Weighted Bandwidth Reduction and Preconditioning Sparse Systems

Ananth Grama
Department of Computer Science, Purdue University

July 9, 2007

Acknowledgment: Funding for this work was provided by XXX.

Bandwidth Reduction

- Traditional algorithms (*e.g.*, Cuthill-McKee (Cuthill & McKee, 1969), Spectral reordering (Barnard *et al.*, 1995)) are aimed at minimizing the bandwidth

$$BW(A) = \max_{i,j:A(i,j)>0} |i - j|$$

- Heavy (high-magnitude) nonzeros that are distant from the diagonal may significantly degrade the performance (*e.g.* convergence rate)
  - Particularly, for matrices whose entries vary significantly in magnitude

Accounting for Heavy Entries

- We relax the definition of bandwith
- For given b, we define bandweight as

$$w_b(A) = \sum_{i,j:|i-j|<b} |A(i,j)|$$

- Then, for given α, we define α-bandwidth as the smallest bandwidth that encapsulates an α fraction of total matrix weight

$$BW_\alpha(A) = \min b \text{ such that } w_b(A) \geq \alpha \times \sum_{i,j} |A(i,j)|$$

  - Observe that this is a generalization of bandwidth, such that $BW_1(A) = BW(A)$

Spectral Ordering

- Commonly used in graph-theoretic applications and matrix algorithms

- Find x that minimizes

$$\sum_{i,j:A(i,j)>0} (x(i) - x(j))^2$$

  - Reorder rows and columns of A accordingly

- The eigenvector that corresponds to the smallest non-zero eigenvalue of the Laplacian

$$\begin{aligned} L(i,j) &= -1 && \textbf{if } i \neq j \wedge A(i,j) > 0 \\ L(i,i) &= |\{j : A(i,j) > 0\}|, \end{aligned}$$

a.k.a, Fiedler vector, minimizes this cost function (Fiedler, 1973)

  - Can be computed effectively using iterative techniques (*e.g.*, CG (Kruyt, 1995))

Weighted Spectral Ordering

- Fiedler's result generalizes to weighted graphs (matrices) as well

- Define Weighted Laplacian as

$$\begin{array}{ll} \bar{L}(i,j) = -|A(i,j)| & \quad \textbf{if } i \neq j \\ \bar{L}(i,i) = \sum_j |A(i,j)| & \end{array}$$

- The eigenvector that corresponds to the smallest non-zero eigenvalue of the weighted Laplacian minimizes

$$x^T \bar{L} x = \sum_{i,j} |A(i,j)|(x(i) - x(j))^2,$$

- Observe that $x^T L x$ is closely related to $\sum_{i,j} |A(i,j)| - w_b(A)$, with proper quantization

Weighted Bandwidth Reduction

- For large enough α, use Weighted Spectral Ordering as a heuristic to minimize α-bandwidth
  - Find $\hat{x}$, the eigenvector corresponding to smallest non-zero eigenvalue of $\bar{L}$
  - Find permutation $\Pi = \{i_1, i_2, \ldots, i_n :$ if $j < k,\ x(i_j) < x(i_k)\}$
  - Reorder rows and columns of A accordingly to obtain $\bar{A} = A(\Pi, \Pi)$

- Observe that the heavy entries of the reordered matrix, $\bar{A}$ are close to its diagonal, *i.e.*, $\bar{A}$ has a smaller α-bandwidth compared to A

- Drop all entries that are outside α-bandwidth of $\bar{A}$

$$\tilde{A} = \{\tilde{A}(i,j) : \tilde{A}(i,j) = \bar{A}(i,j) \text{ if } |i-j| \leq BW_\alpha(\bar{A}),\ 0 \text{ else}\}$$

- Use $\tilde{A}$ as a banded preconditioner to solve the system that involves A

Experimental Results

- Matrices gathered from UF Sparse Matrix Collection
  - All software is implemented in Fortran
  - Sequential timings were done on a clovertown machine
  - BICGSTAB is used as the iterative solver
  - All matrices are first reordered using MC64, to move heaviest entries to the diagonal

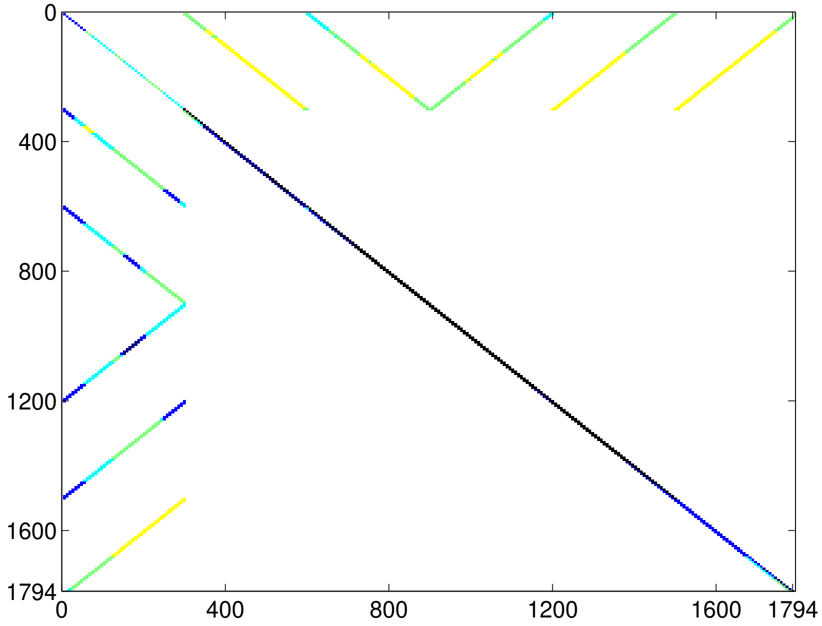
- Application of Weighted Spectral Ordering (WSO)
  - Reorder $|A| + |A^T|$ using MC73
  - Find the bandwidth that encapsulates 99% of overall matrix norm ($\alpha = 0.99$)
  - Drop entries that fall out of this bandwidth to obtain the Weighted Spectral Preconditioner

- Comparison with no preconditioner and ILU
  - We use ILUT (Saad, 1994)
  - Fill-in is set to match the storage required for dense storage of WSO's required bandwidth (ILUBW)
  - Drop tolerance is set to 10^{-1} (ILU1), 10^{-3} (ILU3)

epb0 Matrix

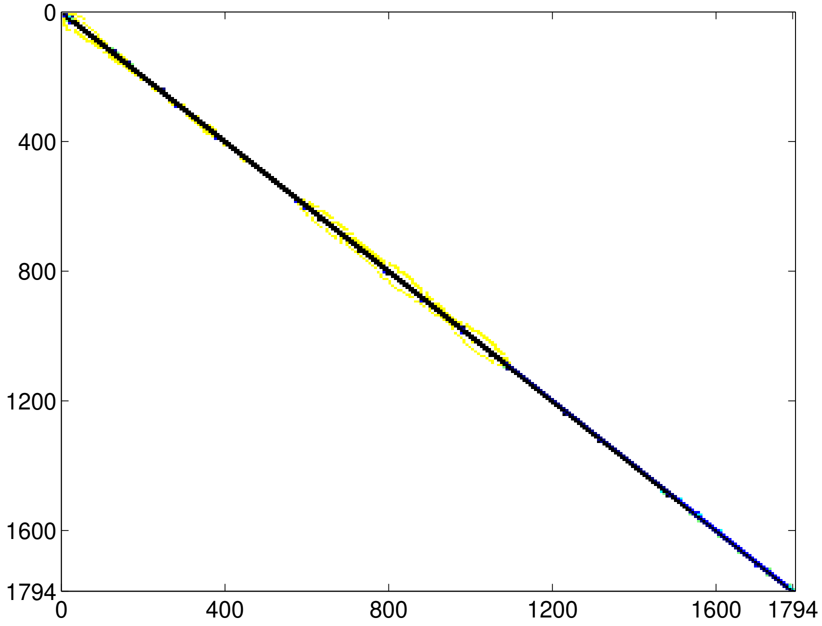
- Plate-fin heat exchanger w/ simple model
  - 1794×1794, 7764 non-zeros

Original Matrix



Bandwidth: 3587

Reordered Matrix

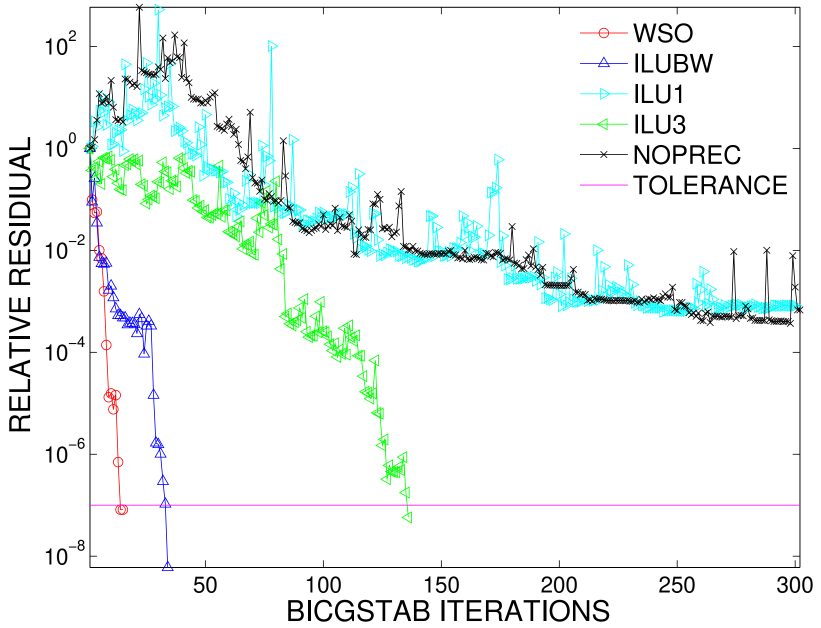


Bandwidth: 76

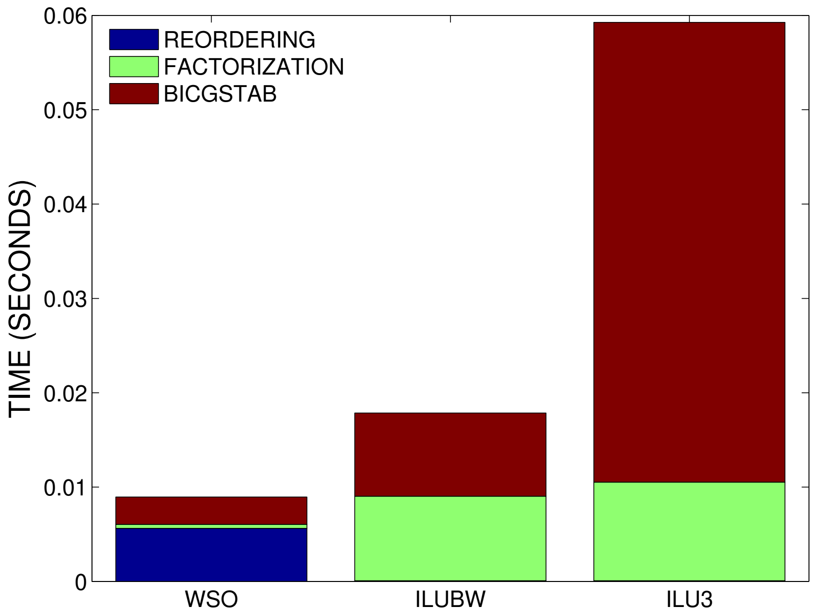
99% of matrix norm lies within a bandwidth of 19 after WSO

epb0 Results

Residual Plot



Runtime Performance

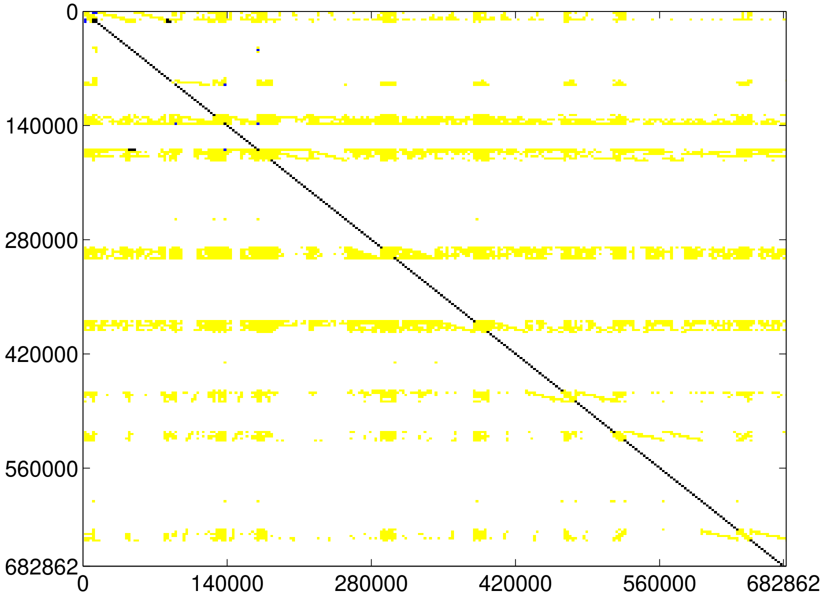


- BICGSTAB converges in 14 iterations with WSO preconditioner
  - No convergence after 300 iterations with no preconditioner or ILU with drop tolerance 10^{-3}

ASIC_680k Matrix

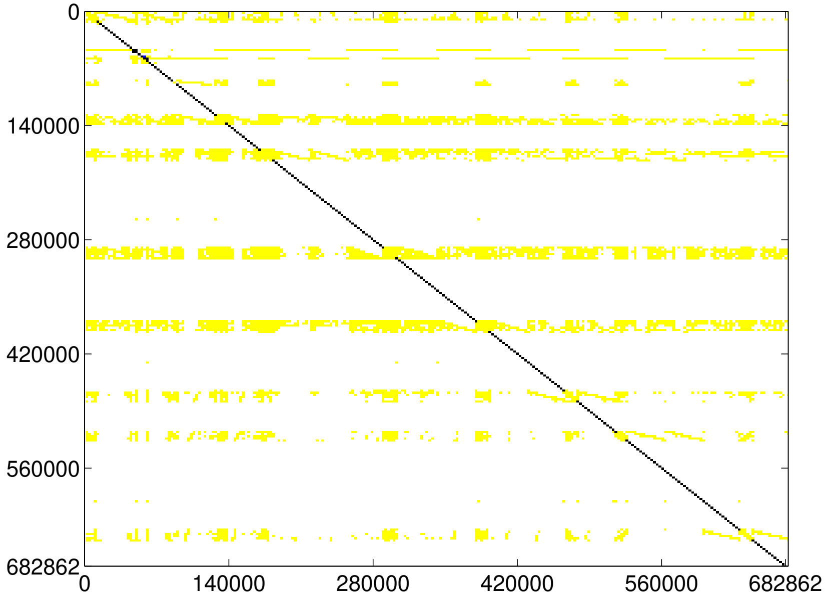
- Sandia, Xyce circuit simulation matrix (stripped)
  - 682862×682862, 2638997 non-zeros

Original Matrix



Bandwidth: 1328578

Reordered Matrix

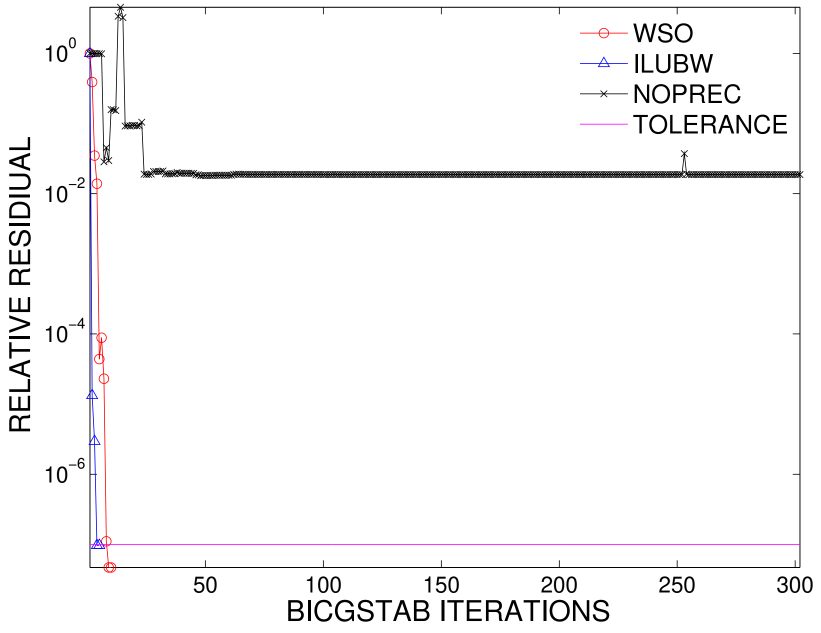


Bandwidth: 1328726

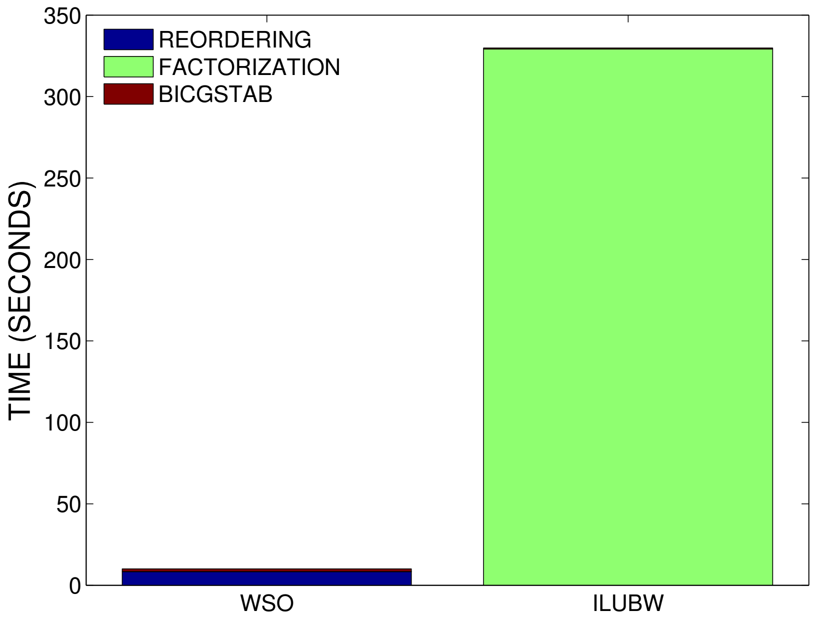
99% of matrix norm lies within a bandwidth of 5 after WSO

ASIC_680k Results

Residual Plot



Runtime Performance

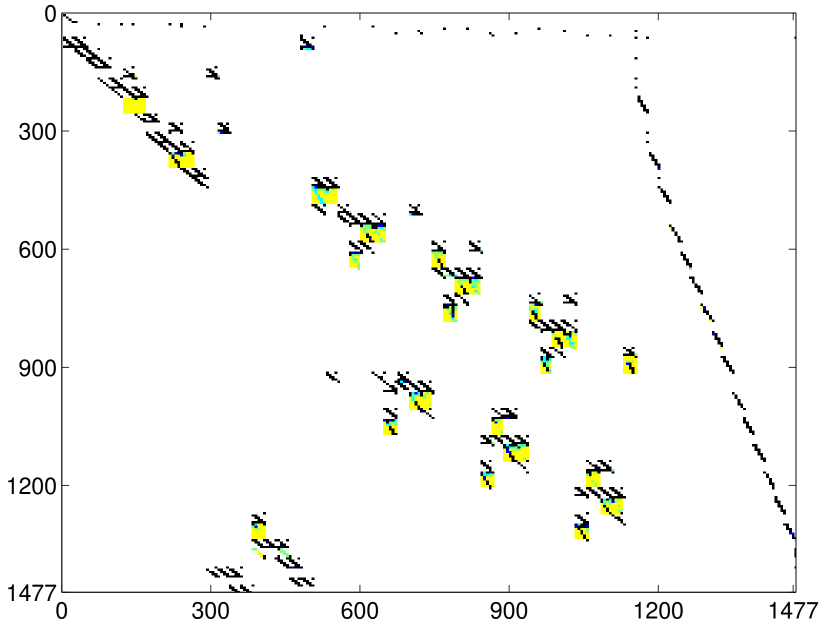


- BICGSTAB converges in 9 iterations with WSO preconditioner
  - ILU with fill-in equivalent to bandwith of WSO preconditioner converges faster (4 iterations), but factorization takes too much time
  - ILU factorization unsucessful for drop tolerance 10^{-1},10^{-3}
  - No convergence after 300 iterations with no preconditioner

lhr01 Matrix

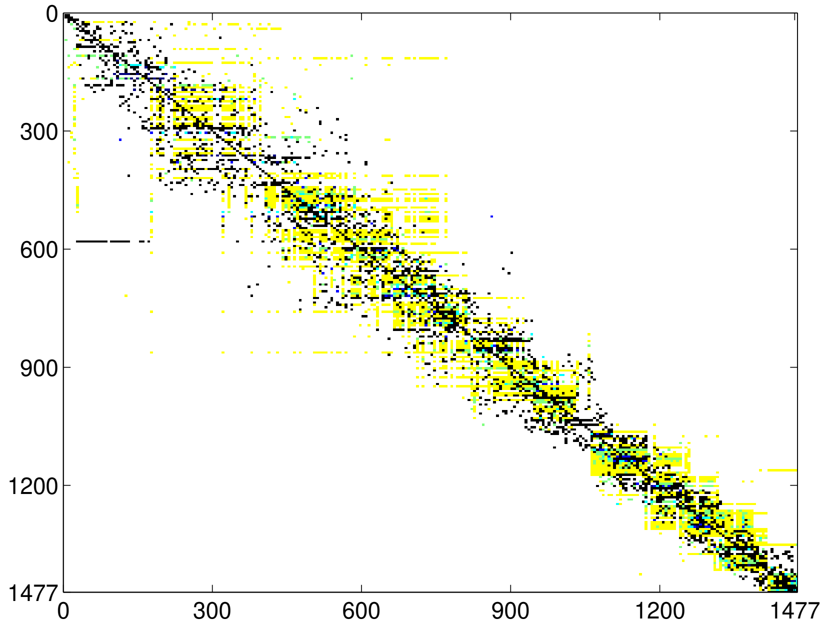
- Light hydrocarbon recovery
  - 1477×14777, 18427 non-zeros

Original Matrix



Bandwidth: 2537

Reordered Matrix

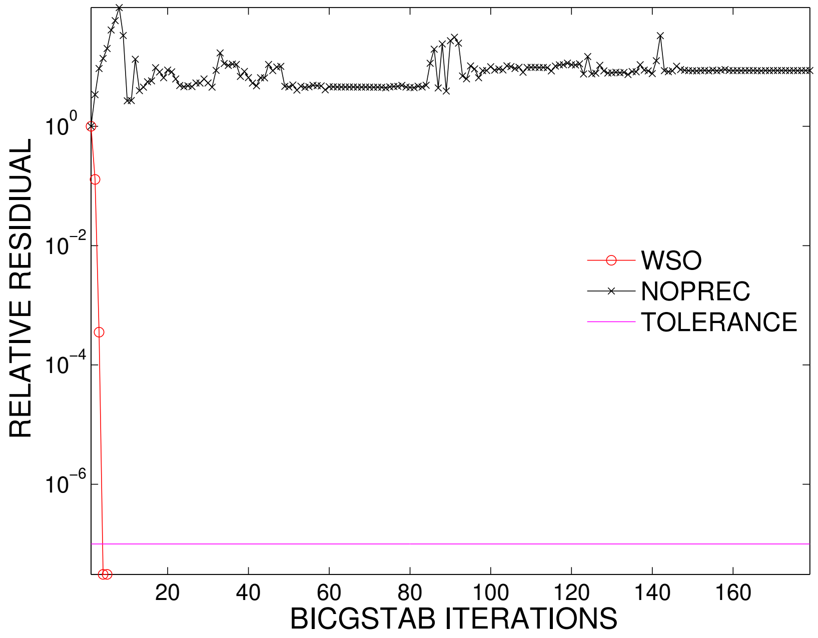


Bandwidth: 1341

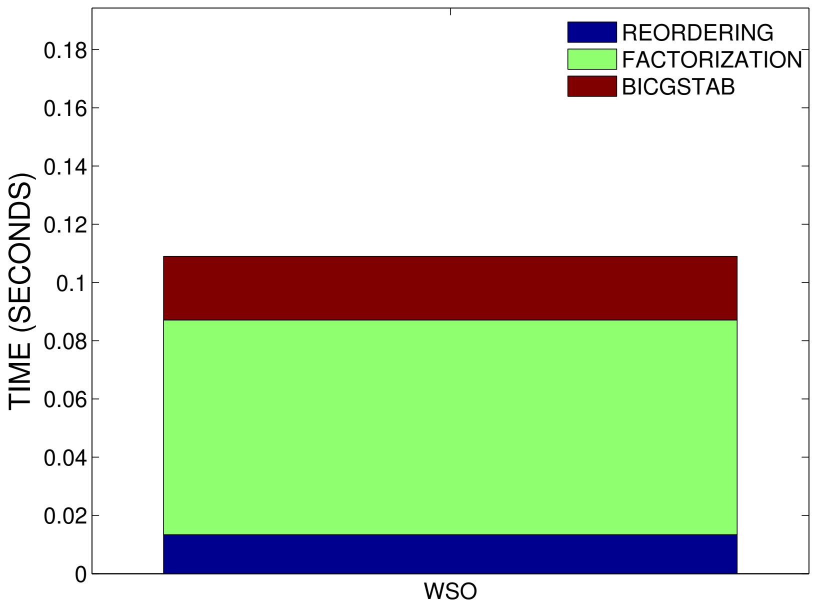
99% of matrix norm lies within a bandwidth of 601 after WSO

lhr01 Results

Residual Plot



Runtime Performance

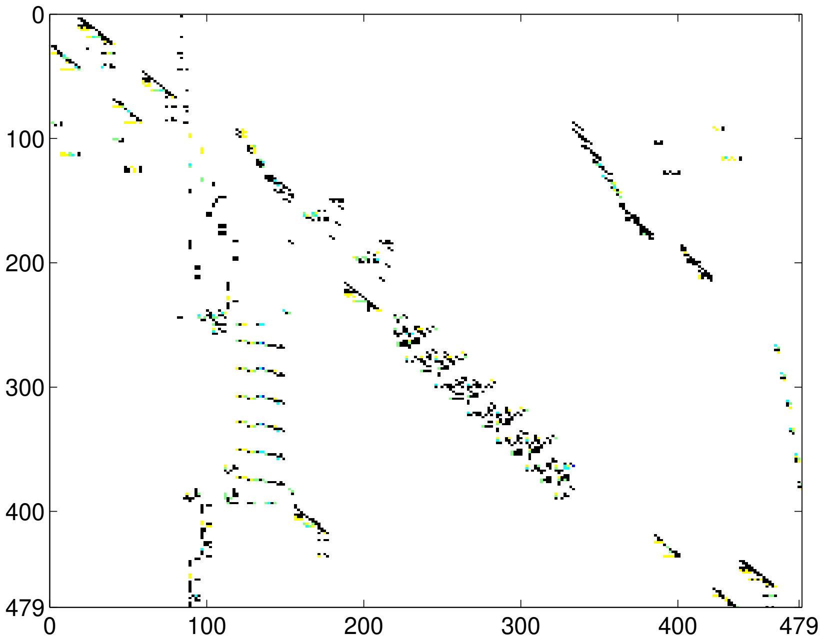


- BICGSTAB converges in only 4 iterations with WSO preconditioner!
  - No convergence with no preconditioner in 300 iterations
  - ILU factorization unsucessful for all variants

west0479 Matrix

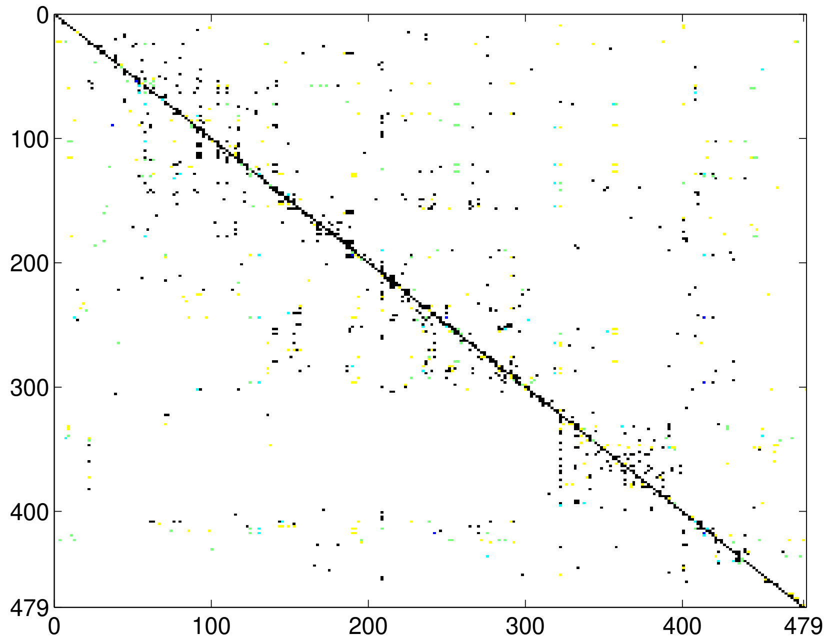
- U8 stage column section, all sections rigorous (chemical engineering)
  - 479×479, 1888 non-zeros
  - Known as a horror matrix

Original Matrix



Bandwidth: 726

Reordered Matrix

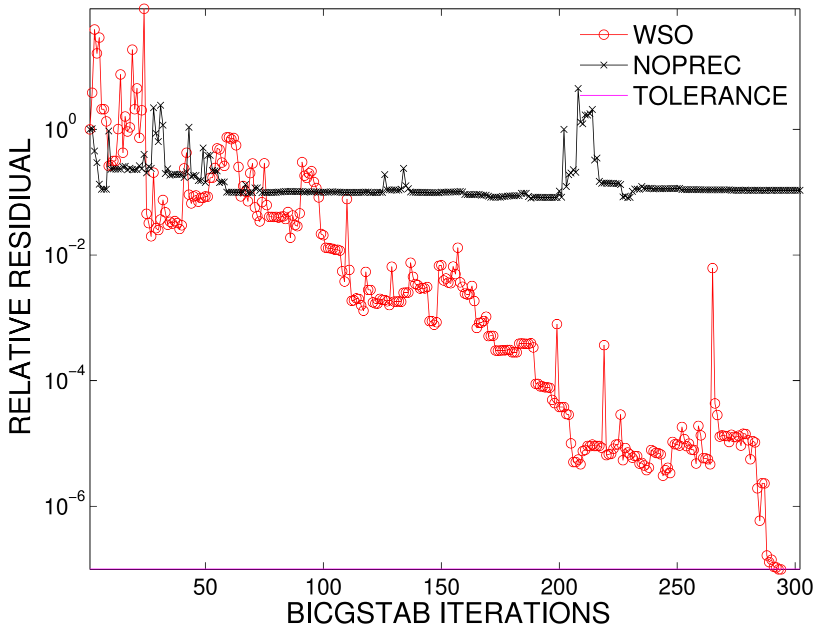


Bandwidth: 872

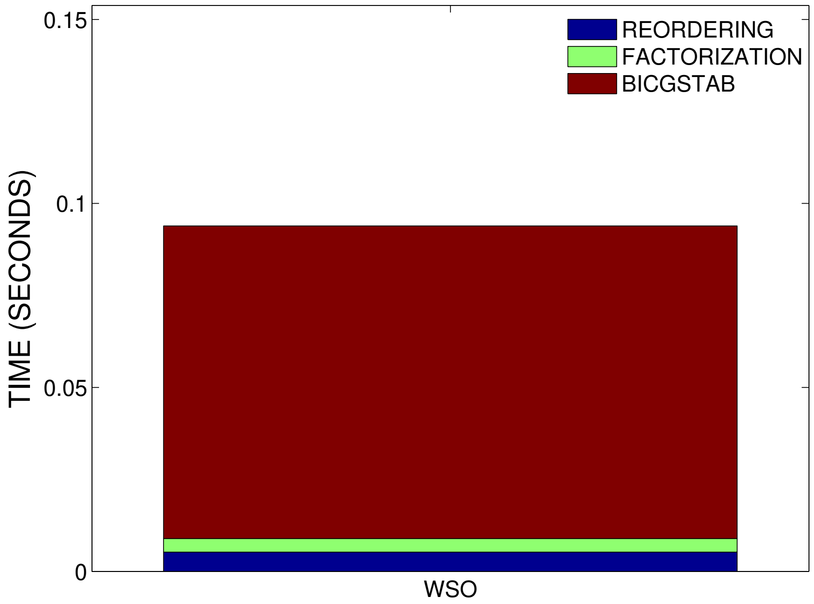
99% of matrix norm lies within a bandwidth of 221 after WSO

west0479 Results

Residual Plot



Runtime Performance

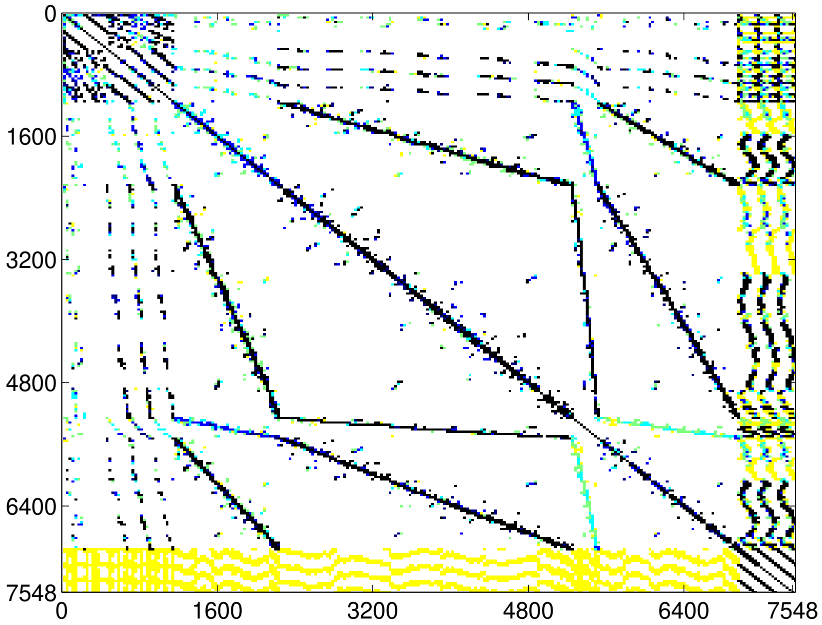


- BICGSTAB converges in 293 iterations with WSO preconditioner
  - No convergence with no preconditioner in 300 iterations
  - ILU factorization unsucessful for all variants

fp Matrix

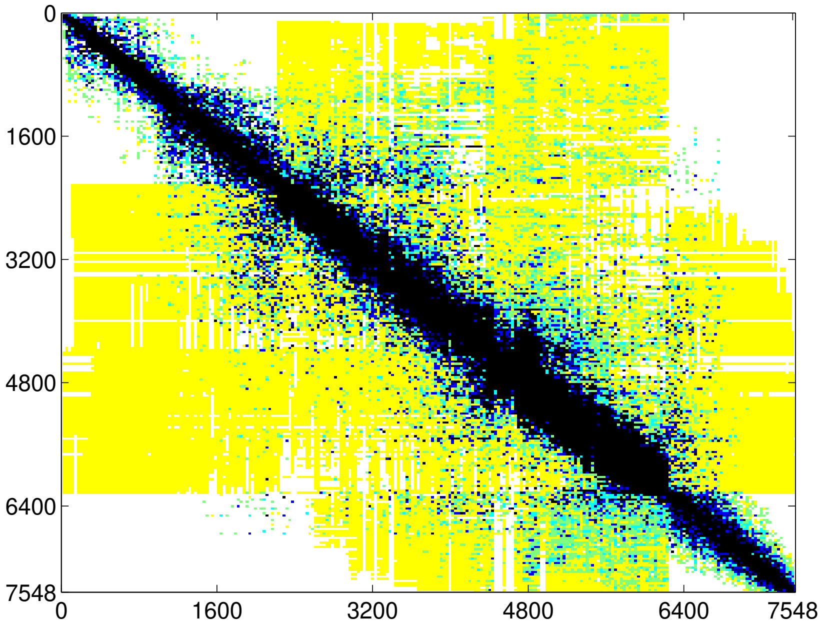
- 2-D Fokker Planck equation, electron dynamics in external field
  - 7548×7548, 834222 non-zeros

Original Matrix



Bandwidth: 15086

Reordered Matrix

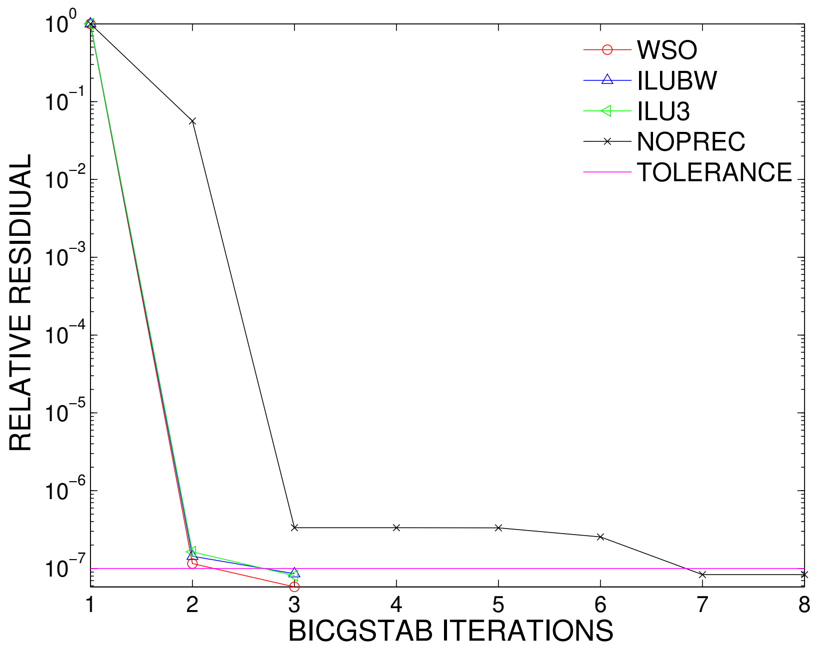


Bandwidth: 12368

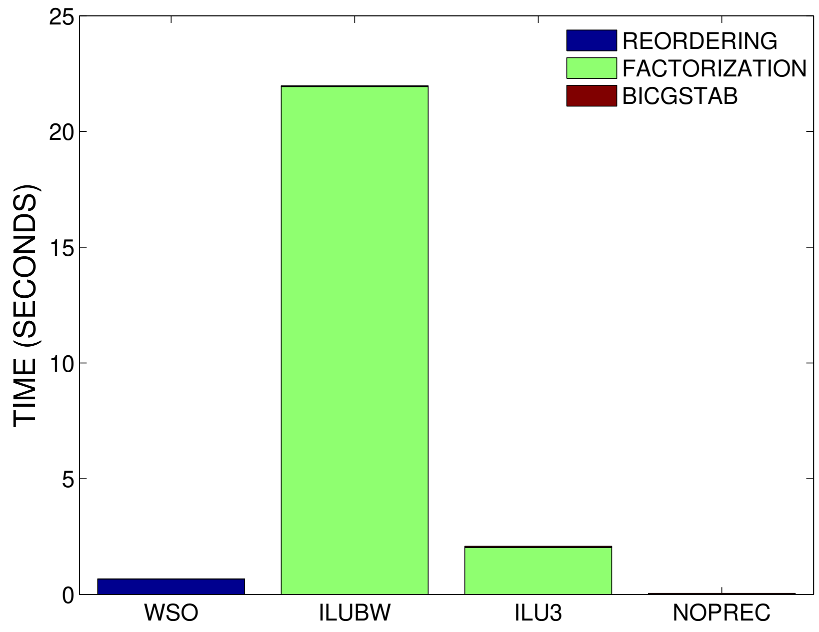
99% of matrix norm lies within a bandwidth of 5 after WSO

fp Results

Residual Plot



Runtime Performance

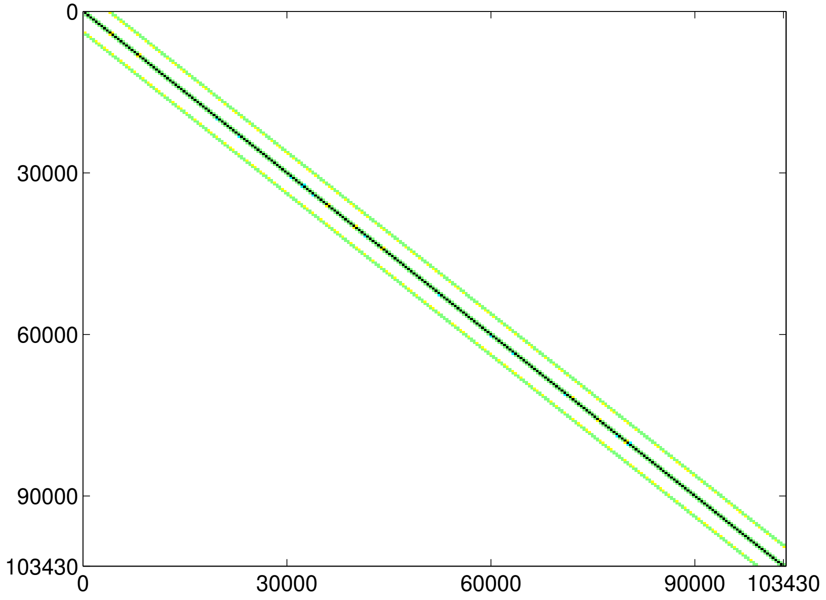


- BICGSTAB converges in 2 iterations with WSO preconditioner, as well as ILU with 10^{-3} drop tolerance, 2 fill-in
  - Convergence in 7 iterations with no preconditioner
  - Reordering, factorization take too much time as compared to preconditioner savings

matrix_9 Matrix

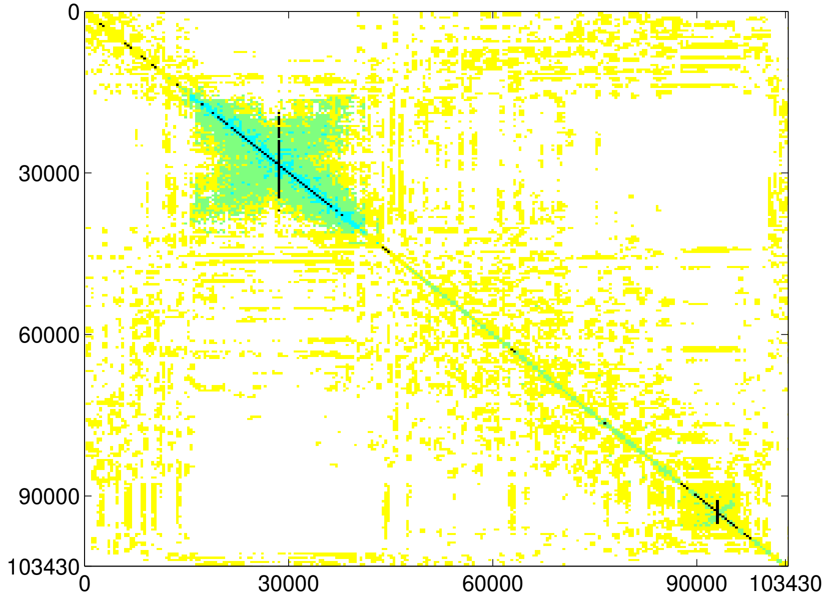
- Semiconductor device problem
  - 103430×103430, 1205518 non-zeros

Original Matrix



Bandwidth: 107410

Reordered Matrix

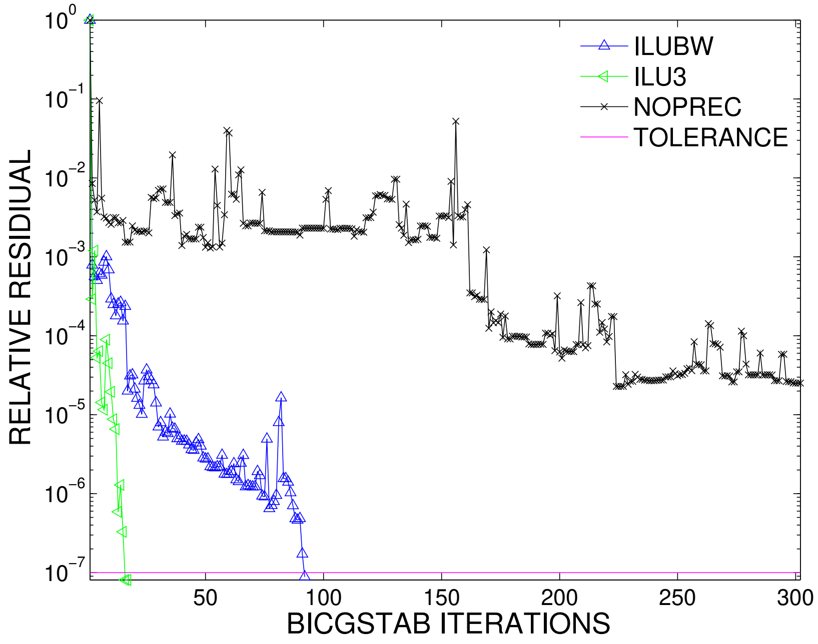


Bandwidth: 198951

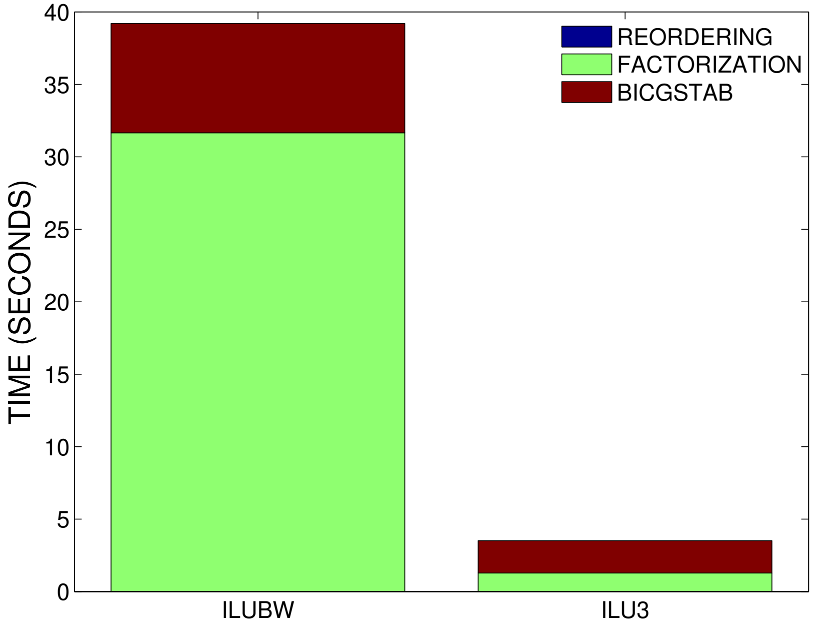
99% of matrix norm lies within a bandwidth of 10673 after WSO

fp Results

Residual Plot



Runtime Performance



- BICGSTAB runs out of memory with WSO preconditioner
  - BICGSTAB does not converge with no preconditioner
  - ILU with 10^{-3} fill-in tolerance converges in 16 iterations

Summary

	Preconditioner				
Matrix	None	ILUBW	ILU1	ILU3	WSO
epb0	> 0.019	0.018	> 0.029	0.059	0.009
	> 300	33	> 300	135	14
ASIC_680k	38.2	329.8	∞	∞	10.1
	> 300	4	∞	∞	9
lhr01	0.02	∞	∞	∞	0.11
	178	∞	∞	∞	4
west0479	> 0.009	∞	∞	∞	0.094
	> 300	∞	∞	∞	293
fp	0.05	21.97	∞	2.09	0.68
	7	2	∞	2	2
matrix_9	> 8.2	39.2	∞	3.5	∞
	> 300	91	∞	16	∞

Total runtime (reordering+factorization+bicgstab) is reported in seconds.

Number of iterations are reported on the row below.

References

(1) E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *In Proc. 24th Nat. Conf. ACM*, pages 157–172, 1969.

(2) M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 1973.

(3) Y. Saad. I ILUT: A dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.

(4) S. T. Barnard, A. Pothen, and H. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Numerical Linear Algebra with Applications*, 2(4):317–334, 1995.

(5) N. P. Kruyt. A conjugate gradient method for the spectral partitioning of graphs. *Parallel Computing*, 22(11):1493–1502, 1996.