High Performance Matrix Inversion of SPD Matrices on Graphics Processors

P. Benner[1], P. Ezzatti[2], E.S. Quintana-Ortí[3], **Alfredo Remón**[3]

[1]Max-Planck-Institute for Dynamics of Complex Technical Systems (Magdeburg, Germany).
[2]Centro de Cálculo-Inst. de la Computación,Univ. de la República (Montevideo, Uruguay).
[3]Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I (Castellón, Spain).

WEHA'11 - July 2011

Why matrix inversion?

- Matrix inversion requires an important computational effort
- Sometimes can be by-passed by solving systems of linear equations

  $\rightarrow$ But in some situations is necessary

- Examples include earth sciences and the matrix sign function method for expectral decomposition

Why matrix inversion?

- Matrix inversion requires an important computational effort
- Sometimes can be by-passed by solving systems of linear equations

$\rightarrow$ But in some situations is necessary

- Examples include earth sciences and the matrix sign function method for expectral decomposition

Why SPD matrices?

- In previous works we targeted the inversion of general matrices
- In this case the structure and properties of the matrix can be exploited, reporting important savings in terms of memory and computational time.

Outline

Algorithm 2 Matrix_inversion

1: Compute the Cholesky factorization $A = U^T U$, where $U \in \mathbb{R}^{n \times n}$ is upper triangular

2: Invert the triangular factor $U \to U^{-1}$

3: Obtain the inverse from the product $U^{-1} U^{-T} = A^{-1}$

Requires n^3 floating-point operations

Sweeps throught the matrix 3 times

Matrix inversion of an SPD matrix

The the Gauss-Jordan elimination algorithm

- In essence, it is a reordering of the operations
- Presents the same arithmetical cost

Implementation

- The algorithm sweeps through the matrix once
  $\rightarrow$ Less memory accesses
- Most of the computations are highly parallel
  $\rightarrow$ More parallelism

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 1

Algorithm: $[A] := \text{GJE}_{\text{BLK\_V1}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$W := -A_{00} \cdot A_{01}$	SYMM
$A_{11} := A_{11} + A_{01}^T \cdot A_{01}$	GEMM
$A_{11} := \text{CHOL}(A_{11})$	POTRF
$\text{TRIU}(A_{11}) := \text{TRIU}(A_{11})^{-1}$	TRTRI
$W := W \cdot A_{11}$	TRMM
$A_{01} := W \cdot A_{11}^T$	TRMM
$A_{00} := A_{00} + W \cdot W^T$	SYRK
$A_{11} := \text{TRIU}(A_{11}) \cdot \text{TRIU}(A_{11})^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 8 operations per iteration
- 6 of them are MM products

Limitations

- Data dependencies
- Except A_{00} all blocks are "small"

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 1

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_V1}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$W := -A_{00} \cdot A_{01}$	SYMM
$A_{11} := A_{11} + A_{01}^T \cdot A_{01}$	GEMM
$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11})^{-1}$	TRTRI
$W := W \cdot A_{11}$	TRMM
$A_{01} := W \cdot A_{11}^T$	TRMM
$A_{00} := A_{00} + W \cdot W^T$	SYRK
$A_{11} := \mathrm{TRIU}(A_{11}) \cdot \mathrm{TRIU}(A_{11})^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 8 operations per iteration
- 6 of them are MM products

Limitations

- Data dependencies
- Except A_{00} all blocks are "small"

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 1

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_v1}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$
where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**
Determine block size b
Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$W := -A_{00} \cdot A_{01}$	SYMM
$A_{11} := A_{11} + A_{01}^T \cdot A_{01}$	GEMM
$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11})^{-1}$	TRTRI
$W := W \cdot A_{11}$	TRMM
$A_{01} := W \cdot A_{11}^T$	TRMM
$A_{00} := A_{00} + W \cdot W^T$	SYRK
$A_{11} := \mathrm{TRIU}(A_{11}) \cdot \mathrm{TRIU}(A_{11})^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 8 operations per iteration
- 6 of them are MM products

Limitations

- Data dependencies
- Except A_{00} all blocks are "small"

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 1

Algorithm: $[A] := \text{GJE}_{\text{BLK\_v1}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$
where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**
Determine block size b
Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$W := -A_{00} \cdot A_{01}$	SYMM
$A_{11} := A_{11} + A_{01}^T \cdot A_{01}$	GEMM
$A_{11} := \text{CHOL}(A_{11})$	POTRF
$\text{TRIU}(A_{11}) := \text{TRIU}(A_{11})^{-1}$	TRTRI
$W := W \cdot A_{11}$	TRMM
$A_{01} := W \cdot A_{11}^T$	TRMM
$A_{00} := A_{00} + W \cdot W^T$	SYRK
$A_{11} := \text{TRIU}(A_{11}) \cdot \text{TRIU}(A_{11})^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 8 operations per iteration
- 6 of them are MM products

Limitations

- Data dependencies
- Except A_{00} all blocks are "small"

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 1

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_V1}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$W := -A_{00} \cdot A_{01}$	SYMM
$A_{11} := A_{11} + A_{01}^T \cdot A_{01}$	GEMM
$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11})^{-1}$	TRTRI
$W := W \cdot A_{11}$	TRMM
$A_{01} := W \cdot A_{11}^T$	TRMM
$A_{00} := A_{00} + W \cdot W^T$	SYRK
$A_{11} := \mathrm{TRIU}(A_{11}) \cdot \mathrm{TRIU}(A_{11})^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 8 operations per iteration
- 6 of them are MM products

Limitations

- Data dependencies
- Except A_{00} all blocks are "small"

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 2

Algorithm: $[A] := \text{GJE}_{\text{BLK\_V2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} := \text{CHOL}(A_{11})$	POTRF
$\text{TRIU}(A_{11}) := \text{TRIU}(A_{11}^{-1})$	TRTRI
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{00} := A_{00} + A_{01} \cdot A_{01}^T$	SYRK
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{12} := A_{11}^{-T} \cdot A_{12}$	TRMM
$A_{22} := A_{22} - A_{12}^T \cdot A_{12}$	SYRK
$A_{02} := A_{02} - A_{01} \cdot A_{12}$	GEMM
$A_{12} := -(A_{11} \cdot A_{12})$	TRMM
$A_{11} := A_{11} \cdot A_{12}^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 10 operations per iteration
- 8 of them MM products
- Updates of A_{00} and A_{22} concentrate the cost

Limitations

- Data dependencies

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 2

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_V2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11}^{-1})$	TRTRI
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{00} := A_{00} + A_{01} \cdot A_{01}^T$	SYRK
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{12} := A_{11}^{-T} \cdot A_{12}$	TRMM
$A_{22} := A_{22} - A_{12}^T \cdot A_{12}$	SYRK
$A_{02} := A_{02} - A_{01} \cdot A_{12}$	GEMM
$A_{12} := -(A_{11} \cdot A_{12})$	TRMM
$A_{11} := A_{11} \cdot A_{12}^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 10 operations per iteration
- 8 of them MM products
- Updates of A_{00} and A_{22} concentrate the cost

Limitations

- Data dependencies

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 2

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_V2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11}^{-1})$	TRTRI
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{00} := A_{00} + A_{01} \cdot A_{01}^T$	SYRK
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{12} := A_{11}^{-T} \cdot A_{12}$	TRMM
$A_{22} := A_{22} - A_{12}^T \cdot A_{12}$	SYRK
$A_{02} := A_{02} - A_{01} \cdot A_{12}$	GEMM
$A_{12} := -(A_{11} \cdot A_{12})$	TRMM
$A_{11} := A_{11} \cdot A_{12}^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 10 operations per iteration
- 8 of them MM products
- Updates of A_{00} and A_{22} concentrate the cost

Limitations

- Data dependencies

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 2

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_V2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11}^{-1})$	TRTRI
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{00} := A_{00} + A_{01} \cdot A_{01}^T$	SYRK
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{12} := A_{11}^{-T} \cdot A_{12}$	TRMM
$A_{22} := A_{22} - A_{12}^T \cdot A_{12}$	SYRK
$A_{02} := A_{02} - A_{01} \cdot A_{12}$	GEMM
$A_{12} := -(A_{11} \cdot A_{12})$	TRMM
$A_{11} := A_{11} \cdot A_{12}^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 10 operations per iteration
- 8 of them MM products
- Updates of A_{00} and A_{22} concentrate the cost

Limitations

- Data dependencies

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 2

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_V2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$A_{11} := \mathrm{CHOL}(A_{11})$	POTRF
$\mathrm{TRIU}(A_{11}) := \mathrm{TRIU}(A_{11}^{-1})$	TRTRI
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{00} := A_{00} + A_{01} \cdot A_{01}^T$	SYRK
$A_{01} := A_{01} \cdot A_{11}$	TRMM
$A_{12} := A_{11}^{-T} \cdot A_{12}$	TRMM
$A_{22} := A_{22} - A_{12}^T \cdot A_{12}$	SYRK
$A_{02} := A_{02} - A_{01} \cdot A_{12}$	GEMM
$A_{12} := -(A_{11} \cdot A_{12})$	TRMM
$A_{11} := A_{11} \cdot A_{12}^T$	LAUUM

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- 10 operations per iteration
- 8 of them MM products
- Updates of A_{00} and A_{22} concentrate the cost

Limitations

- Data dependencies

Outline

High performance implementations

Based on the Cholesky factorization

On a multi-core CPU

- Based on routines `potrf` and `potri` of a multithread MKL version

On a many-core GPU

- Routines `gpu_potrf` and `gpu_potri` for the GPU have been implemented

On a hybrid CPU-GPU platform

- Each operation is executed on the most covenient device
- CPU and GPU work jointly in the computation of the Cholesky factorization

High performance implementations

Based on the GJE algorithm

On a multi-core CPU

- Two implementations, one per each variant of the algorithm
- Based on the usage of MKL routines

On a many-core GPU

- Two implementations, one per each variant of the algorithm
- Based on the usage of `gpu_potrf` and `gpu_potri` routines and CUBLAS kernels

Matrix inversion of an SPD matrix

Gauss-Jordan elimination method - variant 2

On a hybrid CPU-GPU platform

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_v2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

  where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

  Determine block size b

  Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

  where A_{11} is $b \times b$

Transfer block A_{11} *to the CPU*

A_{11}	$:=$	$\mathrm{Chol}(A_{11})$	(CPU)
$\mathrm{triu}(A_{11})$	$:=$	$\mathrm{triu}(A_{11}^{-1})$	(CPU)

Transfer block A_{11} *to the GPU*

A_{01}	$:=$	$A_{01} \cdot A_{11}$	(GPU)
A_{00}	$:=$	$A_{00} + A_{01} \cdot A_{01}^T$	(GPU)
A_{01}	$:=$	$A_{01} \cdot A_{11}$	(GPU)
A_{12}	$:=$	$A_{11}^{-T} \cdot A_{12}$	(GPU)
A_{22}	$:=$	$A_{22} - A_{12}^T \cdot A_{12}$	(GPU)
A_{02}	$:=$	$A_{02} - A_{01} \cdot A_{12}$	(GPU)
A_{12}	$:=$	$-(A_{11} \cdot A_{12})$	(GPU)
A_{11}	$:=$	$A_{11} \cdot A_{12}^T$	(CPU)

Transfer block A_{11} *to the GPU*

  Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- Only 3 transfers are necessary per step
- 7 operations executed on the GPU, all of them MM products
- Only 3 small operations executed on the CPU

On a hybrid CPU-GPU platform

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_v2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

Transfer block A_{11} *to the CPU*

A_{11}	$:=$	$\mathrm{Chol}(A_{11})$	(CPU)
$\mathrm{triu}(A_{11})$	$:=$	$\mathrm{triu}(A_{11}^{-1})$	(CPU)
Transfer block A_{11} *to the GPU*			
A_{01}	$:=$	$A_{01} \cdot A_{11}$	(GPU)
A_{00}	$:=$	$A_{00} + A_{01} \cdot A_{01}^T$	(GPU)
A_{01}	$:=$	$A_{01} \cdot A_{11}$	(GPU)
A_{12}	$:=$	$A_{11}^{-T} \cdot A_{12}$	(GPU)
A_{22}	$:=$	$A_{22} - A_{12}^T \cdot A_{12}$	(GPU)
A_{02}	$:=$	$A_{02} - A_{01} \cdot A_{12}$	(GPU)
A_{12}	$:=$	$-(A_{11} \cdot A_{12})$	(GPU)
A_{11}	$:=$	$A_{11} \cdot A_{12}^T$	(CPU)

Transfer block A_{11} *to the GPU*

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- Only 3 transfers are necessary per step
- 7 operations executed on the GPU, all of them MM products
- Only 3 small operations executed on the CPU

On a hybrid CPU-GPU platform

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_v2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$

  where A_{TL} is 0×0 and A_{BR} is $n \times n$

while $m(A_{TL}) < m(A)$ **do**

  Determine block size b

  Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

  where A_{11} is $b \times b$

Transfer block A_{11} to the CPU

$$\begin{array}{rcll} A_{11} & := & \mathrm{Chol}(A_{11}) & \text{(CPU)} \\ \mathrm{triu}(A_{11}) & := & \mathrm{triu}(A_{11}^{-1}) & \text{(CPU)} \end{array}$$

Transfer block A_{11} to the GPU

$$\begin{array}{rcll} A_{01} & := & A_{01} \cdot A_{11} & \text{(GPU)} \\ A_{00} & := & A_{00} + A_{01} \cdot A_{01}^T & \text{(GPU)} \\ A_{01} & := & A_{01} \cdot A_{11} & \text{(GPU)} \\ A_{12} & := & A_{11}^{-T} \cdot A_{12} & \text{(GPU)} \\ A_{22} & := & A_{22} - A_{12}^T \cdot A_{12} & \text{(GPU)} \\ A_{02} & := & A_{02} - A_{01} \cdot A_{12} & \text{(GPU)} \\ A_{12} & := & -(A_{11} \cdot A_{12}) & \text{(GPU)} \\ A_{11} & := & A_{11} \cdot A_{12}^T & \text{(CPU)} \end{array}$$

Transfer block A_{11} to the GPU

  Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- Only 3 transfers are necessary per step
- 7 operations executed on the GPU, all of them MM products
- Only 3 small operations executed on the CPU

On a hybrid CPU-GPU platform

Algorithm: $[A] := \mathrm{GJE}_{\mathrm{BLK\_v2}}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right)$
  where A_{TL} is 0×0 and A_{BR} is $n \times n$
while $m(A_{TL}) < m(A)$ **do**
  Determine block size b
  Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

  where A_{11} is $b \times b$

Transfer block A_{11} *to the CPU*

A_{11}	$:=$	$\mathrm{Chol}(A_{11})$	(CPU)
$\mathrm{triu}(A_{11})$	$:=$	$\mathrm{triu}(A_{11}^{-1})$	(CPU)

Transfer block A_{11} *to the GPU*

A_{01}	$:=$	$A_{01} \cdot A_{11}$	(GPU)
A_{00}	$:=$	$A_{00} + A_{01} \cdot A_{01}^T$	(GPU)
A_{01}	$:=$	$A_{01} \cdot A_{11}$	(GPU)
A_{12}	$:=$	$A_{11}^{-T} \cdot A_{12}$	(GPU)
A_{22}	$:=$	$A_{22} - A_{12}^T \cdot A_{12}$	(GPU)
A_{02}	$:=$	$A_{02} - A_{01} \cdot A_{12}$	(GPU)
A_{12}	$:=$	$-(A_{11} \cdot A_{12})$	(GPU)
A_{11}	$:=$	$A_{11} \cdot A_{12}^T$	(CPU)

Transfer block A_{11} *to the GPU*

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline \star & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline \star & A_{11} & A_{12} \\ \hline \star & \star & A_{22} \end{array}\right)$$

endwhile

- Only 3 transfers are necessary per step
- 7 operations executed on the GPU, all of them MM products
- Only 3 small operations executed on the CPU

Outline

Numerical results

Hardware and software

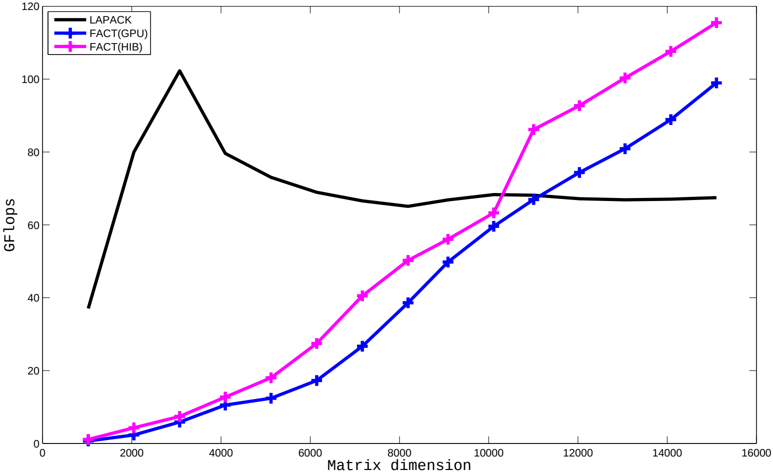
- **Hardware**
  - Platform consisting of eight INTEL Xeon QuadCore X7550 processors at 2.0GHz. (32 cores) connected to an NVIDIA C2050 (448 cores)
- **Software**
  - Computations on the CPU are performed using kernels from MKL v.11.0
  - While computations on the GPU are performed using CUBLAS v.3.2

All experiments performed using single precision arithmetic

Results for matrices with $1000 \leq n \leq 15000$

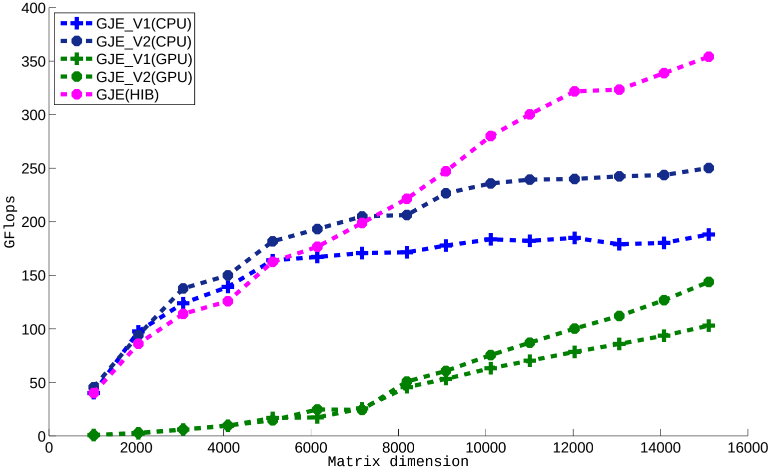
Numerical results

Implementations based on the Cholesky factorization



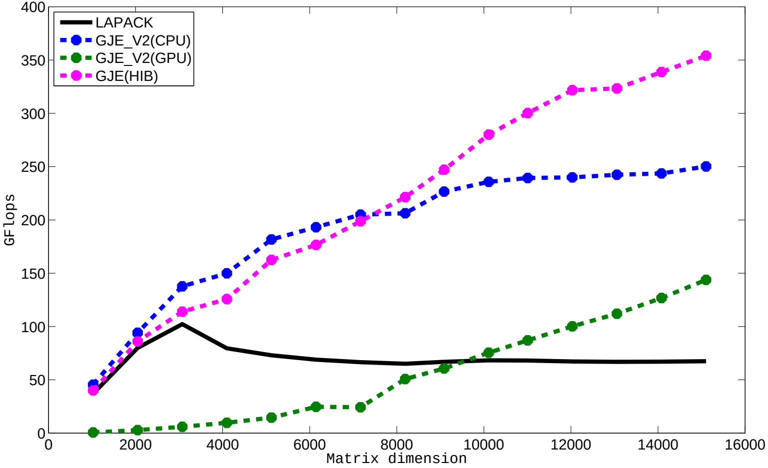
Numerical results

Implementations based on the Gauss-Jordan elimination



Numerical results

Best implementations



Outline

Conclusions

- We have studied the inversion of symmetric positive definite matrices on a hybrid CPU-GPU platform
- This operation appears in many scientific applications and features a high computational cost
- GJE-based routines exhibit a remarkable performance
- An hybrid implementation, where each task is executed on the most convenient device, provides the best performance

Future works

- Overlap communications and computations using asynchronous transfers
- Use of different block sizes for each architecture
- Use of multiple GPUs
- Employ other GPU kernels that outperform CUBLAS

Thanks.