GAMM annual meeting
Section 17
Erlangen March 12, 2014

A Spectral Divide-and-Conquer Approach for the Non-Symmetric Generalized Eigenvalue Problem

Peter Benner   Martin Köhler   Jens Saak

Computational Methods in Systems and Control Theory
Max Planck Institute for Dynamics of Complex Technical Systems

MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

Motivation

Non-Symmetric Generalized Eigenvalue Problem

We consider the non-symmetric generalized eigenvalue problem:

$$Ax = \lambda Bx,$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$ are non-singular matrices and $\lambda \in \mathbb{C}$ is an eigenvalue with its eigenvector $x \in \mathbb{R}^n$.

Motivation

Non-Symmetric Generalized Eigenvalue Problem

We consider the non-symmetric generalized eigenvalue problem:

$$Ax = \lambda Bx,$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$ are non-singular matrices and $\lambda \in \mathbb{C}$ is an eigenvalue with its eigenvector $x \in \mathbb{R}^n$.

Key idea behind the solution:

Compute the *generalized Schur decomposition*:

$$\underbrace{Q^H A Z}_{S} y = \lambda \underbrace{Q^H B Z}_{T} y,$$

where $S \in \mathbb{C}^{n \times n}$ and $T \in \mathbb{C}^{n \times n}$ are upper triangular and $Q \in \mathbb{C}^{n \times n}$ and $Z \in \mathbb{C}^{n \times n}$ are unitary matrices. [Stewart '72]

Motivation

Non-Symmetric Generalized Eigenvalue Problem

We consider the non-symmetric generalized eigenvalue problem:

$$Ax = \lambda Bx,$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$ are non-singular matrices and $\lambda \in \mathbb{C}$ is an eigenvalue with its eigenvector $x \in \mathbb{R}^n$.

Key idea behind the solution:

Compute the *generalized Schur decomposition*:

$$\underbrace{Q^T AZ}_{S} y = \lambda \underbrace{Q^T BZ}_{T} y,$$

where $S \in \mathbb{R}^{n \times n}$ and $T \in \mathbb{R}^{n \times n}$ are quasi upper triangular and $Q \in \mathbb{R}^{n \times n}$ and $Z \in \mathbb{R}^{n \times n}$ are orthogonal matrices. [Stewart '72]

Motivation

QZ Algorithm

[MOLER, STEWART '73]

Common way to compute the generalized Schur decomposition:

QZ Algorithm

1. Compute $\tilde{B} = QB$ using the QR decomposition and transform A into $\tilde{A} = Q^H A$.
2. Reduce the pair $(\tilde{A}, \tilde{B})$ to Hessenberg-Triangular form using Givens-Rotations.
3. Apply QZ steps to $(\tilde{A}, \tilde{B})$ until the matrix $\tilde{A}$ has reduced Hessenberg form. $\rightarrow$ generalized Schur form.

Motivation

QZ Algorithm

[Moler, Stewart '73]

Common way to compute the generalized Schur decomposition:

QZ Algorithm

1. Compute $\tilde{B} = QB$ using the QR decomposition and transform A into $\tilde{A} = Q^H A$.
2. Reduce the pair $(\tilde{A}, \tilde{B})$ to Hessenberg-Triangular form using Givens-Rotations.
3. Apply QZ steps to $(\tilde{A}, \tilde{B})$ until the matrix $\tilde{A}$ has reduced Hessenberg form. $\rightarrow$ generalized Schur form.

Givens-Rotations perform badly on modern computer architectures.
$\rightarrow$ Multicore features not usable. ☹

Motivation

QZ Algorithm

[Moler, Stewart '73]

Common way to compute the generalized Schur decomposition:

QZ Algorithm

1. Compute $\tilde{B} = QB$ using the QR decomposition and transform A into $\tilde{A} = Q^H A$.
2. Reduce the pair $(\tilde{A}, \tilde{B})$ to Hessenberg-Triangular form using Givens-Rotations.
3. Apply QZ steps to $(\tilde{A}, \tilde{B})$ until the matrix $\tilde{A}$ has reduced Hessenberg form. $\rightarrow$ generalized Schur form.

$\rightarrow$ Implemented in LAPACK as `DGGES`.

$\rightarrow$ [Adlerborn, Kågström,Kressner '14]: distributed parallel implementation.

Spectral Division and the Sign Function

Spectral Division

From the block generalized Schur form:

$$\underbrace{\begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix}}_{Q^T} A \underbrace{\begin{pmatrix} Z_1 & Z_2 \end{pmatrix}}_{Z} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

and

$$\underbrace{\begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix}}_{Q^T} B \underbrace{\begin{pmatrix} Z_1 & Z_2 \end{pmatrix}}_{Z} = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix},$$

we get two independent eigenvalue problems (A_{11}, B_{11}) and (A_{22}, B_{22}).

Our Aim: Split (A, B) such that $\Lambda(A_{11}, B_{11}) \subset \mathbb{C}_-$ and $\Lambda(A_{22}, B_{22}) \subset \mathbb{C}_+$.

Spectral Division and the Sign Function

Spectral Division

From the block generalized Schur form:

$$\boxed{\underbrace{\begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix}}_{Q^T}} A \boxed{\underbrace{\begin{pmatrix} Z_1 & Z_2 \end{pmatrix}}_{Z}} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

and

$$\boxed{\underbrace{\begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix}}_{Q^T}} B \boxed{\underbrace{\begin{pmatrix} Z_1 & Z_2 \end{pmatrix}}_{Z}} = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}$$

Problem:

Find a way to compute two orthogonal matrices $Q = [Q_1, Q_2]$ and $Z = [Z_1, Z_2]$ by using parallely scaling level-3 BLAS operations.

we get two independent eigenvalue problems (A_{11}, B_{11}) and (A_{22}, B_{22}).

Our Aim: Split (A, B) such that $\Lambda(A_{11}, B_{11}) \subset \mathbb{C}_-$ and $\Lambda(A_{22}, B_{22}) \subset \mathbb{C}_+$.

Spectral Division and the Sign Function

Generalized Sign Function

Generalized Matrix Sign Function [Gardiner, Laub '86]

Let (A, B) be a matrix pencil with no purely imaginary eigenvalue, than we define

$$\operatorname{sign}(A, B) := B \operatorname{sign}\left(B^{-1}A\right)$$

as the sign of the pencil where $\operatorname{sign}\left(B^{-1}A\right)$ is the sign of the matrix $B^{-1}A$.

Useful properties:

- $\operatorname{Range}(B + \operatorname{sign}(A, B))$ is the right deflating subspace corresponding to all eigenvalues with positive real part.
- $\operatorname{Range}(B - \operatorname{sign}(A, B))$ is the right deflating subspace corresponding to all eigenvalues with negative real part.
- $(B^{-1}\operatorname{sign}(A, B))^2 = I$.

Spectral Division and the Sign Function

Generalized Sign Function

[Gardiner, Laub '86]

From $(B^{-1}\operatorname{sign}(A, B))^2 = I$ follows the Newton iteration:

$$A_0 \leftarrow A, \quad A_{k+1} \leftarrow \frac{1}{2}\left(A_k + \quad BA_k^{-1}B\right), \quad k = 0, 1, 2, \ldots$$

to compute $\operatorname{sign}(A, B)$.

Spectral Division and the Sign Function

Generalized Sign Function

[Gardiner, Laub '86]

From $(B^{-1}\operatorname{sign}(A,B))^2 = I$ follows the Newton iteration:

$$A_0 \leftarrow A, \quad A_{k+1} \leftarrow \frac{1}{2c_k}\left(A_k + c_k^2 B A_k^{-1} B\right), \quad k = 0, 1, 2, \dots$$

to compute $\operatorname{sign}(A,B)$. c_k is a additional scaling factor.

Example: determinantal scaling: $c_k = \left(\frac{|\det(A_k)|}{|\det(B)|}\right)^{\frac{1}{n}}$.

Spectral Division and the Sign Function

Generalized Sign Function

[Gardiner, Laub '86]

From $(B^{-1}\text{sign}\,(A, B))^2 = I$ follows the Newton iteration:

$$A_0 \leftarrow A, \quad A_{k+1} \leftarrow \frac{1}{2c_k}\left(A_k + c_k^2 B A_k^{-1} B\right), \quad k = 0, 1, 2, \ldots$$

to compute $\text{sign}\,(A, B)$. c_k is a additional scaling factor.

Example: determinantal scaling: $c_k = \left(\frac{|\det(A_k)|}{|\det(B)|}\right)^{\frac{1}{n}}$.

Observations

- The generalized sign function iteration employs only level-3 routines: `DGETRF`, `DGETRS`, and `DGEMM`.
- The matrix $Z = [Z_1, Z_2]$ can be constructed using the range properties.

Spectral Division and the Sign Function

Spectral Division using the Sign Function

[Sun, Quintana-Ortí '04]

Questions:

1. How to contruct Z using level-3 operations in a robust way?
2. How to compute the corresponding Q?

Spectral Division and the Sign Function

Spectral Division using the Sign Function

[Sun, Quintana-Ortí '04]

Questions:

1. How to contruct Z using level-3 operations in a robust way?
2. How to compute the corresponding Q?

Computation of Z: From the range properties follows:

$$(B + \operatorname{sign}(A, B))Z_1 = 0 \quad \text{and} \quad (B + \operatorname{sign}(A, B))Z_2 = K.$$

Computation of Q:

- Q_1 lies in the range of $AZ_1 + BZ_1$,
- Q_2 is complementary orthogonal to $AZ_1 + BZ_1$.

Spectral Division and the Sign Function

Spectral Division using the Sign Function

[Sun, Quintana-Ortí '04]

Questions:

1. How to contruct Z using level-3 operations in a robust way?
2. How to compute the corresponding Q?

Computation of Z: From the range properties follows:

$$(B + \operatorname{sign}(A, B))^T = [Z_2, Z_1] \begin{pmatrix} K \\ 0 \end{pmatrix}.$$

Computation of Q:

$$[AZ_1, BZ_1] = [Q_1, Q_2] \begin{pmatrix} M \\ 0 \end{pmatrix}.$$

Spectral Division and the Sign Function

Spectral Division using the Sign Function

[Sun, Quintana-Ortí '04]

Questions:

1. How to compute Z using level-3 operations in a robust way?
2. How to compute the corresponding Q?

We can compute Q and Z from $\operatorname{sign}(A, B)$ using two RRQR procedures.
$\rightarrow$ use level-3 subroutine `DGEQP3` from LAPACK.

Computation of Z: From the range properties follows:

$$(B + \operatorname{sign}(A, B))^T = [Z_2, Z_1] \begin{pmatrix} K \\ 0 \end{pmatrix}.$$

Computation of Q:

$$[AZ_1, BZ_1] = [Q_1, Q_2] \begin{pmatrix} M \\ 0 \end{pmatrix}.$$

The Divide, Shift and Conquer Algorithm

Recursive Spectral Division

We get **two independent** eigenvalue problems for (A_{11}, B_{11}) and (A_{22}, B_{22}) from the spectral division.

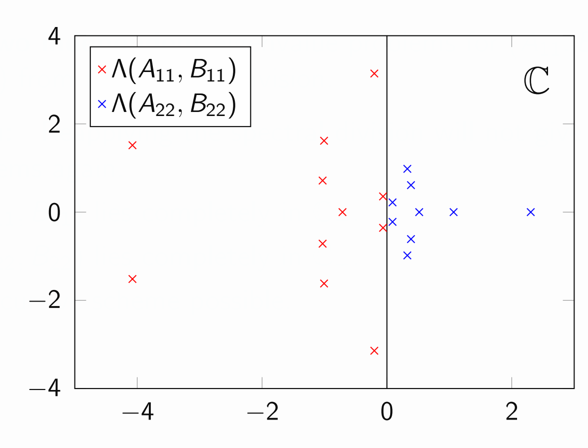
Problem: Reapplying the spectral division will not give smaller subproblems again.

- $\Lambda(A_{11}, B_{11})$ lies completely in $\mathbb{C}_-$.
- $\Lambda(A_{22}, B_{22})$ lies completely in $\mathbb{C}_+$.

$\rightarrow$ No recursive scheme possible.

The Divide, Shift and Conquer Algorithm

Original Spectrum:



The Divide, Shift and Conquer Algorithm

Recursive Spectral Division

We get **two independent** eigenvalue problems for (A_{11}, B_{11}) and (A_{22}, B_{22}) from the spectral division.

Problem: Reapplying the spectral division will not give smaller subproblems again.

- $\Lambda(A_{11}, B_{11})$ lies completely in $\mathbb{C}_-$.
- $\Lambda(A_{22}, B_{22})$ lies completely in $\mathbb{C}_+$.

$\rightarrow$ No recursive scheme possible.

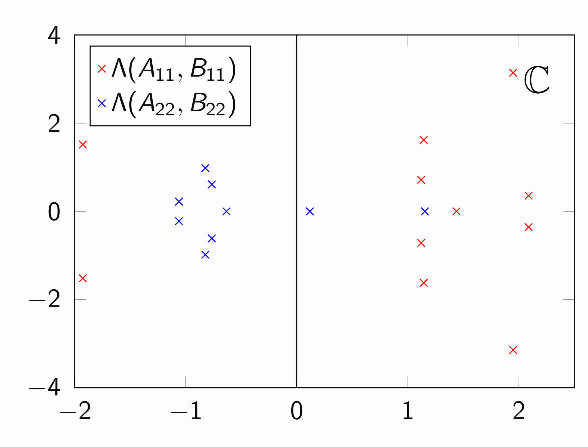
Idea

Shift $\Lambda(A_{11}, B_{11})$ by θ_- to the right and $\Lambda(A_{22}, B_{22})$ by θ_+ to the left to get two new spectra which enclose the imaginary axis.

The Divide, Shift and Conquer Algorithm

Recurs

Shifted Spectrum:



W ... B_{11}) and (A_{22}, B_{22} ...

Problem ... smaller subproble...

- $\Lambda(A_1$...
- $\Lambda(A_2$...

$\longrightarrow$ No re...

Idea

Shift $\Lambda(A$ ... $_+$ to the left to get two new spectra which enclose the imaginary axis.

The Divide, Shift and Conquer Algorithm

Optimal Shift Parameter Approximation

Optimal Choice of θ_*: Choose θ_- or θ_+ respectively such that the problems emerging out of $(\tilde{A}_{11}, B_{11})$ and $(\tilde{A}_{22}, B_{22})$ after the spectral division are equally sized. $\rightarrow$ Problem nearly solved.

The Divide, Shift and Conquer Algorithm

Optimal Shift Parameter Approximation

Optimal Choice of θ_*: Choose θ_- or θ_+ respectively such that the problems emerging out of $(\tilde{A}_{11}, B_{11})$ and $(\tilde{A}_{22}, B_{22})$ after the spectral division are equally sized. $\rightarrow$ Problem nearly solved.

If the real parts of the eigenvalues are equally distributed, the optimal θ_- is obviously given by

$$\theta_- := \frac{1}{2}\Re(\lambda_{\mathrm{left}}),$$

where λ_{left} is the left-most eigenvalue of (A_{11}, B_{11}).

The Divide, Shift and Conquer Algorithm

Optimal Shift Parameter Approximation

Optimal Choice of θ_*: Choose θ_- or θ_+ respectively such that the problems emerging out of $(\tilde{A}_{11}, B_{11})$ and $(\tilde{A}_{22}, B_{22})$ after the spectral division are equally sized. $\rightarrow$ Problem nearly solved.

If the real parts of the eigenvalues are equally distributed, the optimal θ_- is obviously given by

$$\theta_- := \frac{1}{2}\Re(\lambda_{\text{left}}),$$

where λ_{left} is the left-most eigenvalue of (A_{11}, B_{11}).

Cheap approximation of $\Re(\lambda_{\text{left}})$:

$$-\Re(\lambda_{\text{left}}) \ \leq \ \rho(A_{11}, B_{11}) \leq \ \|B_{11}^{-1}A_{11}\|_2 \leq \ \|B_{11}^{-1}A_{11}\|_F,$$

where $\rho(A_{11}, B_{11})$ is the spectral radius of of (A_{11}, B_{11}).

The Divide, Shift and Conquer Algorithm

The Algorithm

Combining the spectral division and the shift parameter computation gives the following recursive scheme:

The Divide, Shift and Conquer Algorithm

The Algorithm

Combining the spectral division and the shift parameter computation gives the following recursive scheme:

Algorithm 1 `[Q,Z] = dscqz(A,B)`

Input: $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$ non-singular, $\Lambda(A,B) \cap \imath\mathbb{R} = \{\}$
Output: $(Q^T AZ, Q^T BZ)$ in real Schur form.

1: **if** (A, B) is trivial to solve **then**
2: Compute Q, Z directly and return them.
3: **end if**
4: Compute Q and Z using Algorithm 1 and transform (A, B).
5: Set $\theta_- = -\frac{1}{2}\|B_{11}^{-1}A_{11}\|_F$ and $\theta_+ = \frac{1}{2}\|B_{22}^{-1}A_{22}\|_F$.
6: $[\tilde{Q}_1, \tilde{Z}_1]$=`dscqz(`$A_{11} - \theta_- B_{11}$, B_{11}`)`.
7: $[\tilde{Q}_2, \tilde{Z}_2]$=`dscqz(`$A_{22} - \theta_+ B_{22}$, B_{22}`)`.
8: Update $Q := Q \begin{pmatrix} \tilde{Q}_1 & 0 \\ 0 & \tilde{Q}_2 \end{pmatrix}$ and $Z := Z \begin{pmatrix} \tilde{Z}_1 & 0 \\ 0 & \tilde{Z}_2 \end{pmatrix}$.
9: **return** $[Q, Z]$

The Divide, Shift and Conquer Algorithm

The Algorithm

Combining the spectral division and the shift parameter computation gives the following recursive scheme:

Algorithm 1 `[Q,Z] = dscqz(A,B)`

Input: $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$ non-singular, $\Lambda(A,B) \cap \imath\mathbb{R} = \{\}$

Output: (Q^TAZ, Q^TBZ) in real Schur form.

if (A,B) is $\boxed{\text{trivial to solve}}$ **then**
    Compute Q, Z directly and return them.
end if
Compute Q and Z using Algorithm 1 and transform (A,B).
Set $\theta_- = -\frac{1}{2}\|B_{11}^{-1}A_{11}\|_F$ and $\theta_+ = \frac{1}{2}\|B_{22}^{-1}A_{22}\|_F$.
$[\tilde{Q}_1, \tilde{Z}_1]$=`dscqz(`$A_{11} - \theta_- B_{11}$`,`$B_{11}$`)`.
$[\tilde{Q}_2, \tilde{Z}_2]$=`dscqz(`$A_{22} - \theta_+ B_{22}$`,`$B_{22}$`)`.
Update $Q := Q\begin{pmatrix}\tilde{Q}_1 & 0\\ 0 & \tilde{Q}_2\end{pmatrix}$ and $Z := Z\begin{pmatrix}\tilde{Z}_1 & 0\\ 0 & \tilde{Z}_2\end{pmatrix}$.
return `[`Q`,`Z`]`

Trivial: The Schur form can be computed directly, i.e. the problem is of size 1×1 or 2×2.

The Divide, Shift and Conquer Algorithm

Implementation Details

- The evaluation of $\theta_- = -\frac{1}{2}\|B_{11}^{-1}A_{11}\|_F$ and $\theta_+ = \frac{1}{2}\|B_{22}^{-1}A_{22}\|_F$ is only necessary after the first step.

The spectral radius can not increase during the recursion.
$\rightarrow$ We pass $|\theta_-|$ and $|\theta_+|$ as spectral radius θ to the next step and use

$$\theta_- := -\frac{1}{2}\theta \quad \text{and} \quad \theta_+ := \frac{1}{2}\theta$$

as new parameters in the next step.
$\rightarrow$ We can guarantee $\theta_* \rightarrow 0$ during the recursion.

The Divide, Shift and Conquer Algorithm

Implementation Details

- The evaluation of $\theta_- = -\frac{1}{2}\|B_{11}^{-1}A_{11}\|_F$ and $\theta_+ = \frac{1}{2}\|B_{22}^{-1}A_{22}\|_F$ is only necessary after the first step.
- Reformulate the recursion as an iterative scheme.

The Divide, Shift and Conquer Algorithm

Implementation Details

- The evaluation of $\theta_- = -\frac{1}{2}\|B_{11}^{-1}A_{11}\|_F$ and $\theta_+ = \frac{1}{2}\|B_{22}^{-1}A_{22}\|_F$ is only necessary after the first step.
- Reformulate the recursion as an iterative scheme.
- Stop the recursion if the remaining eigenvalue problem is trivial to solve, i.e. it can be solved inside the cache of a single CPU-core by DGGES.

The trivial size n_{triv} given by:

$$n_{\text{triv}} \leq -\frac{11}{8} + \sqrt{-\frac{135}{64} + \frac{C}{4}} \approx \sqrt{\frac{C}{4}},$$

where C is the cache size counted in floating point numbers of the desired precision.

The Divide, Shift and Conquer Algorithm

Implementation Details

- The evaluation of $\theta_{-} = -\frac{1}{2}\|B_{11}^{-1}A_{11}\|_F$ and $\theta_{+} = \frac{1}{2}\|B_{22}^{-1}A_{22}\|_F$ is only necessary after the first step.
- Reformulate the recursion as an iterative scheme.
- Stop the recursion if the remaining eigenvalue problem is trivial to solve, i.e. it can be solved inside the cache of a single CPU-core by DGGES.
- Use a multi-threaded BLAS for the divide and conquer phase and a single threaded BLAS to solve the trivial problems in parallel.

Numerical Results

Test hardware:

	Workstation Xeon E3-1245	Compue-Server Xeon E5-2690
CPU:	Xeon E3-1245 @ 3.3GHz	Dual Xeon E5-2690 @ 2.9 GHz
Cores:	4	16 (2×8)
L2 Cache:	256KiB per core	256KiB per core
n_{triv}	90	90
RAM:	8 GiB DDR3	32 GiB DDR3
OS:	Ubuntu 12.04	Ubuntu 12.04
Compiler:	GCC 4.6.3	GCC 4.6.3
BLAS:	Intel MKL 10.2	Intel MKL 10.2

Test matrices from MatrixMarket and the Oberwolfach Collection:

	Name	Dimension		Name	Dimension
(a)	rbs480	480	(b)	bsst09	1 083
(c)	spiral inductor	1434	(d)	bcsst11	1 473
(e)	filter2D	1 668	(f)	bcsst21	3 600
(g)	steel profile	5 177	(h)	steel profile	20 209

Numerical Results

Runtime and Speedup

Matrix	Xeon E3-1245		Dual Xeon E5-2690 - MKL 10.2			
	QZ	4 Thr.	QZ	1 Thr.	16 Thr.	speedup
(a)	1.31s	0.59s	1.75s	1.16s	0.51s	3.57
(b)	17.27s	10.48s	18.99s	22.68s	6.29s	3.02
(c)	40.16s	15.05s	39.86s	32.47s	8.16s	4.88
(d)	46.77s	43.09s	64.38s	86.90s	25.69s	2.51
(e)	77.35s	28.38s	80.40s	67.40s	14.41s	4.68
(f)	616.05s	526.22s	740.78s	1189.69s	383.08s	1.93
(g)	3 046.40s	1 006.25s	3 286.61s	2 684.74s	598.35s	5.49
(h)	out of	memory	255 057.00s	207 198.00s	38 200.00s	6.68

Numerical Results

Runtime and Speedup

Matrix	Xeon E3-1245		Dual Xeon E5-2690 - MKL 10.2			
	QZ	4 Thr.	QZ	1 Thr.	16 Thr.	speedup
(a)	1.31s	0.59s	1.75s	1.16s	0.51s	3.57
(b)	17.27s	10.48s	18.99s	22.68s	6.29s	3.02
(c)	40.16s	15.05s	39.86s	32.47s	8.16s	4.88
(d)	46.77s	43.09s	64.38s	86.90s	25.69s	2.51
(e)	77.35s	28.38s	80.40s	67.40s	14.41s	4.68
(f)	616.05s	526.22s	740.78s	1189.69s	383.08s	1.93
(g)	3 046.40s	1 006.25s	3 286.61s	2 684.74s	598.35s	5.49
(h)	out of	memory	255 057.00s	207 198.00s	38 200.00s	6.68

Reduce the runtime from $\approx$ **3 days** to $\approx$ **10.6 hours.**

Power Consumption:

QZ: 16.20KWh

DSCQZ: 4.24KWh $\rightarrow$ **save 74% energy!** ☺

Numerical Results

Accuracy

Assuming that QZ gives the correct result, we define a global (average) error:

$$err_{global}(A, B) := \frac{\|\Lambda^{QZ}(A, B) - \Lambda^{DSCQZ}(A, B)\|_2}{\|\Lambda^{QZ}(A, B)\|_2}$$

and a local (point wise) error:

$$err_{local}(A, B) := \max_{i=1,\dots,n} \frac{|\lambda_i^{QZ}(A, B) - \lambda_i^{DSCQZ}(A, B)|}{|\lambda_i^{QZ}(A, B)|}$$

for the eigenvalues of (A, B).

Numerical Results

Accuracy

Assuming that QZ gives the correct result, we define a global (average) error:

$$err_{global}(A, B) := \frac{\|\Lambda^{QZ}(A, B) - \Lambda^{DSCQZ}(A, B)\|_2}{\|\Lambda^{QZ}(A, B)\|_2}$$

and a local (point wise) error:

$$err_{local}(A, B) := \max_{i=1,\ldots,n} \frac{|\lambda_i^{QZ}(A, B) - \lambda_i^{DSCQZ}(A, B)|}{|\lambda_i^{QZ}(A, B)|}$$

for the eigenvalues of (A, B).

Matrix	$err_{global}(A, B)$	$err_{local}(A, B)$	Matrix	$err_{global}(A, B)$	$err_{local}(A, B)$
(a)	3.10 e-10	3.15 e-10	(e)	7.60 e-15	5.32 e-11
(b)	4.63 e-13	4.40 e-11	(f)	6.17 e-15	1.72 e-10
(c)	1.39 e-14	3.77 e-12	(g)	1.71 e-14	1.06 e-10
(d)	4.62 e-15	9.44 e-09	(h)	5.21 e-14	1.02 e-09

Conclusions

We have seen that:

- We can formulate a level-3 BLAS based solver for the NGEP.
- The new solver scales on multicore architectures.
- The level-3 BLAS operations make extensive use of the vector registers. ($\rightarrow$ see 1 thread results)
- We get an acceptable approximation of the NGEP solution in drastically reduced time.

Conclusions

We have seen that:

- We can formulate a level-3 BLAS based solver for the NGEP.
- The new solver scales on multicore architectures.
- The level-3 BLAS operations make extensive use of the vector registers. ($\rightarrow$ see 1 thread results)
- We get an acceptable approximation of the NGEP solution in drastically reduced time.

Further Research:

- Include more parallelism from the recursive structure
  $\rightarrow$ use properties of NUMA architectures to share the work.
- Develop a hybrid CPU/Accelerator implementation.
- Improve robustness
  $\rightarrow$ develop fall back situations if the DSCQZ algorithm fails.

Conclusions

We have seen that:

- We can formulate a level-3 BLAS based solver for the NGEP.
- The new solver scales on multicore architectures.
- The level-3 BLAS operations make extensive use of the vector registers. ($\rightarrow$ see 1 thread results)
- We get an acceptable approximation of the NGEP solution in drastically reduced time.

Further Research:

- Include more parallelism from the recursive structure
  $\rightarrow$ use properties of NUMA architectures to share the work.
- Develop a hybrid CPU/Accelerator implementation.
- Improve robustness
  $\rightarrow$ develop fall back situations if the DSCQZ algorithm fails.

Thank you for your attention!

Questions?