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

1. Introduction
2. Parallel solution of banded linear systems - the Spike method
3. Solvers for linear systems with multiple right-hand sides
4. Numerical experiments
5. Bibliography
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 \( \Rightarrow \text{nnz}(A) = O(n) \)
- Banded with semi-bandwidth \( m \) (if \( m \ll n \Rightarrow \text{nnz}(A) = O(n) \)).
- The banded coefficient matrix can be:

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

**Figure**: Dense within the band

**Figure**: Sparse within the band
Outline

1. Introduction

2. Parallel solution of banded linear systems - the Spike method

3. Solvers for linear systems with multiple right-hand sides

4. Numerical experiments

5. Bibliography
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
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 → Block tridiagonal matrix

- It can be divided in two phases
  - Pre-processing
  - Post-processing
An example for \( p = 4 \) partitions

\[
\begin{bmatrix}
A_1 & B_1 \\
C_2 & A_2 & B_2 \\
C_3 & A_3 & B_3 \\
C_4 & A_4
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix} =
\begin{bmatrix}
F_1 \\
F_2 \\
F_3 \\
F_4
\end{bmatrix}
\]

\( A_j \in \mathbb{R}^{n_j \times n_j}, \ F_j \in \mathbb{R}^{n_j \times s}, \ B_j, \ C_j \in \mathbb{R}^{m \times m} \)
The Spike algorithm II

**Matrix A is factorized as**

\[
A = DS
\]  

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

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**

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

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**

\[ A_j G_j = F_j, \quad j = 1, \ldots, p. \]  

and

\[ SX = G. \]

- Solution of (3) is performed *locally* in each processor.
- Solution of (4) demands communication between neighboring processors.
Solution of $SX = GI$

Partitioning of $V_j, W_j, X_j, G_j$

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

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 $p = 3$

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

Intermediate blocks of rows are computed locally as

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

1. Introduction
2. Parallel solution of banded linear systems - the Spike method
3. Solvers for linear systems with multiple right-hand sides
4. Numerical experiments
5. Bibliography
Local linear systems in Spike algorithm

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

\[ A_j[V_j, W_j, G_j] = [\hat{B}_j, \hat{C}_j, F_j]. \]  \hspace{1cm} (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**
  \[ P_j A_j = L_j U_j. \]  
  - CC: \( O(n_j^3) \)
  - Sparse structure? Pardiso, MUMPS, SuperLu...
  - Banded structure? CC: \( O(n_j m^2) \)

- **By Cholesky decomposition**
  \[ A_j = L_j L_j^T. \]  

Iterative methods

- **Stationary point methods**
- **Krylov subspace methods**
Krylov subspace iterative methods

Solution of $A_jx = b$

- Solution is sought in the Krylov subspace,
  \[ \mathcal{K}_\mu(A_j, b) = \text{span}\{b, A_jb, \ldots, A_j^{\mu-1}b\}. \]

- If $q(A_j) = \alpha_0 I_n + \alpha_1 A_j + \ldots + \alpha_n A_j^n = 0$, then
  \[ A_j^{-1}b = \frac{-1}{\alpha_0} \sum_{i=0}^{n-1} \alpha_{i+1} A_j^i. \]  
  (9)

- In the $i$-th iteration
  \[ x_i \in x_0 + \mathcal{K}_i, \quad r_i = (b - A_jx_i) \perp C_i. \]

Different choices for subspace $C_i$: $C_i \equiv \mathcal{K}_i$ or $C_i \equiv A_j\mathcal{K}_i$. 
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?
Goal: Reduce the solution cost of $J$ linear systems

How: Multiple Right Hand Sides Solvers

Ideally $(J - K) \rightarrow J$
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 $K_i(A, r_0^{(1)})$. Then

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

(10)

with

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

(11)

- Procedure continues till all mrhs are solved
- The $j$-th right-hand side is projected on the subspaces Krylov $K_i(A, r_0^{(1,1)}), \ldots, K_{i-1}(A, r_0^{(j-2,j-1)})$
Algorithm 1 Generic algorithm of seed methods (seed) (SPM89,Saad87,SimG95)

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

\begin{align*}
  \text{for } j = 1, \ldots, s \text{ do} \\
  \quad \text{for } i = 1, \ldots \text{ do} \\
  \quad \quad \text{the } j\text{-th linear system performs the } i\text{-th iteration} \\
  \quad \quad \text{for } g = j + 1, \ldots, s \text{ do} \\
  \quad \quad \quad \text{update solution of } x_i^{(j,g)} \\
  \quad \quad \text{end for} \\
  \quad \text{end for} \\
  \text{end for}
\end{align*}
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 $\|r^{(j)}_i\| / \|f^{(j)}\| \geq tol$ do

\[ \alpha = \left( (r^{(j)}_i)^\top r^{(j)}_i \right) / \left( (p^{(j)}_i)^\top Ap^{(j)}_i \right) \]
\[ x^{(j)}_{i+1} = x^{(j)}_i + p^{(j)}_i \alpha; \quad r^{(j)}_{i+1} = r^{(j)}_i - Ap^{(j)}_i \alpha \]
\[ \beta = \left( (r^{(j)}_{i+1})^\top r^{(j)}_{i+1} \right) / \left( (r^{(j)}_i)^\top r^{(j)}_i \right) \]
\[ p^{(j)}_{i+1} = r^{(j)}_{i+1} + p^{(j)}_i \beta \]

for $k = j + 1, \ldots, s$ do

\[ \eta^{(j,k)}_i = \left( p^{(j)}_i \right)^\top r^{(j,k)}_i / \left( p^{(j)}_i \right)^\top Ap^{(j)}_i \]
\[ x^{(j,k)}_{i+1} = x^{(j,k)}_i + \eta^{(j,k)}_i p^{(j)}_i; \quad r^{(j,k)}_{i+1} = r^{(j,k)}_i - \eta^{(j,k)}_i Ap^{(j)}_i \]

end for

end while

end for
Theorem (ChanWan97)

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

$$\|r_{0^{(j,k)}}\| \leq \alpha \sum_{h=1}^{j} |\beta_{m_p+1}|, \quad k = j + 1, \ldots, s,$$

where value $\beta_{m_p+1}$ 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

\[ n_x = n_y = n_z = 15 \]

\[ AX = F, \ s = 50 \]

![Norm of relative residual for different blocksizes](image)

**Figure**: Norm of relative residual for different blocksizes

Accelerating the spike family of algorithms by solving linear systems with multiple right-hand sides
An approach based on matrix inversion I

- For an SPD matrix $H$ we have
  \[
  \left| \left( H^{-1} \right)_{ij} \right| \leq C \lambda^{|i-j|}, \quad i, j = 1, \ldots, n
  \]
  where
  \[
  \lambda = \left( \frac{\sqrt{\kappa(H)} - 1}{\sqrt{\kappa(H)} + 1} \right)^{2/m}.
  \]

- Spikes $V_j, W_j \rightarrow \text{mult/ion of } A_j^{-1}$ with $B_j, C_j$

- We can approximate submatrices of $A_j^{-1}$ by lower rank matrices

- Can we take advantage of it?
Express $A_j$ as a $2 \times 2$ block matrix

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

(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

$$A_j^{-1} = \begin{bmatrix} \ast & -(A_j^{(1)} - A_j^{(2)}(A_j^{(4)})^{-1}A_j^{(3)})^{-1}A_j^{(2)}(A_j^{(4)})^{-1} \\ \ast & (A_j^{(4)})^{-1} + (A_j^{(4)})^{-1}A_j^{(3)}(A_j^{(1)} - A_j^{(2)}(A_j^{(4)})^{-1}A_j^{(3)})^{-1}A_j^{(2)}(A_j^{(4)})^{-1} \end{bmatrix}$$
An approach based on matrix inversion III

Spike $V_j$ can be computed as

\begin{align*}
A_j^{(4)} Y_j^{(1)} &= \tilde{B}_j \quad (13) \\
S_{A_j^{(4)}} Y_j^{(2)} &= A_j^{(2)} Y_j^{(1)} \quad (14) \\
A_j^{(4)} Y_j^{(3)} &= A_j^{(3)} Y_j^{(2)} \quad (15)
\end{align*}

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 $\tilde{B}_j$ we denote the (upwards) extension of $B_j$ by $z - m$ null rows.
In the same manner, we can express $A_j$ as a $2 \times 2$ block matrix

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

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,

$$A_j^{-1} = \begin{bmatrix} (A_j^{(1)})^{-1} + (A_j^{(1)})^{-1} A_j^{(2)} (A_j^{(4)} - A_j^{(3)} (A_j^{(1)})^{-1} A_j^{(2)})^{-1} A_j^{(3)} (A_j^{(1)})^{-1} & * \\ - (A_j^{(4)} - A_j^{(3)} (A_j^{(1)})^{-1} A_j^{(2)})^{-1} A_j^{(3)} (A_j^{(1)})^{-1} & * \end{bmatrix}$$
Spike $W_j$ can be computed as

\begin{align*}
    A_j^{(1)} Y_j^{(1)} &= \bar{C}_j \\ 
    S_{A_j^{(1)}} Y_j^{(2)} &= A_j^{(3)} Y_j^{(1)} \\ 
    A_j^{(1)} Y_j^{(3)} &= A_j^{(2)} Y_j^{(2)}
\end{align*}

(17)\hspace{1cm} (18)\hspace{1cm} (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 $j < p$):}

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

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

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

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

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

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

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

Solve $A_j G_j = F_j$ by an iterative method

Solve $S X = G$
**Algorithm 4** \texttt{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. $[Q, R] = QR(L)$
2. $\ell = \{ i : l^{(i)} \text{ belongs in basis of } L \}$
3. Solve $KY_\ell = L_\ell$
4. $D = L_\ell (L_\ell^T L_\ell)^{-1} (L_\ell^T L_{\ell^c})$
5. $Y_{\ell^c} = Y_\ell D$
Outline

1 Introduction

2 Parallel solution of banded linear systems - the Spike method

3 Solvers for linear systems with multiple right-hand sides

4 Numerical experiments

5 Bibliography
### 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)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>n</th>
<th>nnz(A)</th>
<th>semi-bandwidth (m)</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BenElechi1</td>
<td>245,874</td>
<td>13M</td>
<td>821</td>
<td>2D/3D</td>
</tr>
<tr>
<td>2</td>
<td>kim2</td>
<td>456,976</td>
<td>11.5M</td>
<td>1354</td>
<td>2D/3D</td>
</tr>
<tr>
<td>3</td>
<td>af_5_k101</td>
<td>503,625</td>
<td>17.5M</td>
<td>859</td>
<td>Structural problem</td>
</tr>
<tr>
<td>4</td>
<td>mc2depi</td>
<td>525,825</td>
<td>2.1M</td>
<td>513</td>
<td>2D/3D</td>
</tr>
</tbody>
</table>
Matrices from UFSC I

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

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

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

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

\[ a_{ij} = \frac{1}{|i - j|}, \quad a_{ii} = \delta \sum_{i=1, i \neq j}^{n} |a_{ij}| \]

\[ \delta = \frac{1}{2} \]

\[ \delta = 1^+ \]

Figure: Rank of submatrix \((V_1)_{1:m}^{1:n} - z\) for a Toeplitz matrix with varying \(m\) and degree of diagonal dominance.
Numerical experiments with banded matrices III

About

- Matrices were selected from UFSC
- \( F = \text{rand}(n, 10) \)

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 ($s = 10$).

<table>
<thead>
<tr>
<th></th>
<th>$p = 2$</th>
<th></th>
<th>$p = 4$</th>
<th></th>
<th>$p = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cholesky</td>
<td>Alg1</td>
<td>Alg2</td>
<td>Cholesky</td>
<td>Alg1</td>
</tr>
<tr>
<td>bcsstk16</td>
<td>0.9</td>
<td>5.1</td>
<td>2.4</td>
<td>0.4</td>
<td>2.5</td>
</tr>
<tr>
<td>fv1</td>
<td>2.8</td>
<td>0.5</td>
<td>0.4</td>
<td>1.3</td>
<td>0.3</td>
</tr>
<tr>
<td>t2dah_e</td>
<td>2.7</td>
<td>7.5</td>
<td>2.1</td>
<td>1.0</td>
<td>3.1</td>
</tr>
<tr>
<td>crystm02</td>
<td>54.7</td>
<td>7.7</td>
<td>0.9</td>
<td>23.8</td>
<td>5.8</td>
</tr>
<tr>
<td>wathen100</td>
<td>12.7</td>
<td>16.7</td>
<td>5.3</td>
<td>6.5</td>
<td>7.3</td>
</tr>
<tr>
<td>wathen120</td>
<td>15.7</td>
<td>21.7</td>
<td>8.2</td>
<td>6.5</td>
<td>6.9</td>
</tr>
<tr>
<td>jnbrng1</td>
<td>12.8</td>
<td>11.0</td>
<td>6.2</td>
<td>5.8</td>
<td>7.9</td>
</tr>
<tr>
<td>apache1</td>
<td>204.2</td>
<td>F</td>
<td>644.9</td>
<td>117.1</td>
<td>F</td>
</tr>
</tbody>
</table>
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

Figure: Non-zero pattern before and after RCM and WSO permutations for matrix opf10000
Reordering of sparse matrices II

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

Algorithmic sketch

1. Reordering of the coefficient matrix $A$
2. Extraction of a banded preconditioner $M$ with semi-bandwidth $\hat{m}$
3. Solve
   \[ M^{-1}(PAP^T)X = M^{-1}F \]
   by a Krylov subspace method. Preconditioner is applied through Spike.
4. Form final solution,
   \[ X = P^T \hat{X}. \]
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 $\|M\|_F \geq 0.95 \times \|A\|_F$ ($\|M\|_F \geq 0.99 \times \|A\|_F$)

Reordering methods

- RCM
- Fiedler
- WSO
Table: Characteristics of the matrices selected from UFSC

<table>
<thead>
<tr>
<th></th>
<th>Application</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>bcsstk28</td>
<td>4410</td>
</tr>
<tr>
<td>2.</td>
<td>eurqsa</td>
<td>7245</td>
</tr>
<tr>
<td>3.</td>
<td>OPF_3754</td>
<td>15735</td>
</tr>
<tr>
<td>4.</td>
<td>dubcova1</td>
<td>16129</td>
</tr>
<tr>
<td>5.</td>
<td>raefsky4</td>
<td>19779</td>
</tr>
</tbody>
</table>

Structure problem
Power electricity
2D/3D
Structural problem
Table: Original semi-bandwidth $m$ and semi-bandwidths $\hat{m}_1$, $\hat{m}_2$ so that $\|M\|_F \geq 0.95 \ast \|A\|_F$ and $\|M\|_F \geq 0.99 \ast \|A\|_F$ after symmetric reorderings.

<table>
<thead>
<tr>
<th></th>
<th>RCM</th>
<th></th>
<th></th>
<th>Fiedler</th>
<th></th>
<th></th>
<th>WSO</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m$</td>
<td>$\hat{m}_1$</td>
<td>$\hat{m}_2$</td>
<td>$m$</td>
<td>$\hat{m}_1$</td>
<td>$\hat{m}_2$</td>
<td>$m$</td>
<td>$\hat{m}_1$</td>
<td>$\hat{m}_2$</td>
</tr>
<tr>
<td>bcsstk28</td>
<td>406</td>
<td>271</td>
<td>274</td>
<td>815</td>
<td>224</td>
<td>344</td>
<td>993</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>eurqsa</td>
<td>438</td>
<td>357</td>
<td>383</td>
<td>1568</td>
<td>335</td>
<td>978</td>
<td>1733</td>
<td>77</td>
<td>321</td>
</tr>
<tr>
<td>OPF_3754</td>
<td>800</td>
<td>550</td>
<td>729</td>
<td>3240</td>
<td>74</td>
<td>393</td>
<td>5289</td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>dubcova1</td>
<td>500</td>
<td>108</td>
<td>446</td>
<td>653</td>
<td>97</td>
<td>245</td>
<td>691</td>
<td>95</td>
<td>236</td>
</tr>
<tr>
<td>raefsky4</td>
<td>1052</td>
<td>539</td>
<td>823</td>
<td>5918</td>
<td>911</td>
<td>1251</td>
<td>5105</td>
<td>12</td>
<td>24</td>
</tr>
</tbody>
</table>
Accelerating the spike family of algorithms by solving linear systems with multiple right-hand sides

(a) raeffsky4 ($\hat{m}_1$) 

(b) raeffsky4 ($\hat{m}_2$)
Accelerating the spike family of algorithms by solving linear systems with multiple right-hand sides
Accelerating the spike family of algorithms by solving linear systems with multiple right-hand sides
Acceleration of the spike family of algorithms by solving linear systems with multiple right-hand sides.

\[ \text{the singular values of spike } V_1 \]

\( \text{(g) eurqa (} \hat{m}_1) \)

\( \text{(h) eurqa (} \hat{m}_2) \)
Accelerating the spike family of algorithms by solving linear systems with multiple right-hand sides

The singular values of spike $V_1$

(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

1 Introduction

2 Parallel solution of banded linear systems - the Spike method

3 Solvers for linear systems with multiple right-hand sides

4 Numerical experiments

5 Bibliography


