A Domain-Specific Compiler for Linear Algebra Operations

Diego Fabregat-Traver and Prof. Paolo Bientinesi

AICES, RWTH Aachen
fabregat@aices.rwth-aachen.de

7th Intl Workshop on Automatic Performance Tuning (iWAPT)
Kobe, July 17th, 2012

How to efficiently solve... ?

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$

How to efficiently solve... ?

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$
  - ➥ GELS

How to efficiently solve... ?

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$
  - ➥ GELS
- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
  - ➥ ?

How to efficiently solve... ?

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$
  - ➥ GELS
- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
  - ➥ ? $\rightarrow$ Reduce to above

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$
  - ➥ GELS
- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
  - ➥ ? $\rightarrow$ Reduce to above

... sequences of such problems

- $\begin{cases} b_{ij} := (X_i^T M_j^{-1} X_i)^{-1} X_i^T M_j^{-1} y_j \\ M_j := h_j \Phi + (1 - h_j) I \end{cases}$

How to efficiently solve... ?

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$
  - ➥ GELS
- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
  - ➥ ? $\rightarrow$ Reduce to above

... sequences of such problems

- $\begin{cases} b_{ij} := (X_i^T M_j^{-1} X_i)^{-1} X_i^T M_j^{-1} y_j \\ M_j := h_j \Phi + (1 - h_j) I \end{cases}$
  - ➥ Smart mapping onto BLAS/LAPACK

RWTH AACHEN UNIVERSITY

... classic problems

- $b := \left(X^T X\right)^{-1} X^T y$
  - ➥ GELS
- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
  - ➥ ? $\rightarrow$ Reduce to above

... sequences of such problems

- $\begin{cases} b_{ij} := (X_i^T M_j^{-1} X_i)^{-1} X_i^T M_j^{-1} y_j \\ M_j := h_j \Phi + (1 - h_j) I \end{cases}$
  - ➥ Smart mapping onto BLAS/LAPACK
  - ➥ The decomposition is not unique: many algorithms

AICES

How to efficiently solve... ?

... derivatives

- $f : AX = B$
- $\frac{df}{dv}$?

... derivatives

- $f : AX = B$
- $\frac{df}{dv}$?
  - $A'X + AX' = B'$

... derivatives

- $f : AX = B$
- $\frac{df}{dv}$?
  - $A'X + AX' = B'$
  - $AX' = B'$
  - $A'X + AX' = 0$

... derivatives

- $f : AX = B$
- $\frac{df}{dv}$?
  - $A'X + AX' = B'$
  - $AX' = B'$
  - $A'X + AX' = 0$

➥ Smart mapping onto BLAS/LAPACK

➥ Different patterns require different algorithms

Linear Algebra Compiler

Input Matrix equation + Knowledge

Linear Algebra Compiler

Input	Matrix equation + Knowledge
Output	Family of algorithms

Linear Algebra Compiler	
Input	Matrix equation + Knowledge
Output	Family of algorithms
Approach	Map onto high-performance kernels

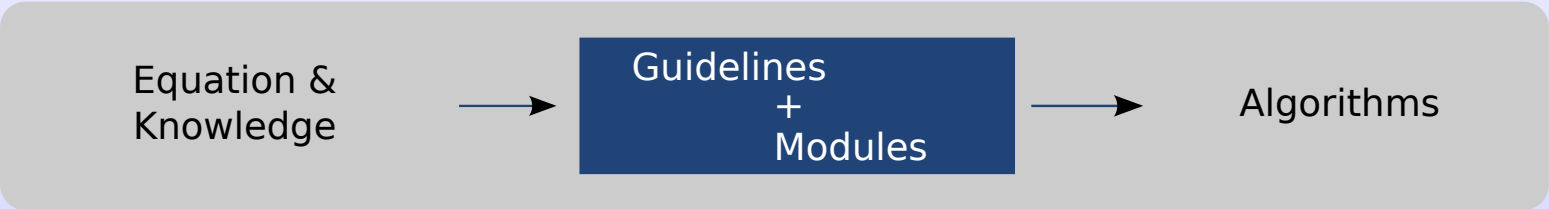
Linear Algebra Compiler	
Input	Matrix equation + Knowledge
Output	Family of algorithms
Approach	Map onto high-performance kernels

Heuristics: Search led by knowledge (mathematical and app-specific)

1. Automation
2. Automation (II)
3. What's next?
4. Conclusions

Automation

Automation



How to explore the search space

- Inverse operator:
  - A^{-1}: factorization
    - ➥ $LL^T = A, \quad QR = A, \quad ZWZ^T = A, \ldots$

How to explore the search space

- Inverse operator:
  - A^{-1}: factorization
    - $LL^T = A, \quad QR = A, \quad ZWZ^T = A$, ...
  - $(X^T X)^{-1}$: factorization or mapping onto kernels
    - $S := X^T X$
    - $QR = X$

How to explore the search space

- Inverse operator:
  - A^{-1}: factorization
    - $LL^T = A, \quad QR = A, \quad ZWZ^T = A$, ...
  - $(X^T X)^{-1}$: factorization or mapping onto kernels
    - $S := X^T X$
    - $QR = X$
- Mapping onto kernels
  - Reuse computations: $S = X^T L^{-T} L^{-1} XC$
    (1) $K := L^{-1} X$
    (2) $S = K^T KC$

How to explore the search space

- Inverse operator:
  - A^{-1}: factorization
    - ➥ $LL^T = A, \quad QR = A, \quad ZWZ^T = A$, ...
  - $(X^T X)^{-1}$: factorization or mapping onto kernels
    - ➥ $S := X^T X$
    - ➥ $QR = X$
- Mapping onto kernels
  - Reuse computations: $S = X^T L^{-T} L^{-1} XC$
    - (1) $K := L^{-1} X$
    - (2) $S = K^T KC$
  - Reducing flops: $S = R^{-1} Q^T Ly$

Modules

- Matrix Algebra
- Properties and Inference
- Kernels
- Sequences

Properties of the operands

- Size and shape
- Diagonal, Lower/Upper triangular
- Symmetric, SPD, ...
- Orthogonal, ...

Properties of the operands

- Size and shape
- Diagonal, Lower/Upper triangular
- Symmetric, SPD, ...
- Orthogonal, ...

Propagation of properties

- $X \in R^{n \times p},\ M \in R^{n \times n} \Rightarrow X^T M^{-1} X \in R^{p \times p}$

Properties of the operands

- Size and shape
- Diagonal, Lower/Upper triangular
- Symmetric, SPD, ...
- Orthogonal, ...

Propagation of properties

- $X \in R^{n \times p},\ M \in R^{n \times n} \Rightarrow\ X^T M^{-1} X \in R^{p \times p}$
- $A := X^T X \Rightarrow A$ is SPD

Encoded knowledge

- Operators: +, -, *, T, -1
- Properties: commutative, associative, distributive, ...

Encoded knowledge

- Operators: +, -, *, ^{T}, $^{-1}$
- Properties: commutative, associative, distributive, ...

Algebraic transformations

- $(LL^T)^{-1} \Rightarrow L^{-T}L^{-1}$

Encoded knowledge

- Operators: +, -, *, ^{T}, $^{-1}$
- Properties: commutative, associative, distributive, ...

Algebraic transformations

- $(LL^T)^{-1} \Rightarrow L^{-T}L^{-1}$
- $(R^TQ^TQR)^{-1}R^TQ^T$

Encoded knowledge

- Operators: +, -, *, ^{T}, $^{-1}$
- Properties: commutative, associative, distributive, ...

Algebraic transformations

- $(LL^T)^{-1} \Rightarrow L^{-T}L^{-1}$
- $(R^TQ^TQR)^{-1}R^TQ^T$

  $\Rightarrow (R^TR)^{-1}R^TQ^T$

Encoded knowledge

- Operators: +, -, *, ^{T}, $^{-1}$
- Properties: commutative, associative, distributive, ...

Algebraic transformations

- $(LL^T)^{-1} \Rightarrow L^{-T}L^{-1}$
- $(R^TQ^TQR)^{-1}R^TQ^T$

  $\Rightarrow (R^TR)^{-1}R^TQ^T$

  $\Rightarrow R^{-1}R^{-T}R^TQ^T$

Encoded knowledge

- Operators: +, -, *, ^{T}, $^{-1}$
- Properties: commutative, associative, distributive, ...

Algebraic transformations

- $(LL^T)^{-1} \Rightarrow L^{-T}L^{-1}$
- $(R^TQ^TQR)^{-1}R^TQ^T$

  $\Rightarrow (R^TR)^{-1}R^TQ^T$

  $\Rightarrow R^{-1}R^{-T}R^TQ^T$

  $\Rightarrow R^{-1}Q^T$

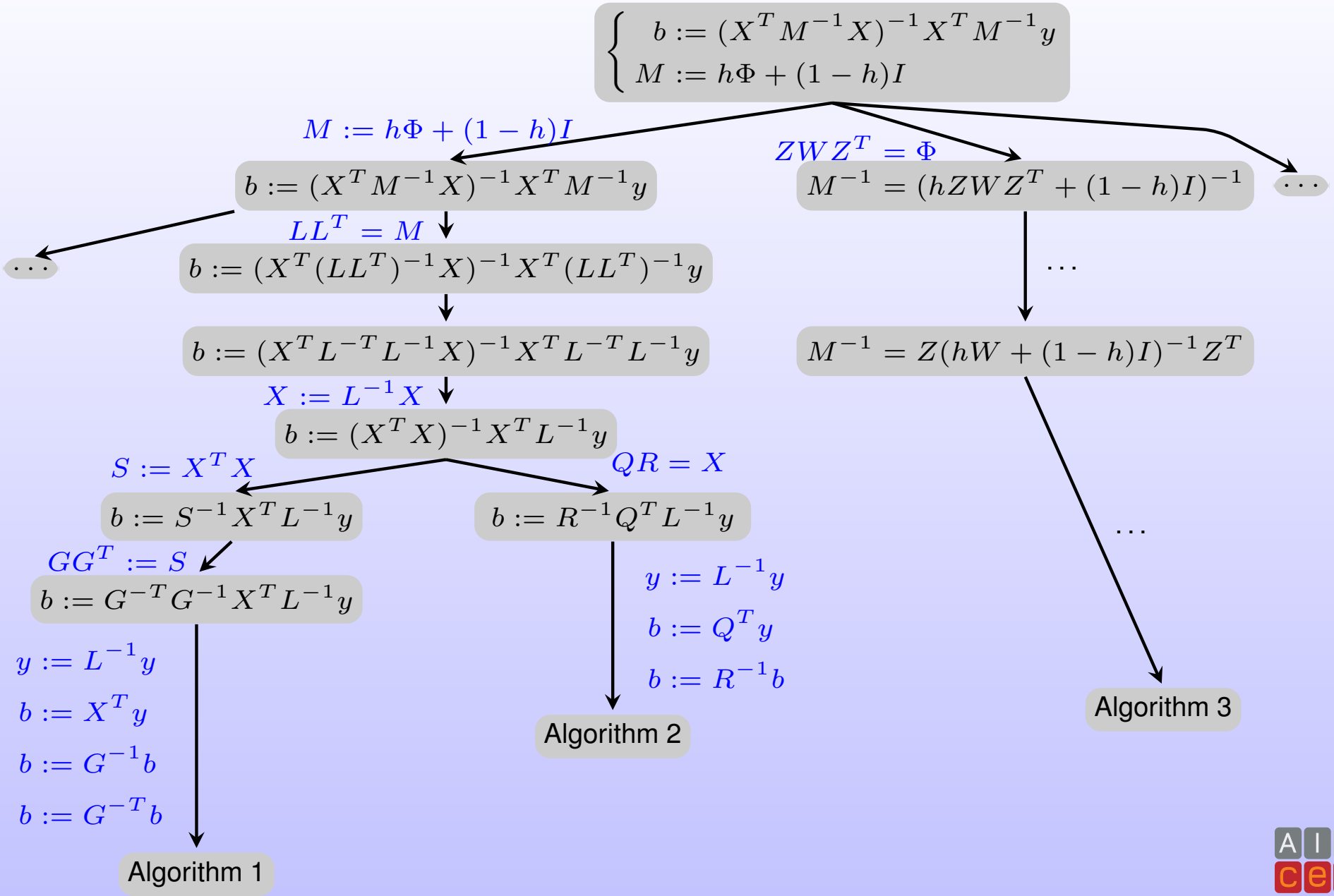
Database of building blocks

- BLAS / LAPACK
  - Factorizations: QR, LU, Cholesky, Eigen, ...
  - BLAS: GEMM, TRSM, GEMV, DOT, ...
  - LAPACK: inverse of a triangular matrix, ...
- Extensible

Example: GLS in GWAS

$$\begin{cases} b := (X^T M^{-1} X)^{-1} X^T M^{-1} y \\ M := h\Phi + (1-h)I \end{cases}$$

Example: GLS in GWAS



$$\begin{cases} b_{ij} = (X_i^T M_j^{-1} X_i)^{-1} X_i^T M_j^{-1} y_j & \text{with } 1 \leq i \leq m \\ M_j = h_j \Phi + (1 - h_j) I & \text{and } 1 \leq j \leq t. \end{cases}$$

- We have to solve not one but a sequence of **correlated** problems

$$\begin{cases} b_{ij} = (X_i^T M_j^{-1} X_i)^{-1} X_i^T M_j^{-1} y_j & \text{with } 1 \leq i \leq m \\ M_j = h_j \Phi + (1 - h_j) I & \text{and } 1 \leq j \leq t. \end{cases}$$

- We have to solve not one but a sequence of **correlated** problems
- Goal: **reuse of computation**

Naive approach: `for i, for j, ...`

$$b_{ij} = \left(X_i^T M_j^{-1} X_i\right)^{-1} X_i^T M_j^{-1} y_j$$

`for` $i = 1 : m$

  `for` $j = 1 : t$

    $LL^T = M_j$

    $X^T \leftarrow X_i^T L^{-T}$

    $QR = X$

    $y \leftarrow L^{-1} y_j$

    $b \leftarrow Q^T y$

    $b_{ij} \leftarrow R^{-1} b$

Tracking the dependencies

$$b_{ij} = \left(X_i^T M_j^{-1} X_i\right)^{-1} X_i^T M_j^{-1} y_j$$

$$\begin{aligned}
&\texttt{for } i = 1 : m \\
&\quad \texttt{for } j = 1 : t \\
&\qquad L_j L_j^T = M_j \\
&\qquad X_{ij}^T \leftarrow X_i^T L_j^{-T} \\
&\qquad Q_{ij} R_{ij} = X_{ij} \\
&\qquad y_j \leftarrow L_j^{-1} y_j \\
&\qquad b_{ij} \leftarrow Q_{ij}^T y_j \\
&\qquad b_{ij} \leftarrow R_{ij}^{-1} b_{ij}
\end{aligned}$$

Loop Transposition

$$b_{ij} = \left(X_i^T M_j^{-1} X_i\right)^{-1} X_i^T M_j^{-1} y_j$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

```
for j = 1 : t
    for i = 1 : m
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

Reordering

$$b_{ij} = \left(X_i^T M_j^{-1} X_i\right)^{-1} X_i^T M_j^{-1} y_j$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

```
for j = 1 : t
    for i = 1 : m
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

$$b_{ij} = \left(X_i^T M_j^{-1} X_i\right)^{-1} X_i^T M_j^{-1} y_j$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

```
for j = 1 : t
    L_j L_j^T = M_j
    y_j ← L_j^{-1} y_j
    for i = 1 : m
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

Steps

- Input: Dependencies operand $\leftrightarrow$ dimension
- Label each operand
- Propagation of dependencies
- Reordering
- Loop transposition

How to pick a variant?

- Many algorithms for a single target equation. How do we pick one?
- Flop count: Often not too descriptive

Scenario	Alg. 1	Alg. 2	Alg. 3
One instance	$O(n^3)$	$O(n^3)$	$O(n^3)$
2D sequence	$O(tn^3 + mtn^2)$	$O(tn^3 + mtn^2)$	$O(n^3 + mtn)$

How to pick a variant?

- Many algorithms for a single target equation. How do we pick one?
- Flop count: Often not too descriptive

Scenario	Alg. 1	Alg. 2	Alg. 3
One instance	$O(n^3)$	$O(n^3)$	$O(n^3)$
2D sequence	$O(tn^3 + mtn^2)$	$O(tn^3 + mtn^2)$	$O(n^3 + mtn)$

Next talk: Prediction

Conclusions and future work

So far...

- Domain-specific linear algebra compiler
- Guidelines + Modules + Knowledge
- Families of algorithms
- Not only a theoretical exercise:
  - AD: speedups ranging from 20 to 300
  - GWAS: speedups larger than 100

Conclusions and future work

So far...

- Domain-specific linear algebra compiler
- Guidelines + Modules + Knowledge
- Families of algorithms
- Not only a theoretical exercise:
  - AD: speedups ranging from 20 to 300
  - GWAS: speedups larger than 100

TO-DO

- Encoding more available knowledge
- How to pick the “best” algorithm?
- Working on a Fortran code generator

Thanks to:

- Dr. Edoardo Di Napoli
- Matthias Petschow
- Roman Iakymchuk
- Elmar Peise

Financial support from the **Deutsche Forschungsgemeinschaft** (German Research Association) through grant GSC 111 is gratefully acknowledged.