Reduction to Condensed Forms for Symmetric Eigenvalue Problems on Multi-core Architectures

Paolo Bientinesi[1] Francisco Igual[2]
Daniel Kressner[3] Enrique S. Quintana-Ortí[2]

[1]AICES, RWTH Aachen University
Aachen (Germany)

[2]Departamento de Ingeniería y Ciencia de los Computadores.
University Jaume I
Castellón (Spain)

[3]Seminar für angewandte Mathematik, ETH Zürich (Switzerland)

UNIVERSITAT JAUME•I

Motivation

The symmetric eigenvalue problem

$$AX = X\Lambda,$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathbb{R}^{n\times n}$ are the eigenvalues and $X \in \mathbb{R}^{n\times n}$ are the eigenvectors

Applications (large scale dense A)

- computational quantum chemistry
- finite element modeling
- multivariate statistics
- density functional theory

Motivation

Efficient algorithms for dense eigenproblems

1. $Q_1^T A Q_1 \rightarrow T$, with $Q_1 \in \mathbb{R}^{n\times n}$ orthogonal, $T \in \mathbb{R}^{n\times n}$ tridiagonal
2. $Q_2^T T Q_2 \rightarrow \Lambda$, with $Q_2 \in \mathbb{R}^{n\times n}$ orthogonal, $\Lambda \in \mathbb{R}^{n\times n}$
3. If the eigenvectors of A needed, apply a back-transformation to the eigenvectors of T

Stage	Algorithm	Cost (flops)
1	Two-sided reduction	$O(n^3)$
2	MR^3	$O(n^2)$
3	Back-transform	$O(n^3)$

Outline

Contents

Two-sided reduction: LAPACK SYTRD

$$H_{j-1}^T \cdots H_2^T H_1^T A H_1 H_2 \cdots H_{j-1} = \left(\begin{array}{c|c|c} T_{00} & T_{10}^T & 0 \\ \hline T_{10} & A_{11} & A_{21}^T \\ \hline 0 & A_{21} & A_{22} \end{array}\right),$$

where $T_{00} \in \mathbb{R}^{j-1 \times j-1}$ is in tridiagonal form and $A_{11} \in \mathbb{R}^{b \times b}$.

Current iteration of SYTRD (Step 1)

$\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right)$ reduced to tridiagonal form + build $U, W \in \mathbb{R}^{(n-j-b+1) \times b}$:

$$H_{j+b-1}^T \cdots H_{j+1}^T H_j^T \left(\begin{array}{c|c|c} T_{00} & T_{10}^T & 0 \\ \hline T_{10} & A_{11} & A_{21}^T \\ \hline 0 & A_{21} & A_{22} \end{array}\right) H_j H_{j+1} \cdots H_{j+b-1}$$
$$= \left(\begin{array}{c|c|c} T_{00} & T_{10}^T & 0 \\ \hline T_{10} & T_{11} & T_{21}^T \\ \hline 0 & T_{21} & A_{22} - UW^T - WU^T \end{array}\right),$$

T_{11} tridiagonal and all entries of T_{21} zero but top right corner.

Two-sided reduction: LAPACK SYTRD

$$H_{j-1}^T \cdots H_2^T H_1^T A H_1 H_2 \cdots H_{j-1} = \left(\begin{array}{c|c|c} T_{00} & T_{10}^T & 0 \\ \hline T_{10} & A_{11} & A_{21}^T \\ \hline 0 & A_{21} & A_{22} \end{array} \right),$$

where $T_{00} \in \mathbb{R}^{j-1 \times j-1}$ is in tridiagonal form and $A_{11} \in \mathbb{R}^{b \times b}$.

Current iteration of SYTRD (Step 2)

A_{22} is updated as $A_{22} := A_{22} - UW^T - WU^T$, only the lower (or the upper) half of this matrix is updated.

Cost analysis of LAPACK SYTRD

Step 1
- Four panel-vector multiplications
- One symmetric matrix-vector multiplication with A_{22}

$2(n-j)^2 b$ flops

Step 2
- Update A_{22}: computed with SYR2K

$2(n-j)^2 b$ flops

Overall cost

$4n^3/3$ flops, if $b \ll n$

Contents

The SBR Toolbox

The SBR Toolbox

- SBR: symmetric band reduction via orthogonal transforms
- Routines for:
  - Reduction of dense symmetric matrices to banded form (SYRDB)
  - Reduction of banded matrices to narrower banded form (SBRDB)
  - Reduction to tridiagonal form (SBRDT)

SBR vs. LAPACK

- SYRDB + SBRDT $\Rightarrow$ Dense matrix to tridiagonal form
- Same result as LAPACK SYTRD

The SBR Toolbox

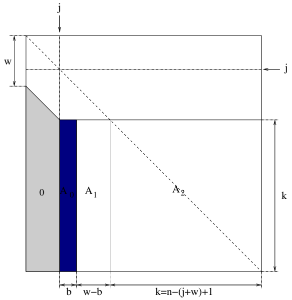
The SBR Toolbox

- SBR: symmetric band reduction via orthogonal transforms
- Routines for:
  - Reduction of dense symmetric matrices to banded form (SYRDB)
  - Reduction of banded matrices to narrower banded form (SBRDB)
  - Reduction to tridiagonal form (SBRDT)

SBR vs. LAPACK

- SYRDB + SBRDT $\Rightarrow$ Dense matrix to tridiagonal form
- Same result as LAPACK SYTRD

Two-sided reduction: SBR SYRDB



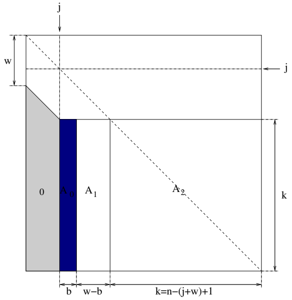
Step 1

Compute the QR factorization of $A_0 \in \mathbb{R}^{k \times b}$, $k = n-(j+w)+1$:

$$A_0 = Q_0 R_0, \tag{1}$$

Cost: $2b^2(k-b/3)$ flops.

Two-sided reduction: SBR SYRDB

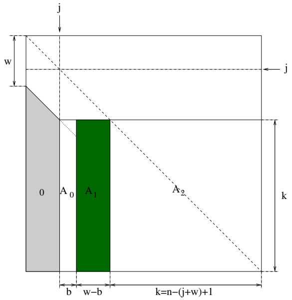


Step 2

Construct the factors of the compact WY representation of $Q_0 = I_k + WTW^T$, with $W \in \mathbb{R}^{k\times b}$ and $T \in \mathbb{R}^{k\times k}$ upper triangular.

Cost: kb^2 flops.

Two-sided reduction: SBR SYRDB



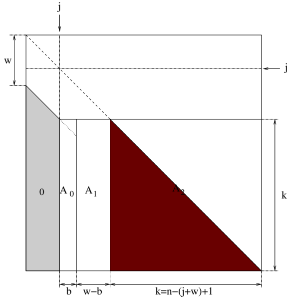
Step 3

Apply the orthogonal matrix to $A_1 \in \mathbb{R}^{k \times w-b}$ from the left:

$$A_1 := Q_0^T A_1 = (I_k + WTW^T)^T A_1 = A_1 + W(T(W^T A_1)). \tag{1}$$

Cost: $4kb(w-b)$ flops. (No operations if $w = b$)

Two-sided reduction: SBR SYRDB



Step 4

Apply orthogonal matrix to $A_2 \in \mathbb{R}^{k\times k}$ from left and right:

$$A_2 := Q_0^T A_2 Q_0 = (I_k + WY^T)^T A_2 (I + WY^T) \quad (1)$$

$$= A_2 + YW^T A_2 + A_2 WY^T + YW^T A_2 WY^T, \quad (2)$$

with $Y = WT$.

Cost analysis of Step 4

Step 4

$$A_2 := A_2 + YW^T A_2 + A_2 WY^T + YW^T A_2 WY^T \tag{3}$$

Computed as a sequence of BLAS operations:

$$\begin{array}{llll} (\text{SYMM}) & X_1 & := & A_2 W, \\ (\text{GEMM}) & X_2 & := & \frac{1}{2} X_1^T W, \\ (\text{GEMM}) & X_3 & := & X_1 + Y X_2, \\ (\text{SYR2K}) & A_2 & := & A_2 + X_3 Y^T + Y X_3^T. \end{array}$$

Total cost

$4k^2b + 4kb^2$ flops

Cost analysis of Step 4

Step 4

$$A_2 := A_2 + YW^TA_2 + A_2WY^T + YW^TA_2WY^T \tag{3}$$

Computed as a sequence of BLAS operations:

(SYMM)	X_1	$:=$	$A_2W,$	$2k^2b$ flops
(GEMM)	X_2	$:=$	$\frac{1}{2}X_1^TW,$	$2kb^2$ flops
(GEMM)	X_3	$:=$	$X_1 + YX_2,$	$2kb^2$ flops
(SYR2K)	A_2	$:=$	$A_2 + X_3Y^T + YX_3^T.$	$2k^2b$ flops

Total cost

$4k^2b + 4kb^2$ flops

Cost of full matrix to band form (SYRDB)

Step 1 $O(kb^2)$

Step 2 $O(kb^2)$

Step 3 $O(\max(kb^2, kbw))$

Step 4 $O(4k^2b + 4kb^2)$

Total $O(4n^3/3)$

Cost of reduction to tridiagonal form (SBRDT)

- Routine SBRDT in SBR (using Householder reflectors)
- SBRDT returns the tridiagonal matrix T
- T constructed one column at the time
  - BLAS-2 operations at best
- Total cost: $6n^2w + 8nw^2$ flops.

Cost of full matrix to band form (SYRDB)

Step 1 $O(kb^2)$

Step 2 $O(kb^2)$

Step 3 $O(\max(kb^2, kbw))$

Step 4 $O(4k^2b + 4kb^2)$

Total $O(4n^3/3)$

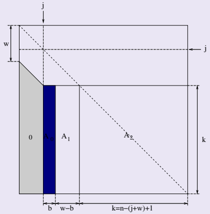
Cost of reduction to tridiagonal form (SBRDT)

- Routine SBRDT in SBR (using Householder reflectors)
- SBRDT returns the tridiagonal matrix T
- T constructed one column at the time
    - BLAS-2 operations at best
- Total cost: $6n^2w + 8nw^2$ flops.

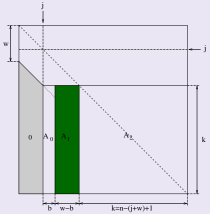
Contents

Reduction to band form on the GPU

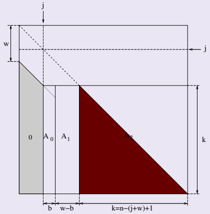
Hybrid strategy



Steps 1 and 2
Narrow M-V products
Better on CPU

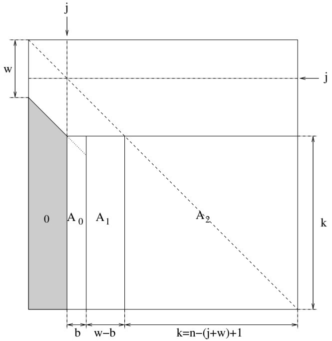


Step 3
$(w-b)$ usually small
Better on CPU



Step 4
Big BLAS-3 operations
Better on GPU

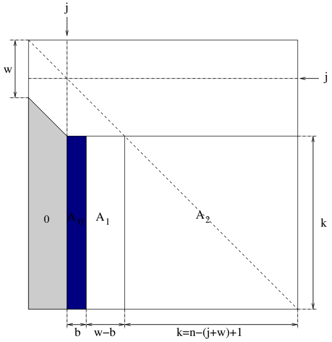
Reduction to band form on the GPU



Matrix in video memory

- Matrix in GPU memory
- For each column block:
  1. $A_0 \Rightarrow$ CPU
  2. Steps 1, 2 on CPU
  3. $A_1 \Rightarrow$ CPU
  4. Step 3 on CPU
  5. Transfer W, Y to GPU
  6. Step 4 on GPU
- Transfer $b \times b$ diagonal blocks to RAM

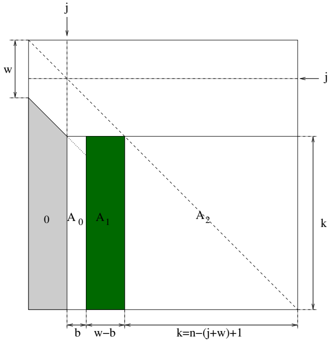
Reduction to band form on the GPU



Transfer A_0 to RAM
Steps 1 and 2 on CPU

- Matrix in GPU memory
- For each column block:
  1. $A_0 \Rightarrow$ CPU
  2. Steps 1, 2 on CPU
  3. $A_1 \Rightarrow$ CPU
  4. Step 3 on CPU
  5. Transfer W, Y to GPU
  6. Step 4 on GPU
- Transfer $b \times b$ diagonal blocks to RAM

Reduction to band form on the GPU

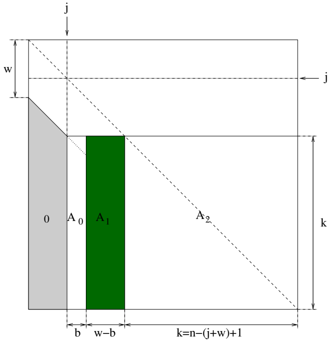


Transfer A_1 to RAM

Step 3 on CPU

- Matrix in GPU memory
- For each column block:
  1. $A_0 \Rightarrow$ CPU
  2. Steps 1, 2 on CPU
  3. $A_1 \Rightarrow$ CPU
  4. Step 3 on CPU
  5. Transfer W, Y to GPU
  6. Step 4 on GPU
- Transfer $b \times b$ diagonal blocks to RAM

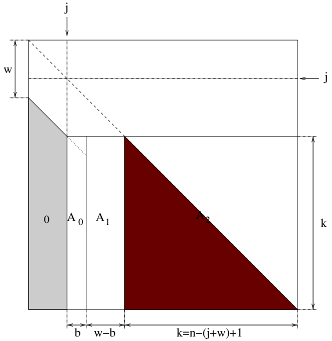
Reduction to band form on the GPU



Transfer W, Y to GPU

- Matrix in GPU memory
- For each column block:
  1. $A_0 \Rightarrow$ CPU
  2. Steps 1, 2 on CPU
  3. $A_1 \Rightarrow$ CPU
  4. Step 3 on CPU
  5. Transfer W, Y to GPU
  6. Step 4 on GPU
- Transfer $b \times b$ diagonal blocks to RAM

Reduction to band form on the GPU

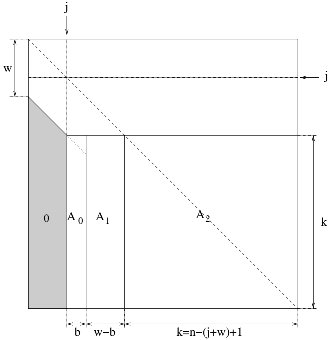


Transfer A_0 to RAM

Step 4 on GPU

- Matrix in GPU memory
- For each column block:
  1. $A_0 \Rightarrow$ CPU
  2. Steps 1, 2 on CPU
  3. $A_1 \Rightarrow$ CPU
  4. Step 3 on CPU
  5. Transfer W, Y to GPU
  6. Step 4 on GPU
- Transfer $b \times b$ diagonal blocks to RAM

Reduction to band form on the GPU

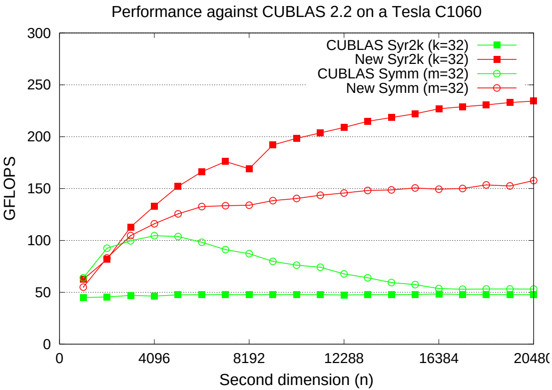


Band matrix and Householder reflectors on main memory

- Matrix in GPU memory
- For each column block:
  1. $A_0 \Rightarrow$ CPU
  2. Steps 1, 2 on CPU
  3. $A_1 \Rightarrow$ CPU
  4. Step 3 on CPU
  5. Transfer W, Y to GPU
  6. Step 4 on GPU
- Transfer $b \times b$ diagonal blocks to RAM

Accelerating the CUBLAS routines

- Bulk of the computation cast in terms of SYR2K and SYMM
- CUBLAS only offers tuned GEMM
- Solution: use our own tuned BLAS-3 implementation on GPU



[Level-3 BLAS on a GPU: Picking the Low Hanging Fruit]
FLAME Working Note #37, May 2009.

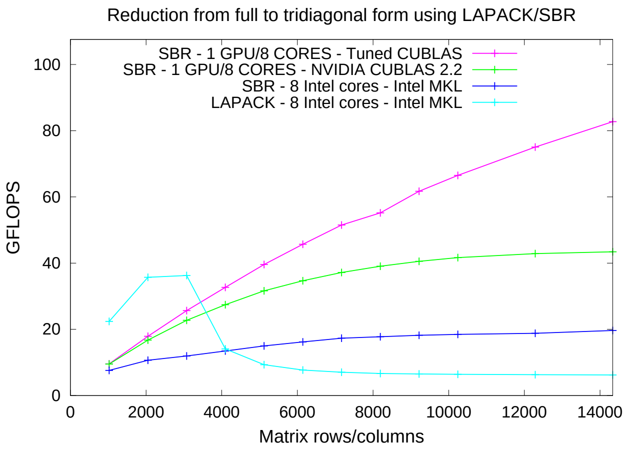
Experimental setup

Experimental setup

CPU	Dual Xeon QuadCore E5410
CPU frequency	2.33 Ghz
RAM memory	8 Gbytes
GPU	Tesla C1060
Processor	Nvidia GT200
GPU frequency	1.3 Ghz
Video memory	4 Gbytes DDR3
Interconnection	PCIExpress Gen2
CUDA (CUBLAS) version	2.2
MKL version	10.0.1
Driver version	185.18

- Reduction to tridiagonal form:
  - LAPACK SYTRD
  - SBR SYRDB + SBRDT
- Consider $4n^3/3$ flops for square matrices (order n). Single precision.

Experimental results



- Speedup $2.2x$ when using GPU acceleration for SBR (19.2 vs 42 GFLOPS)
- Speedup $4.3x$ when using GPU acceleration and tuned BLAS (19.2 vs 82 GFLOPS)

Execution times for SBR routines

		1st stage: Full→ Band				2nd stage: Band→ Tridiagonal
n	w	1 Core	4 Cores	8 Cores	CUBLAS	1 core
2048	32	1.1	0.8	0.8	0.2	0.4
	96	0.9	0.5	0.5	0.2	0.8
6144	32	33.5	23.8	28.5	2.5	3.7
	96	25.3	11.6	11.7	2.7	7.5
10240	32	155.8	110.4	129.5	10.1	10.3
	96	116.6	51.2	51.6	10.6	25.6

Table: Execution time (in seconds) for the two-stage SBR routines.

- Speedup $12x$ for the first stage with $w = 32$
- Times for the first and second stage are comparable if accelerated with GPU

Conclusions and future work

- Evaluation of the performance of existing codes for reductions from full to tridiagonal forms
- LAPACK vs. SBR
- Offloading of the most-expensive operations to the GPU
- Tuned BLAS-3 routines to boost performance
- Future work: single to double precision refinement

Conclusions and future work

Thank you!

More information...

[Reduction to condensed forms for symmetric eigenvalue problems on multi-core architectures]
Technical report 2009-13, Seminar for applied mathematics
ETH Zurich, March 2009

- HPCA Group at University Jaume I (http://www.hpca.uji.es)
- quintana@icc.uji.es, figual@icc.uji.es