Tradeoffs and Performance Results of the Banded PRISM Eigensolver

Christian Bischof
William L. George
Steven Huss-Lederman
Yuan-Jye Jason Wu
Argonne National Laboratory, IL

Anna Tsao
Thomas Turnbull
Center of Computing Science, MD

Xiaobai Sun
Duke University, NC

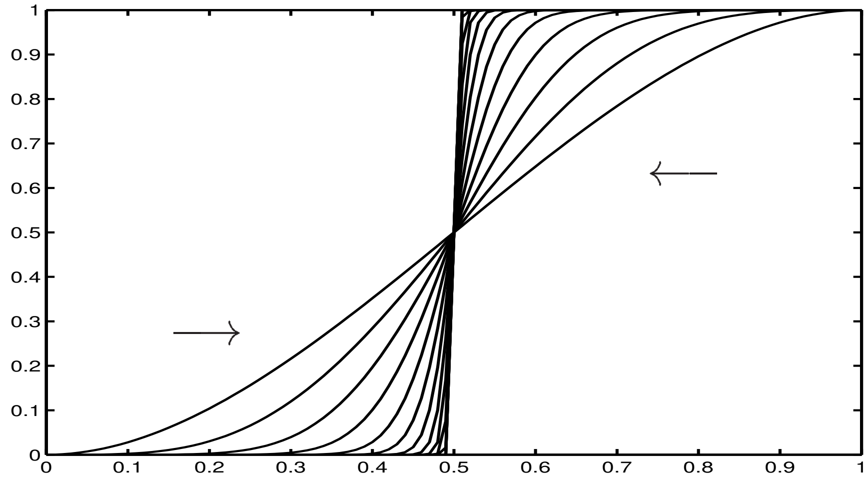
http://www.mcs.anl.gov/Projects/PRISM

This work is part of the PRISM project and in part supported by DARPA grant P-95006.

SYmmetric Invariant Subspace Decomposition Algorithm

Description of Algorithm: Let A be a symmetric matrix.

- Scaling:
  - Compute bounds on $\lambda(A)$ (e.g. Gershgorin disks)
  - Compute α and β so that for $B = \alpha A + \beta I$, $\lambda(B) \subseteq [0, 1]$ with mean eigenvalue of A being mapped to $1/2$
- Eigenvalue smoothing:
  - Find a sequence of polynominals $p_i(x)$ such that for all $x \in [0, 1]$,
  $$p_i(p_{i-1}(\cdots(p_0(x))\cdots)) \longrightarrow \{0, 1\} \text{ as } i \longrightarrow \infty$$
  - Iterate $C_0 = B, C_{i+1} = p_i(C_i), i = 0, 1, \ldots$; then we have
  $$C_\infty = \lim_{i\to\infty} C_i \quad \text{and} \quad \lambda(C_\infty) = \{0, 1\}$$
  - example of polynominals: mapping after applying the first incomplete Beta function $p(x) = 3x^2 - 2x^3$ ten times



- Invariant subspace computation:
  Find an orthonormal matrix $[U\ V]$ such that

$$C_\infty [U\ V] = [U\ V] \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$$

- Decoupling:
  Update the original A with $[U\ V]$, i.e., form

$$[U\ V]^T A\, [U\ V] = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

- Iteration:
  Apply the same routine to submatrices A_1 and A_2

Remarks of SYISDA:

- Richness of Blas-3 operations:
  - Matrix-matrix multiplication in polynominals
  - Block accumulation on orthogonal matrix
- Compute the matrix $[U\ V]$:
  - Tridiagonalize C_∞ to a tridiagonal matrix T.
  - Form orthogonal eigenvectors of T (Bischof and Sun, 1992): a tridiagonal matrix with only two repeated eigenvalues is block diagonal with block size at most 2, i.e.,
  $$T = \begin{bmatrix} T_1 & & 0 \\ & \ddots & \\ 0 & & T_k \end{bmatrix},$$
  where T_i is of order either 1 or 2, $i = 1, \ldots, k$.
- Parallel implementation:
  - Communication interface: MPI
  - Data parallel: larger order of A_1 and A_2
  - Task parallel: small order of A_1 and A_2

Advantages of SYISDA:

- Ordering of eigenvalues:
  - Free choice of the middle point $a \in (0, 1)$
  - Separate eigenvalues of A so that

$$\begin{aligned} \lambda(A_1) &= \{\lambda | \lambda \in \lambda(A) \text{and} \lambda < (a - \beta)/\alpha)\} \\ \lambda(A_2) &= \{\lambda | \lambda \in \lambda(A) \text{and} \lambda \geq (a - \beta)/\alpha)\} \end{aligned}$$

- Subspaces before eigenvalues:

  SYISDA computes and refines invariant subspaces before eigenvalues. For applications that need only bases for some eigen-subspaces, one can terminate the iteration once the bases are formed.

- No problems with repeated eigenvalues:

  Clusters of eigenvalues are quickly gathered in subproblem. It will increase the speed of convergence of the smoothing step.

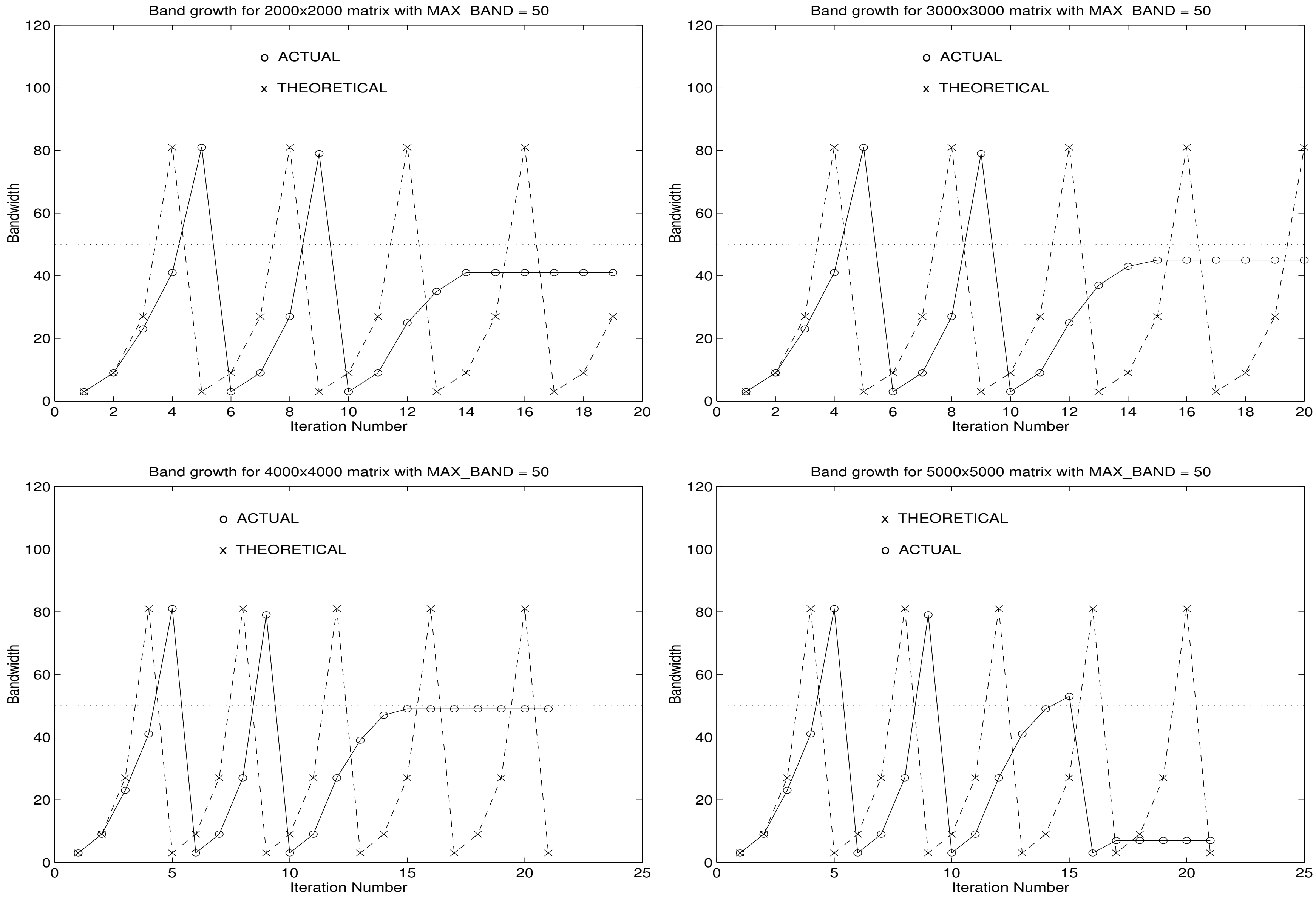
Banded SYISDA

Ideas:
In the SYISDA algorithm, dense matrix multiplications performed in the iterator polynomial step dominate the computational cost. Note that the multiplication of two $n \times n$ matrices of bandwidths b_1 and b_2, resulting in a matrix of bandwidth $(b_1 + b_2)$, requires only $O(b_1 b_2 n)$ work versus $O(n^3)$ for two dense matrices. This gives us the motivation to employ banded matrix operations to SYISDA.

Additional routines:

- In the Eigenvalue Smoothing step:
  - reduce the full, scaled matrix to a banded matrix first
  - replace dense matrix multiplication by banded matrix multiplication
  - perform band reductions, as necessary
- In the Invariant Subspace Computation step:
  apply band-to-tridiagonal reduction only to the rank-revealing tridiagonalization.

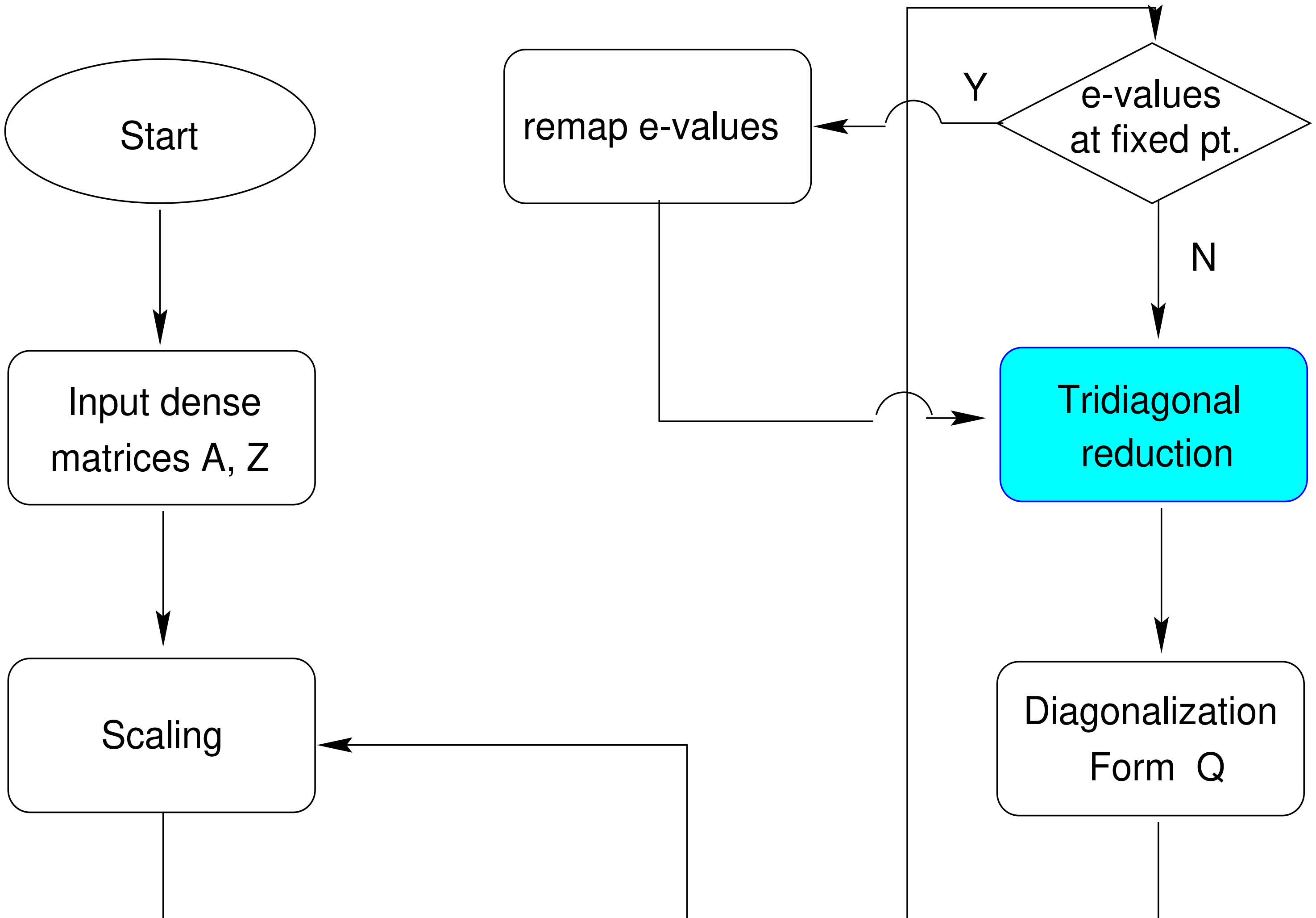
Interesting observation: Growth rate of bandwidth decreases when the matrix converges.

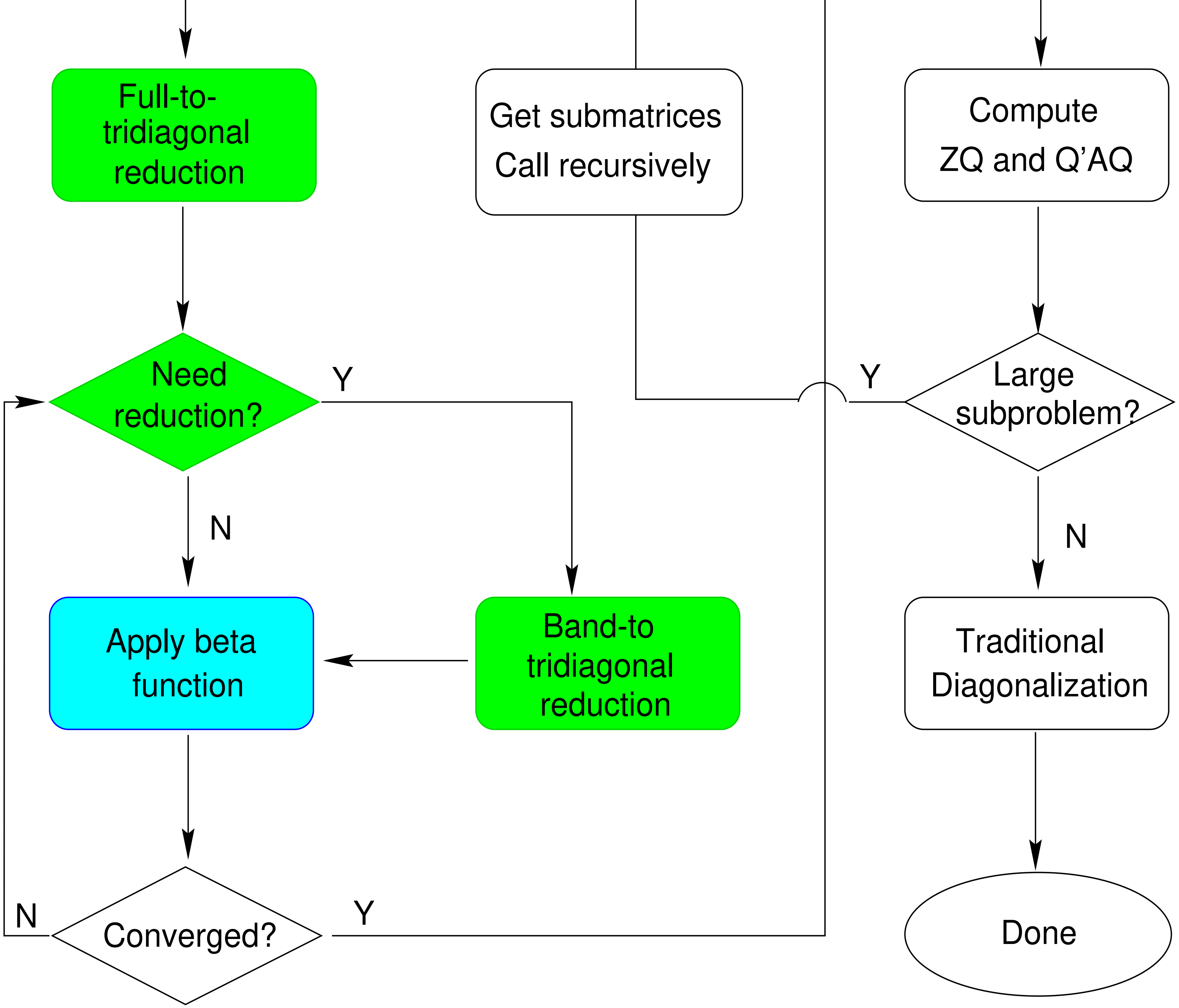


Flowchart of the dense and banded SYISDA algorithm:

Green Box : additional step for banded SYISDA

Cyan Box : different routines for dense and banded SYISDA

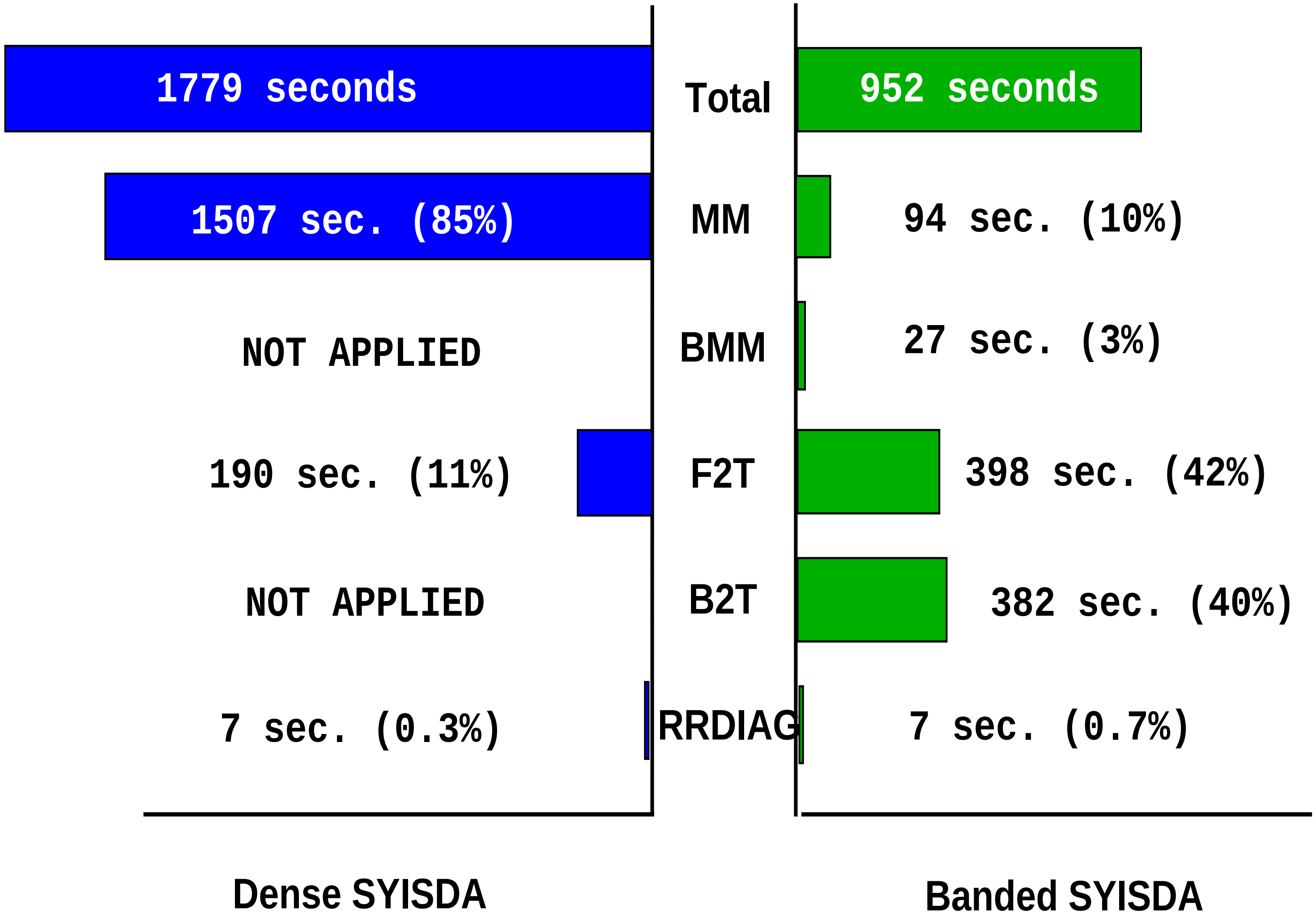



Full-to-tridiagonal reduction
Get submatrices
Call recursively
Compute
ZQ and Q'AQ
Need reduction?
Y
N
Large subproblem?
Y
N
Apply beta function
Band-to tridiagonal reduction
Traditional Diagonalization
Converged?
N
Y
Done

Timings for Dense and Banded SYISDA eigensolvers on IBM SP2

Mesh	Matrix Size	Dense SYISDA			Banded SYISDA			Speedup
		Time(sec)	||QQ'-I||	||AQ-QT||	Time(sec)	||QQ'-I||	||AQ-QT||	Blue/Green
5 X 5	1000	156	9e-15	2e-15	126	1e-14	4e-15	1.24
	2000	608	1e-14	5e-15	437	1e-14	4e-15	1.39
	3000	1779	1e-14	1e-14	952	1e-14	1e-14	1.87
	4000	4015	2e-14	1e-14	2049	2e-14	7e-15	1.95
6 X 6	1000	170	1e-14	2e-15	207	1e-14	4e-15	0.82
	2000	481	1e-14	7e-15	420	1e-14	1e-14	1.16
	3000	1392	2e-14	2e-14	897	2e-14	7e-15	1.55
	4000	3114	2e-14	1e-14	1747	1e-14	7e-15	1.78
8X8	4000	2056	1e-14	1e-14	1481	2e-14	1e-14	1.39

Finding eigenvalues & eigenvectors for a matrix of order 3000 on 5x5 mesh



Preliminary Timing Results

uniformly distributed e-values (time uint=second)				
mesh	matrix size	BSYISDA*	ScaLAPACK	Ratio
5x5	2000	471.9	103.4	4.56
	3000	961.3	277.3	3.47
	4000	2011	583.3	3.45
6x6	3000	894.9	214.8	4.17
	4000	1741	433.2	4.02
	5000	3249	771.1	4.21
cluster e-values				
3x3	1000	101.3	44.74	2.26
	2000	545.1	211.0	2.58
4x4	2000	374.3	188.5	1.99

∗ timing for the prototype code without any optimization

Conclusions and Future Work

- Robustness on any case, even for cluster eigenvalues
- Fine tuning on band reduction and tridiagonalization (data redistribution, nodes assignment)
- Using PLAPACK infrastructure (promising preliminary results)