Specialized Spectral Division Algorithms for Generalized Eigenproblems via the Matrix Disk Function

Mercedes Marqués, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí

Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I de Castellón (Spain)
{mmarques,quintana,gquintan}@icc.uji.es

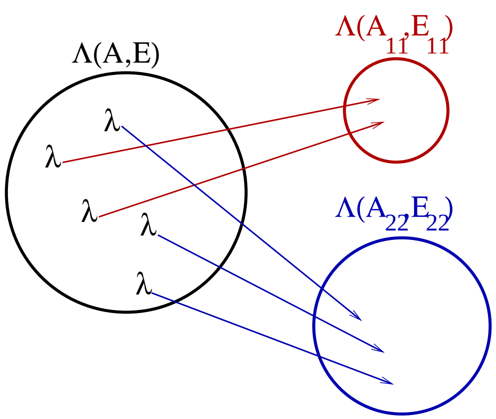
PARA'06 - June 2006

Spectral Division

Given $A,\ E \in \mathbb{R}^{n\times n}$, with (generalized) eigenspectrum $\Lambda(A,E)$, find orthogonal $U,\ V \in \mathbb{R}^{n\times n}$ s.t.

$$U^T A V = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad U^T E V = \begin{bmatrix} E_{11} & E_{12} \\ 0 & E_{22} \end{bmatrix},$$

where $\Lambda(A_{11}, E_{11})$, $\Lambda(A_{22}, E_{22})$ are disjoint



Spectral Division: Applications

- LA: Block diagonalization
- Control: Partial stabilization, model reduction, optimal control
  - Solution of linear matrix equations:

$$A^T X A - X + Q = 0, \quad A^T X + X A + Q = 0, \ldots$$

  - Solution of algebraic Riccati equations:

$$A^T X + X A - X G X + Q = 0, \ldots$$

Spectral Division: Numerical Tools

- Reduction to generalized real Schur form + reordering
  - QZ algorithm *(Moler and Stewart, SINUM'73)*:
- Matrix sign function
  - Spectral division along the imaginary axis *(Roberts, IJControl'80)*:
- Matrix disk function (MDF)

"Inverse-free" Iteration for MDF (Review)

Traditional iteration *(Malyshev, LAA'93)*:

Let $A_0 \leftarrow A$, $E_0 \leftarrow E$,
For $k = 0, 1, 2, \ldots$

$$\begin{aligned} \begin{bmatrix} E_k \\ -A_k \end{bmatrix} &= Q_k \bar{R}_k = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} R_k \\ 0 \end{bmatrix} \\ A_{k+1} &\leftarrow Q_{12}^T A_k, \\ E_{k+1} &\leftarrow Q_{22}^T E_k \end{aligned}$$

The iteration is followed by subspace extraction $\rightarrow$ spectral division along the unit circle

Inverse-free Iteration for MDF (Review)

Truly inverse-free algorithm *(Bai, Demmel, and Gu, Numer. Math.'97)*:

- Inverse-free convergence criterion

$$\|R_{k+1} - R_k\|_F < \tau \|R_k\|_F.$$

- Inverse-free subspace extraction
- Computation of U and V requires double iteration, on (A, E) and (A^T, E^T)

Inverse-free Iteration for MDF (Review)

Simultaneous subspace extraction *(Sun and Quintana-Ortí, Math. Comp.'04)*:

- Compute U and get V (almost) for free
- No need for a second iteration (savings $\approx$ 50%!)

Inverse-free Iteration for MDF (Review)

Traditional implementation:

1. Compute the QR factorization (DGEQRF):

$$\begin{bmatrix} E_k \\ -A_k \end{bmatrix} = Q_k \bar{R}_k = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$$

2. Construct Q_{12}, Q_{22} by accumulating the Householder reflectors on $[0_n, \ I_n]^T$ in reverse order (DORMQR).

3. Compute the matrix products (DGEMM)

$$A_{k+1} \leftarrow Q_{12}^T A_k, \quad E_{k+1} \leftarrow Q_{22}^T E_k$$

Total: $13n^3 + n^3/3$ flops

Comparison of Spectral Division Tools

- Reduction to generalized real Schur form (w/out reordering): $81n^3$ flops (average)
- Generalized matrix sign function: By initially reducing E to bidiagonal form, $2n^3$ flops per iteration *(Sun and Quintana-Ortí, SISC'02)*
- Matrix disk function (MDF):
  - 6 inverse-free iterations $\approx$ reduction to generalized Schur form
  - More than 6 times as expensive per iteration as the matrix sign function

Reducing the Cost of the Inverse-free Iteration

Rationale: Use Givens rotations to keep the sequence $\{A\}_{k=0}^{\infty}$ upper triangular

Let $A_0 - \lambda E_0 = R_A - \lambda(U_A^T E)$ where $A = U_A R_A$ is a QR factorization. Then, for $n = 3$

$$M_0 = \left[\begin{array}{c} E_0 \\ \hline -A_0 \end{array}\right] = \left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \hline \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{array}\right],$$

Reducing the Cost of the Inverse-free Iteration

Apply Givens rotations to reduce M_k by using Givens rotations in the following order:

$$G_1^{4,1}\left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \hline \otimes & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{array}\right] \Rightarrow G_2^{3,1}\left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \\ \otimes & \times & \times \\ \hline 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{array}\right] \Rightarrow G_3^{2,1}\left[\begin{array}{ccc} \times & \times & \times \\ \otimes & \times & \times \\ 0 & \times & \times \\ \hline 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{array}\right] \Rightarrow$$

$$G_4^{5,2}\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ \hline 0 & \times & \times \\ 0 & \otimes & \times \\ 0 & 0 & \times \end{array}\right] \Rightarrow G_5^{4,2}\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ \hline 0 & \otimes & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{array}\right] \Rightarrow G_6^{3,2}\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & \otimes & \times \\ \hline 0 & 0 & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{array}\right] \Rightarrow$$

Reducing the Cost of the Inverse-free Iteration

$$G_7^{6,3}\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & \times \\ 0 & 0 & \times \\ 0 & 0 & \otimes \end{array}\right] \Rightarrow G_8^{5,3}\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & \times \\ 0 & 0 & \otimes \\ 0 & 0 & 0 \end{array}\right] \Rightarrow G_9^{4,3}\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & \otimes \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \Rightarrow$$

$$\left[\begin{array}{ccc} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

Reducing the Cost of the Inverse-free Iteration

If we now accumulate the rotations on $[0_n,\ I_n]^T$ in reverse order:

$$G_9^{4,3}\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline \times & 0 & 0 \\ 0 & \times & 0 \\ 0 & 0 & \times \end{array}\right] \Rightarrow G_8^{5,3}\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \oplus & 0 & 0 \\ \hline \times & 0 & 0 \\ 0 & \times & 0 \\ 0 & 0 & \times \end{array}\right] \Rightarrow G_7^{6,3}\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \times & 0 & 0 \\ \hline \times & \oplus & 0 \\ \oplus & \times & 0 \\ 0 & 0 & \times \end{array}\right] \Rightarrow$$

$$G_6^{3,2}\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \times & 0 & 0 \\ \hline \times & \times & 0 \\ \times & \times & \oplus \\ \oplus & \oplus & \times \end{array}\right] \Rightarrow G_5^{4,2}\left[\begin{array}{ccc} 0 & 0 & 0 \\ \oplus & 0 & 0 \\ \times & 0 & 0 \\ \hline \times & \times & 0 \\ \times & \times & \times \\ \times & \times & \times \end{array}\right] \Rightarrow G_4^{5,2}\left[\begin{array}{ccc} 0 & 0 & 0 \\ \times & 0 & 0 \\ \times & \oplus & 0 \\ \hline \times & \times & 0 \\ \times & \times & \times \\ \times & \times & \times \end{array}\right] \Rightarrow$$

Reducing the Cost of the Inverse-free Iteration

$$G_3^{2,1}\left[\begin{array}{ccc} 0 & 0 & 0 \\ \times & 0 & 0 \\ \times & \times & 0 \\ \hline \times & \times & \oplus \\ \times & \times & \times \\ \times & \times & \times \end{array}\right] \Rightarrow G_2^{3,1}\left[\begin{array}{ccc} \oplus & 0 & 0 \\ \times & 0 & 0 \\ \times & \times & 0 \\ \hline \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{array}\right] \Rightarrow G_1^{4,1}\left[\begin{array}{ccc} \times & 0 & 0 \\ \times & \oplus & 0 \\ \times & \times & 0 \\ \hline \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{array}\right] \Rightarrow$$

$$\left[\begin{array}{ccc} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \oplus \\ \hline \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{array}\right] = \left[\begin{array}{c} Q_{12} \\ Q_{22} \end{array}\right]$$

Thus, $A_1 = Q_{12}^T A_0$ maintains the upper triangular structure

Reducing the Cost of the Inverse-free Iteration

Computational costs:

Step	Traditional	Givens-based
QR fact.	$3n^3 + n^3/3$	$3n^3$
Accumulate Q_k	$6n^3$	$3n^3$
$A_{k+1} \leftarrow Q_{12}^T A_k$	$2n^3$	n^3
$E_{k+1} \leftarrow Q_{22}^T E_k$	$2n^3$	$2n^3$
Total	$13n^3 + n^3/3$	$9n^3$

A saving of 32% per iteration!

However, we have changed a code based on BLAS-3 operations by one based on BLAS-1!

High Performance Algorithm

Rationale: Use Householder reflectors to keep the sequence $\{A\}_{k=0}^{\infty}$ block upper triangular

Consider a blocked partitioning:

$$M_0 = \left[\begin{array}{c} E_0 \\ \hline -A_0 \end{array}\right] = \left[\begin{array}{ccc} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \\ \hline M_{41} & M_{42} & M_{43} \\ 0 & M_{52} & M_{53} \\ 0 & 0 & M_{63} \end{array}\right]$$

with blocks of dimension $b \times b$

High Performance Algorithm

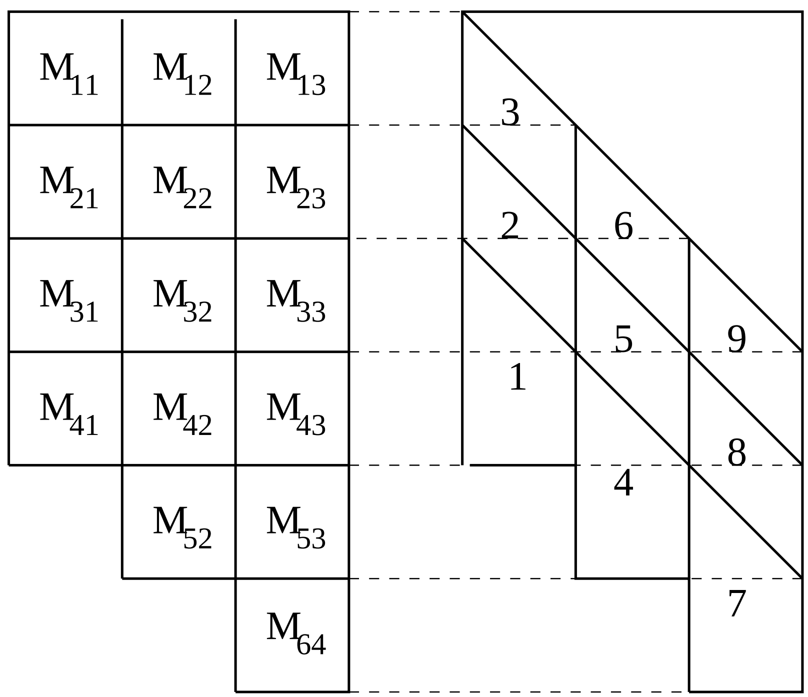
Then, M_0 is reduced to upper triangular form using Householder reflectors to triangularize $2b \times b$ matrices, as in

$$\begin{bmatrix} M_{31} \\ M_{41} \end{bmatrix} = U_1^{4,1} \begin{bmatrix} R_{31} \\ 0 \end{bmatrix}$$

The factorization of the $2b \times b$ blocks can be performed in many different ways: BLAS-2/block Householder, exploit the structure in the lower half, etc.

High Performance Algorithm

Blocks are annihilated exactly in the same order showed for the Givens-based algorithm



High Performance Algorithm

...and the product of the Householder reflectors results in a block triangular matrix Q_{12}

$$Q_1 Q_2 \cdots Q_9 \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline I_b & 0 & 0 \\ 0 & I_b & 0 \\ 0 & 0 & I_b \end{array}\right] = \left[\begin{array}{ccc} V_{11} & 0 & 0 \\ V_{21} & V_{22} & 0 \\ V_{31} & V_{32} & V_{33} \\ \hline V_{41} & V_{42} & V_{43} \\ V_{51} & V_{52} & V_{53} \\ V_{61} & V_{62} & V_{63} \end{array}\right] = \left[\begin{array}{c} Q_{12} \\ Q_{22} \end{array}\right]$$

Now, $A_1 = Q_{12}^T A_0$ is block upper triangular

High Performance Algorithm

Computational costs: Provided $b \ll n$

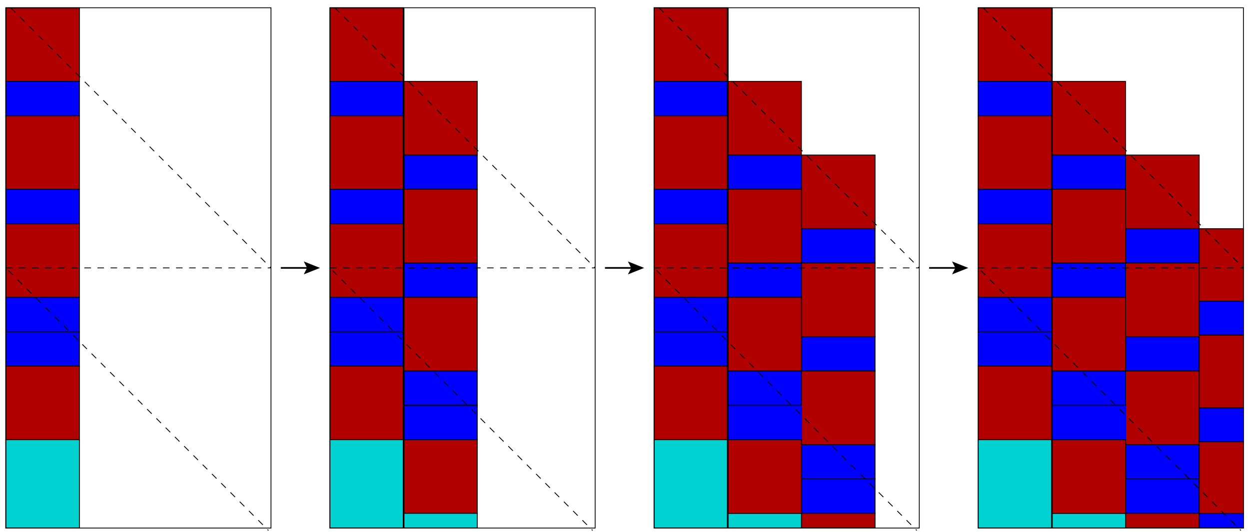
Step	Traditional	Givens-based	Blocked Householder
QR fact.	$3n^3 + n^3/3$	$3n^3$	$3n^3$
Accumulate Q_k	$6n^3$	$3n^3$	$3n^3$
$A_{k+1} \leftarrow Q_{12}^T A_k$	$2n^3$	n^3	n^3
$E_{k+1} \leftarrow Q_{22}^T E_k$	$2n^3$	$2n^3$	$2n^3$
Total	$13n^3 + n^3/3$	$9n^3$	$9n^3$

A saving of 32% per iteration!

In case b is large enough, use of BLAS-3 operations is possible!

High Performance Algorithm

If n is not an exact multiple of b, we need a careful partitioning blocks so that Q_{12} is block triangular: Any partitioning on the first column block needs to be reproduced from the diagonal of any other column block



For simplicity, we choose all blocks of the same row size except for the last in each column block

Experiments

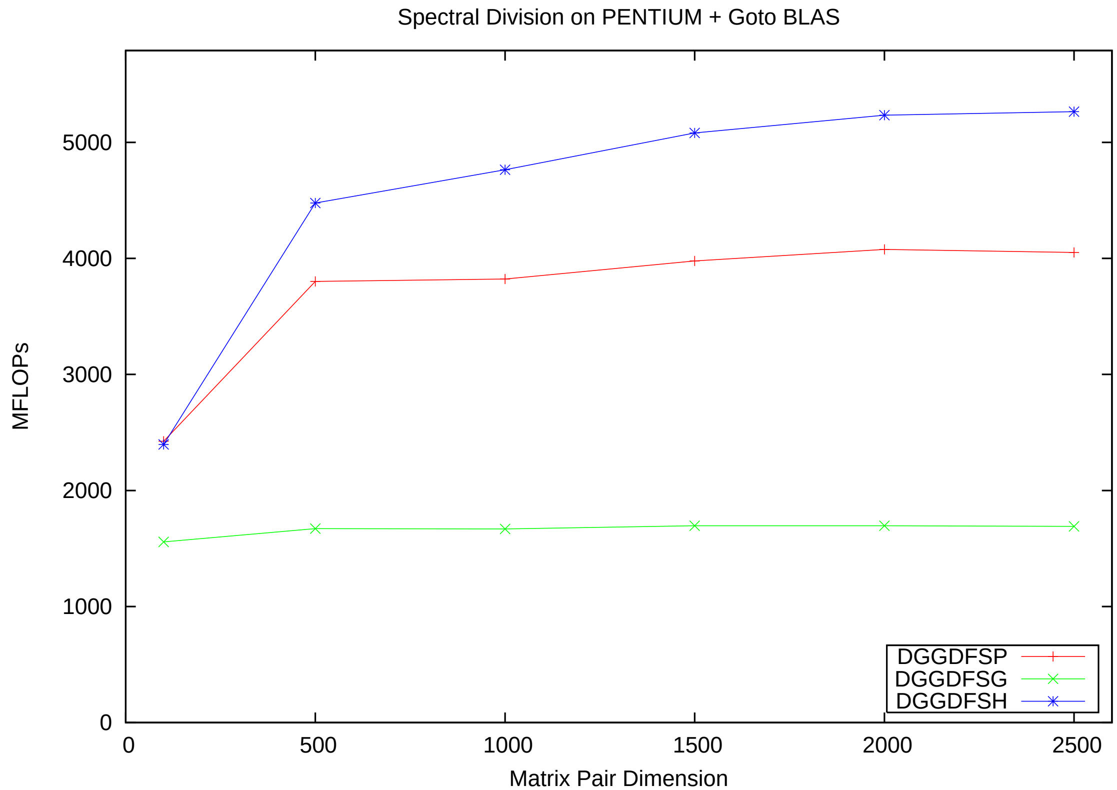
- IEEE double-precision arithmetic
- Routines:
  - DGGDFSP. The traditional inverse-free iterative scheme.
  - DGGDFSG. The Givens-based inverse-free iteration.
  - DGGDFSH. Inverse-free iteration using the blocked Householder high performance algorithm.
  - DGGDFSX. Reduction to generalized real Schur form (QZ algorithm) + reordering procedure.

Experiments

- Architectures+BLAS:
  - PENTIUM: Intel Pentium4 processor@3.2 GHz with 2048 KB of L2 cache and Goto BLAS
  - ITANIUM: Intel Itanium-2 processor@1.5 GHz with 256 KB/4 MB of L2/L3 cache and MKL 8.0
- MFLOPs (millions of flops per second) using normalized flop count of $13.3n^3$ flops per iteration for all iterative schemes
- Remember: at most we aim at attaining a reduction of execution time by 32%!

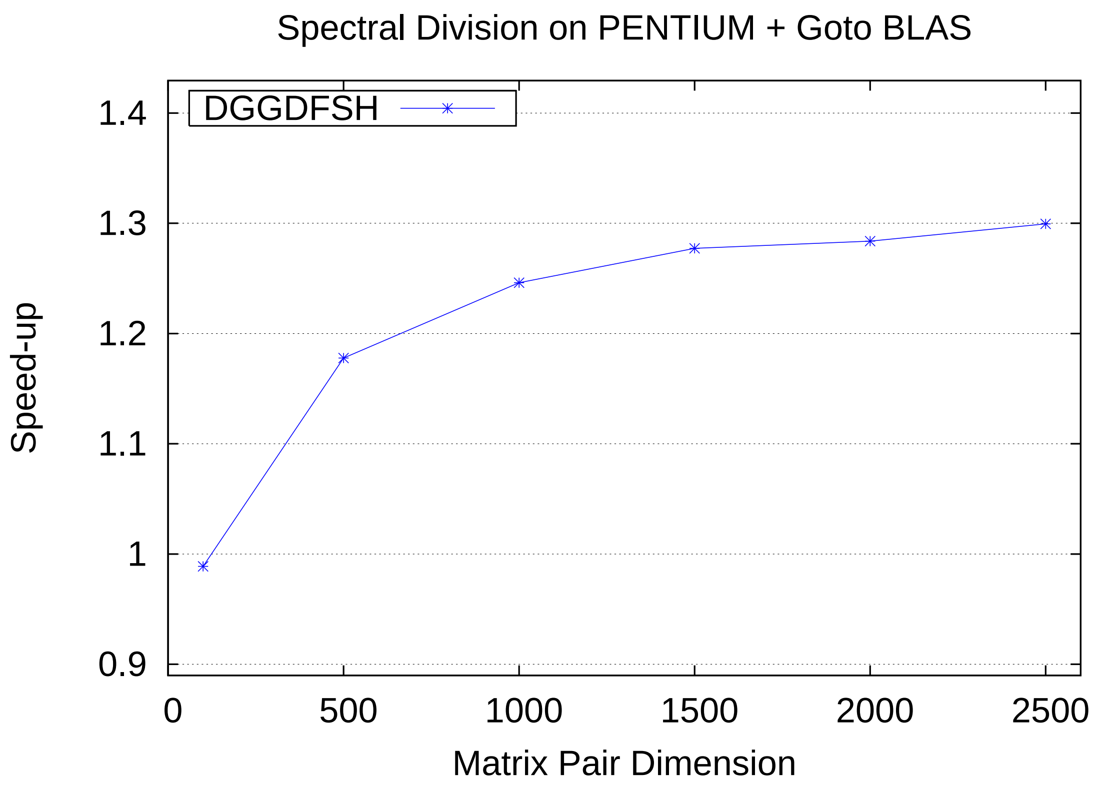
Experiments

MFLOPs on PENTIUM



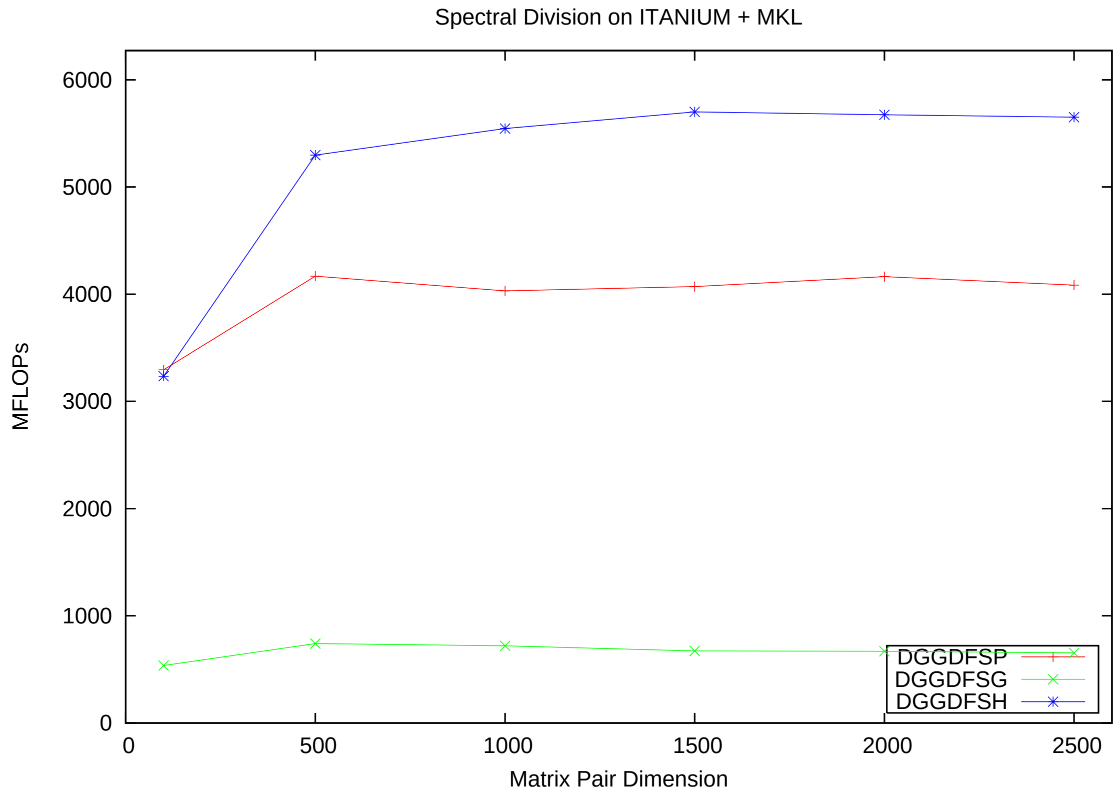
Experiments

Speed-up of DGGDFSH over DGGDFSP on PENTIUM



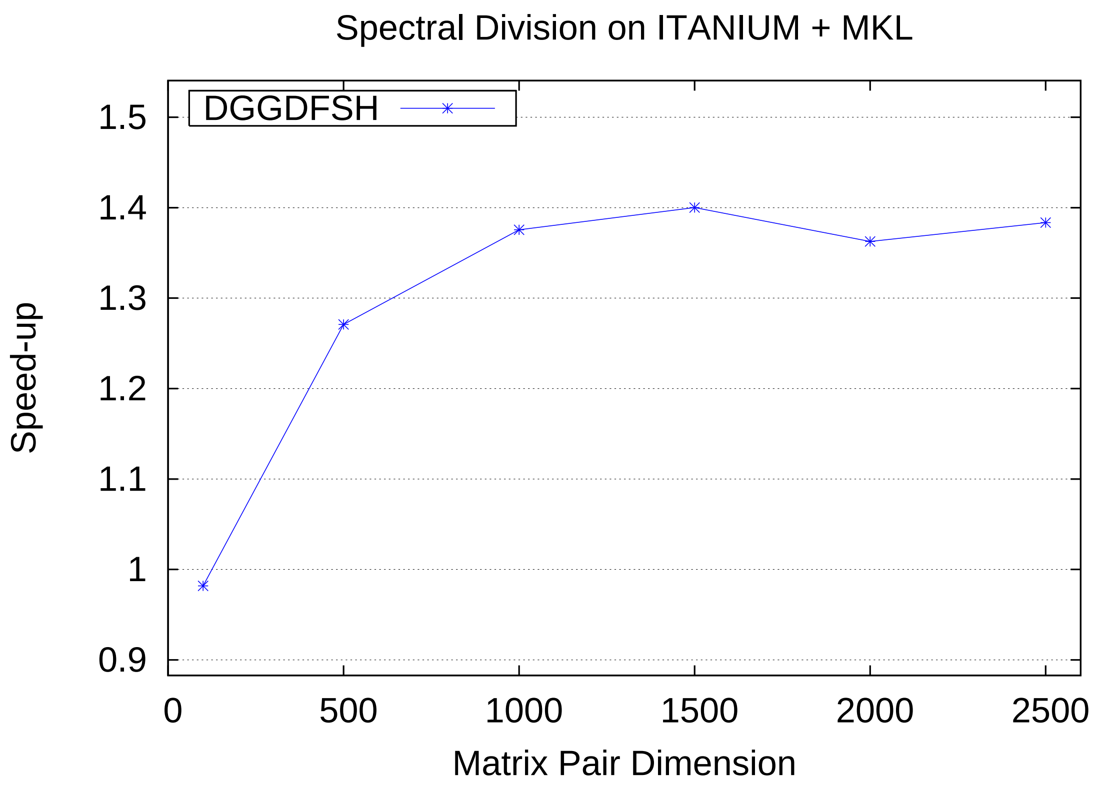
Experiments

MFLOPs on ITANIUM



Experiments

Speed-up of DGGDFSH over DGGDFSP on ITANIUM



Experiments

Comparison with the approach based on reduction to generalized Real Schur form + reordering procedure

#Iter. for DGGDFSH to result in (roughly) the same execution time as DGGDFSX

n	PENTIUM	ITANIUM
100	27	32
500	30	35
1000	27	32
2000	27	30

Conclusions

- Two variants for the inverse-free iteration that reduce the theoretical cost per iteration of the traditional implementation by 32%
- Proceeding by blocks, one of the variants allows the use of BLAS-3 and reduces the practical cost per iteration by 20–40% starting from moderate size matrices ($n \approx 500$)
- Narrows the gap between the costs of the inverse-free iteration and other spectral division tools
- Parallelization on SMP and multicore with high efficiency is direct: use multithreaded BLAS
- Parallelization following the message-passing paradigm is also easy and performance will probably be high too