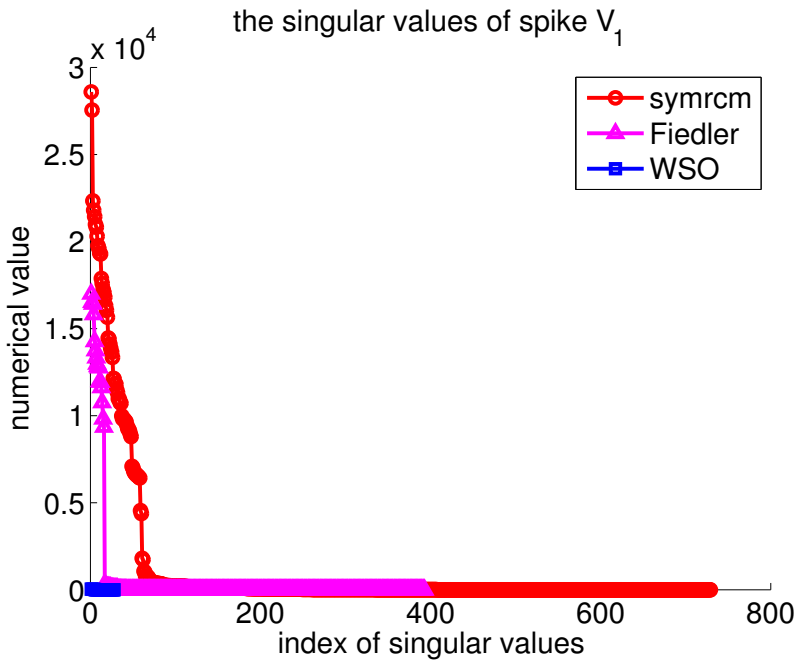
(g) eurqsa ($\hat{m}_1$)



(h) eurqsa ($\hat{m}_2$)



(i) OPF_3574 ($\hat{m}_1$)



(j) OPF_3574 ($\hat{m}_2$)

Talks and manuscripts

Talks

- NUMAN 2012, September 2012, Ioannina, Greece

Manuscripts

- VK, F. Saied, E. Gallopoulos and A. Sameh. *Accelerating Spike family of algorithms by solving linear systems with multiple right-hand sides*, Technical Report

Acknowledgements

PurdueU A. Klinvex, A. Sameh, F. Saied, Y. Zhu, G. Kollias

UPatras E. Gallopoulos, E. Kontopoulou, C. Zaroliagis, A. Boudouvis

Conclusion

In this thesis

- We studied the combination of Spike algorithms with techniques for solving linear systems with multiple right-hand sides
- We saw that under certain conditions, replacing direct solvers by these techniques can lead to more efficient schemes
- In future we seek to develop schemes for accelerating the Spike algorithm when not all multiple right-hand sides are available at once

Outline

Bibliography I


T. Chan and L. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*. SIAM J. Sci. Comput., **18**(6), pp. 1698-1721, 1997.


T. A. Davis and Y. Hu, *The university of Florida sparse matrix collection*. ACM Trans. Math. Soft., **38**(1), pp. 1-25, 2011.


D. P. O'Leary, *The block conjugate gradient algorithms and related methods*. Linear Algebra and its Applications, **29**, pp. 293-322, 1980.


F. Zhang, *Schur complement and its applications*. Springer, 2005.


E. Polizzi and A. Sameh, *A parallel hybrid banded system solver: the SPIKE algorithm*. Parallel Comput., **32**(2), pp. 177-194, 2006, Elsevier Science Publishers B. V.


M. Manguoglu, M. Koyutürk, A. Sameh and A. Grama, *Weighted Matrix Ordering and Parallel Banded Preconditioners for Iterative Linear System Solvers*. SIAM J. Scientific Computing, **32**(3), pp. 1201-1216, 2010.


M. Gutknecht and T. Schmelzer, *The block grade of a Krylov subspace*. Linear Alg. and its Appl., **430**(11), pp. 174-185, 2009.

Bibliography II


C. F. Smith, A. F. Peterson and R. Mittra, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*. IEEE Trans. Antennas and Propagation, **37**, pp. 1490-1493, 1989.


V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides* SIAM J. Sci. Comput, **16**, pp. 917-933, 1995.


Y. Saad, *On the Lanczos method for solving symmetric linear systems with multiple right-hand sides*. Math. Comp., **48**, pp. 651-662, 1987.


E. Gallopoulos, S. Hatzimihail and V. Kalantzis, *Towards Lightweight Projection Methods for Linear Systems with Multiple Right-Hand Sides*. Midwest Numerical Analysis Days 2011, presentation.


V. Simoncini and E. Gallopoulos, *Convergence properties of block GMRES and matrix polynomials*. Linear Alg. and its Appl., **247**, pp. 97-119, 1996.


V. Simoncini and E. Gallopoulos, *A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides*. J. of Comput. and Applied Mathematics, **66**(1), pp. 457-469, 1996.