Teaching computers linear algebra

Paolo Bientinesi
Aachen Institute for Computational Engineering Science
RWTH Aachen University

January 31, 2018
Friedrich-Schiller-Universität Jena

Deutsche Forschungsgemeinschaft DFG

HPAC High Performance and Automatic Computing

RWTH AACHEN UNIVERSITY

30-second talk

30-second talk

- Computers are great with numbers (scalars)

30-second talk

- Computers are great with numbers (scalars)
- Scientists work with vectors, matrices, and tensors

30-second talk

- Computers are great with numbers (scalars)
- Scientists work with vectors, matrices, and tensors
- Computers are not great with those

30-second talk

- Computers are great with numbers (scalars)
- Scientists work with vectors, matrices, and tensors
- Computers are not great with those
- Why? What can we do about it?

The world of scientific computing

$$\frac{\partial}{\partial t}(\rho\ \mathbf{u}) + \nabla \cdot (\rho\ \mathbf{u} \otimes \mathbf{u}) = -\nabla \cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$$

Cauchy momentum eqn.

$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

Lennard-Jones potential

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r}, t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r}, t)\right]\Psi(\mathbf{r}, t)$$

Schrödinger eqn.

$\vdots$

The world of scientific computing

$$\frac{\partial}{\partial t}(\rho\ \mathbf{u}) + \nabla\cdot(\rho\ \mathbf{u}\otimes\mathbf{u}) = -\nabla\cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$$

Cauchy momentum eqn.

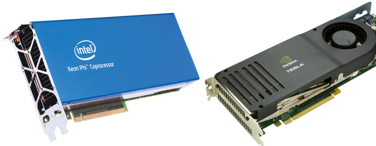
$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

Lennard-Jones potential

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

Schrödinger eqn.

⋮



The world of scientific computing

$$\frac{\partial}{\partial t}(\rho\ \mathbf{u}) + \nabla\cdot(\rho\ \mathbf{u}\otimes\mathbf{u}) = -\nabla\cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$$

CAUCHY MOMENTUM EQN.

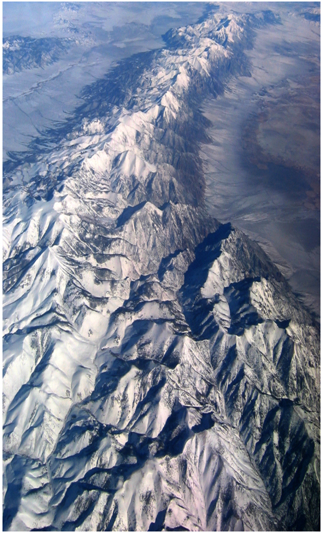
$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

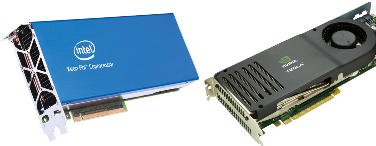
LENNARD-JONES POTENTIAL

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

SCHRÖDINGER EQN.

⋮





Roadmap

$$\frac{\partial}{\partial t}(\rho\ \mathbf{u}) + \nabla\cdot(\rho\ \mathbf{u}\otimes\mathbf{u}) = -\nabla\cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$$

CAUCHY MOMENTUM EQN.

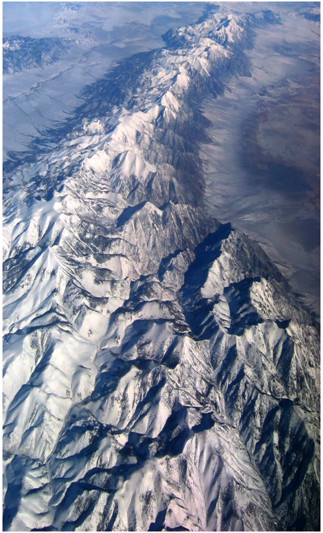
$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

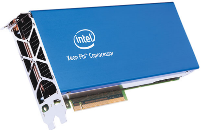
LENNARD-JONES POTENTIAL

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

SCHRÖDINGER EQN.

⋮





Continuous mathematics

Roadmap

$$\frac{\partial}{\partial t}(\rho\ \mathbf{u}) + \nabla\cdot(\rho\ \mathbf{u}\otimes\mathbf{u}) = -\nabla\cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$$

Cauchy momentum eqn.

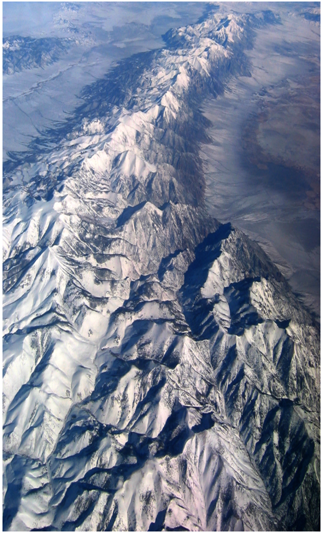
$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

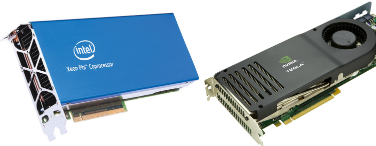
Lennard-Jones potential

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

Schrödinger eqn.

⋮





Continuous mathematics $\rightarrow$ **Discrete mathematics**

Roadmap

$$\frac{\partial}{\partial t}(\rho\ \mathbf{u}) + \nabla \cdot (\rho\ \mathbf{u} \otimes \mathbf{u}) = -\nabla \cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$$

CAUCHY MOMENTUM EQN.

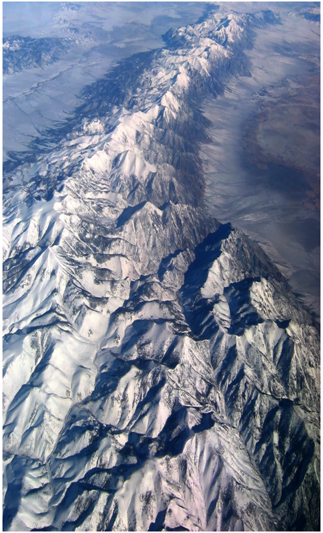
$$V_{LJ} = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$

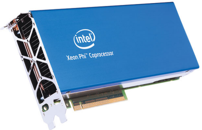
LENNARD-JONES POTENTIAL

$$i\hbar \frac{\partial}{\partial t}\Psi(\mathbf{r}, t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r}, t) \right] \Psi(\mathbf{r}, t)$$

SCHRÖDINGER EQN.

$\vdots$





Continuous mathematics $\rightarrow$ **Discrete mathematics** $\rightarrow$ **. . .** $\rightarrow$ **Computers**

Linear Algebra expressions

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y$$

exponential transient excision

$$\forall i \ \forall j \quad b_{ij} := \left(X_i^T M_j^{-1} X_i \right)^{-1} X_i^T M_j^{1} y_j$$

GWAS

$$\begin{cases} C_\dagger := P C P^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

probabilistic Nordsieck method for ODEs

$$E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

L1-norm minimization on manifolds

$$\begin{cases} x_{k|k-1} = F x_{k-1|k-1} + B u \\ P_{k|k-1} = F P_{k-1|k-1} F^T + Q \\ x_{k|k} = x_{k|k-1} + P_{k|k-1} H^T \times (H P_{k|k-1} H^T + R)^{-1} (z_k - H x_{k|k-1}) \\ P_{k|k} = P_{k|k-1} - P_{k|k-1} H^T \times (H P_{k|k-1} H^T + R)^{-1} H P_{k|k-1} \end{cases}$$

Kalman filter

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y$$

$$\begin{cases} C_\dagger := P C P^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

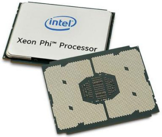
...

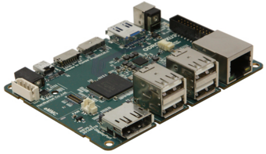
$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y$$

$$\begin{cases} C_\dagger := P C P^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

...





MUL ADD MOV

MOVAPD

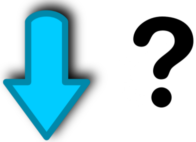
VFMADDPD ...

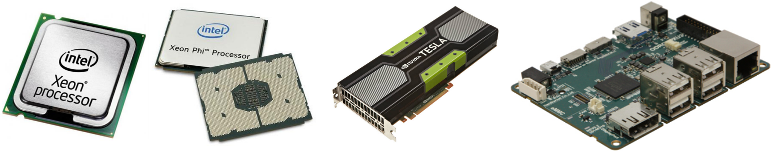
$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y$$

$$\begin{cases} C_\dagger := P C P^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

...





MUL | ADD | MOV

MOVAPD

VFMADDPD ...


IBM
360

Back in the 50s: by hand

- Assembly code (Mnemonics $\rightarrow$ machine code)

Back in the 50s: by hand

- Assembly code (Mnemonics $\rightarrow$ machine code)

- “simple” architectures: no memory hierarchy

  $\text{Cost}(\mathcal{A}lg) \equiv \#\text{operations}(\mathcal{A}lg)$

Back in the 50s: by hand

- Assembly code (Mnemonics $\rightarrow$ machine code)

- "simple" architectures: no memory hierarchy

  $\text{Cost}(\mathcal{A}lg) \equiv \#\text{operations}(\mathcal{A}lg)$

computer efficiency!

Back in the 50s: by hand

- Assembly code (Mnemonics $\rightarrow$ machine code)
- "simple" architectures: no memory hierarchy

  $\text{Cost}(\mathcal{A}lg) \equiv \#\text{operations}(\mathcal{A}lg)$

computer efficiency! … but human productivity?

Since 1957: compiler(s)

- [1954–1957]: FORTRAN (IBM, John Backus)
  "Specifications for the IBM Mathematical FORmula TRANslating system, FORTRAN"

  $\rightarrow$ ...

Since 1957: compiler(s)

- [1954–1957]: FORTRAN (IBM, John Backus)
  "Specifications for the IBM Mathematical FORmula TRANslating system, FORTRAN"

  $\rightarrow$ ... $\rightarrow$ ACM Turing award (1977)

Since 1957: compiler(s)

- [1954–1957]: FORTRAN (IBM, John Backus)
  "Specifications for the IBM Mathematical FORmula TRANslating system, FORTRAN"

  → . . . → ACM Turing award (1977)

- **Pros**: No more Assembly! → increased productivity
  A gigantic body of work on compilers

- **Cons**: Often not the "right" level of abstraction

Since the 70s: libraries

- Identification, standardization, optimization of **building blocks**

  Libraries: LINPACK, BLAS, LAPACK, FFTW, . . .

  Convenience, portability, separation of concerns

Since the 70s: libraries

- Identification, standardization, optimization of **building blocks**

  Libraries: LINPACK, BLAS, LAPACK, FFTW, ...

  Convenience, portability, separation of concerns

- [80s–early 90s]: Memory hierarchy

  Caching, locality, prefetching, ... $\Rightarrow$ $\text{Cost}(\mathcal{A}lg) \neq \#\text{operations}(\mathcal{A}lg)$

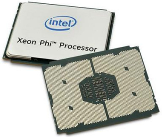
  Libraries: necessity

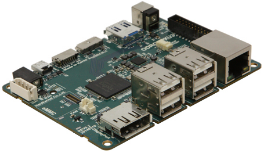
$$x := A(B^TB + A^TR^T\Lambda RA)^{-1}B^TBA^{-1}y$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T(HC_\dagger H^T)^{-1} \end{cases}$$

$$E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T$$

...





MUL ADD MOV

MOVAPD

VFMADDPD ...

$$x := A(B^TB + A^TR^T\Lambda RA)^{-1}B^TBA^{-1}y$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T(HC_\dagger H^T)^{-1} \end{cases}$$

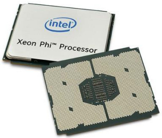
$$E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T$$

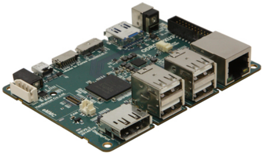
...

$y := \alpha x + y$ | $LU = A$ | $\cdots$ | $C := \alpha AB + \beta C$

$X := A^{-1}B$ | $C := AB^T + BA^T + C$ | $X := L^{-1}ML^{-T}$ | $QR = A$

LINPACK **BLAS** **LAPACK** ...





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$x := A(B^TB + A^TR^T\Lambda RA)^{-1}B^TBA^{-1}y$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T(HC_\dagger H^T)^{-1} \end{cases}$$

$$E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T$$

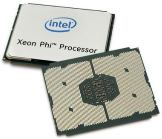
...

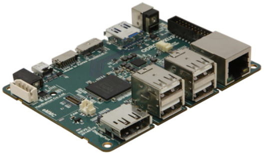
?

$y := \alpha x + y$ | $LU = A$ | $\cdots$ | $C := \alpha AB + \beta C$

$X := A^{-1}B$ | $C := AB^T + BA^T + C$ | $X := L^{-1}ML^{-T}$ | $QR = A$

LINPACK **BLAS** **LAPACK** ...





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$x := A(B^TB + A^TR^T\Lambda RA)^{-1}B^TBA^{-1}y$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T(HC_\dagger H^T)^{-1} \end{cases}$$

$$E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T$$

...

LINEAR ALGEBRA MAPPING PROBLEM (LAMP)

$y := \alpha x + y$

$:= \alpha AB + \beta C$

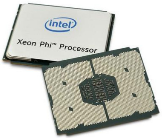
$X := A^{-1}B$

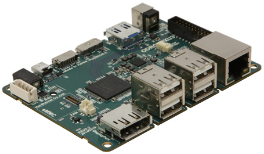
$C := A$

$^{-1}ML^{-T}$

$QR = A$

LINPACK **BLAS** **LAPACK** ...





MUL ADD MOV

MOVAPD

VFMADDPD ...

Linear Algebra Mapping Problem (LAMP)

Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a list of assignments $var_i := EXP_i$

Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a list of assignments $var_i := EXP_i$
- $\mathcal{K}$: a list of available computational kernels (BLAS, LAPACK, ...)

Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a list of assignments $var_i := EXP_i$
- $\mathcal{K}$: a list of available computational kernels (BLAS, LAPACK, ...)
- $\mathcal{M}$: a metric (FLOPs, data movement, stability, time)

Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a list of assignments $var_i := EXP_i$
- $\mathcal{K}$: a list of available computational kernels (BLAS, LAPACK, ...)
- $\mathcal{M}$: a metric (FLOPs, data movement, stability, time)

LAMP:

Find a decomposition of the expressions $\mathcal{E}$ in terms of the kernels $\mathcal{K}$, optimal according to the metric $\mathcal{M}$.

Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a list of assignments $var_i := EXP_i$
- $\mathcal{K}$: a list of available computational kernels (BLAS, LAPACK, ...)
- $\mathcal{M}$: a metric (FLOPs, data movement, stability, time)

LAMP:

Find a decomposition of the expressions $\mathcal{E}$ in terms of the kernels $\mathcal{K}$, optimal according to the metric $\mathcal{M}$.

- Find a decomposition $\rightarrow$ easy
- Achieve optimality $\rightarrow$ NP complete

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

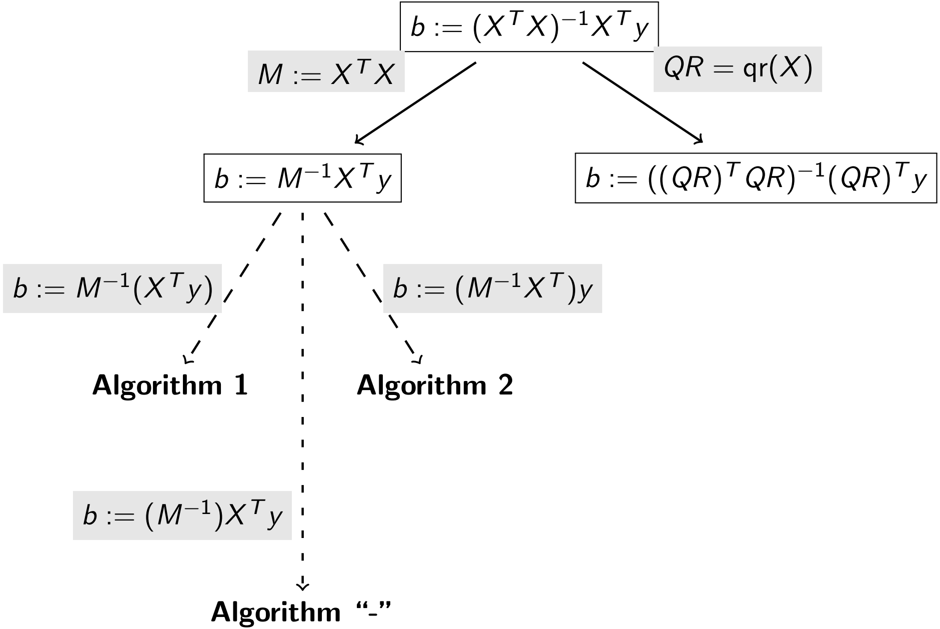
$$b := (X^T X)^{-1} X^T y$$

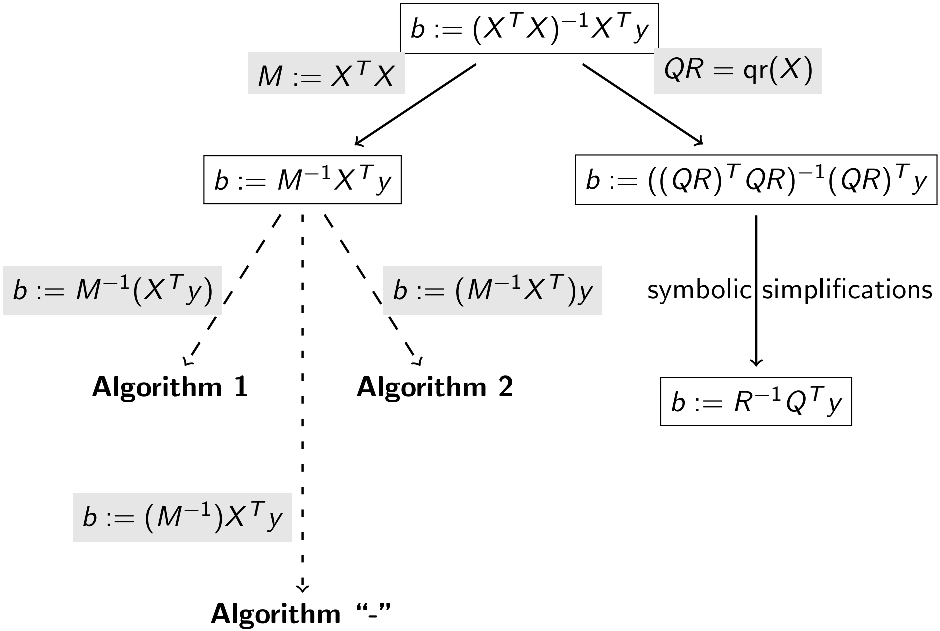
$M := X^T X$

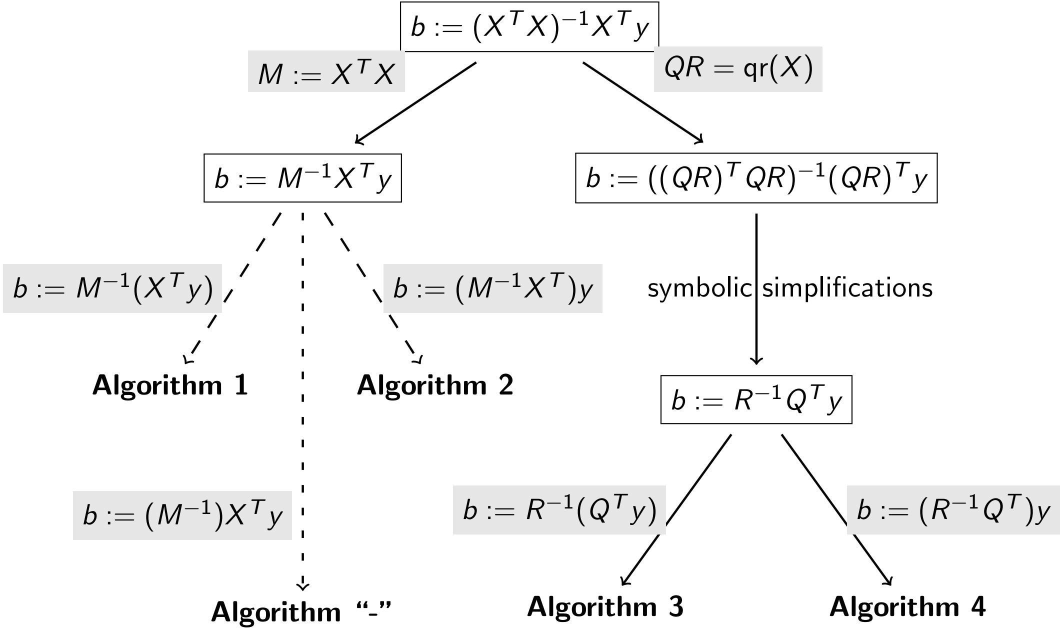
$QR = \mathrm{qr}(X)$

$$b := M^{-1} X^T y$$

$$b := ((QR)^T QR)^{-1} (QR)^T y$$


$b := (X^T X)^{-1} X^T y$
$M := X^T X$
$QR = \mathrm{qr}(X)$
$b := M^{-1} X^T y$
$b := ((QR)^T QR)^{-1} (QR)^T y$
$b := M^{-1}(X^T y)$
$b := (M^{-1} X^T) y$
Algorithm 1
Algorithm 2
$b := (M^{-1}) X^T y$
Algorithm "-"


$b := (X^T X)^{-1} X^T y$
$M := X^T X$
$QR = \mathrm{qr}(X)$
$b := M^{-1} X^T y$
$b := ((QR)^T QR)^{-1} (QR)^T y$
$b := M^{-1}(X^T y)$
$b := (M^{-1} X^T) y$
symbolic simplifications
Algorithm 1
Algorithm 2
$b := R^{-1} Q^T y$
$b := (M^{-1}) X^T y$
Algorithm “-”



$b := (X^T X)^{-1} X^T y$
$M := X^T X$
$QR = \mathrm{qr}(X)$
$b := M^{-1} X^T y$
$b := ((QR)^T QR)^{-1} (QR)^T y$
$b := M^{-1}(X^T y)$
$b := (M^{-1} X^T) y$
symbolic simplifications
Algorithm 1
Algorithm 2
$b := R^{-1} Q^T y$
$b := (M^{-1}) X^T y$
$b := R^{-1}(Q^T y)$
$b := (R^{-1} Q^T) y$
Algorithm "-"
Algorithm 3
Algorithm 4

Many solutions to a well-known problem

High-level languages

- Matlab
- R
- Julia
- Mathematica
- ...

Libraries

- Armadillo
- Blaze
- Blitz
- Eigen
- ...
- NumPy

Many solutions to a well-known problem

High-level languages

- Matlab
- R
- Julia
- Mathematica
- ...

Libraries

- Armadillo
- Blaze
- Blitz
- Eigen
- ...
- NumPy

human productivity!

Many solutions to a well-known problem

High-level languages

- Matlab
- R
- Julia
- Mathematica
- . . .

Libraries

- Armadillo
- Blaze
- Blitz
- Eigen
- . . .
- NumPy

human productivity! . . . but computer efficiency?

A closer look

$\alpha, \beta, x \in \mathbf{R}, \quad A, B, C, X \in \mathbf{R}^{\mathbf{m,n}}$

Commutativity($*$)

$$x := \alpha * \beta * \alpha^{-1} \quad \Rightarrow \quad x := \beta$$

$$X := A * B * A^{-1} \quad \not\Rightarrow \quad x := B$$

A closer look

$\alpha, \beta, x \in \mathbf{R}, \quad A, B, C, X \in \mathbf{R^{m,n}}$

Commutativity($*$)

$$x := \alpha * \beta * \alpha^{-1} \quad \Rightarrow \quad x := \beta$$
$$X := A * B * A^{-1} \quad \not\Rightarrow \quad x := B$$

Associativity($*$)

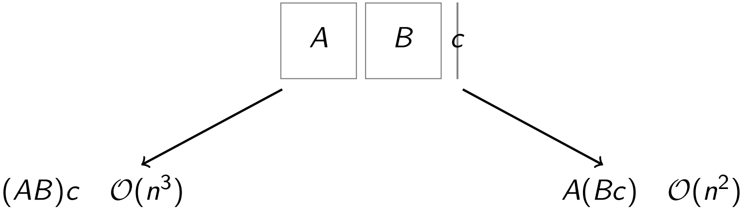
$$X := (A * B) * C \quad \equiv \quad X := A * (B * C)$$

But

$$\text{Cost}\left((A * B) * C\right) \quad \not\equiv \quad \text{Cost}\left(A * (B * C)\right)$$

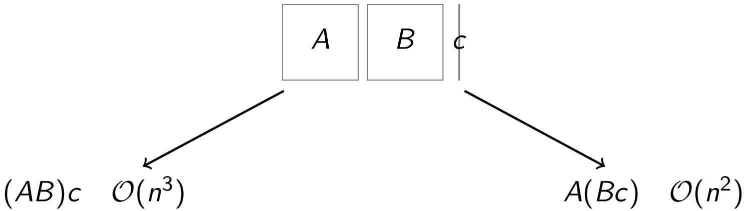
Challenges

- **Parenthesisation**



Challenges

- **Parenthesisation**



$\Rightarrow$ Matrix Chain Algorithm

Challenges

- **Parenthesisation**

  In practice:

  - Unary operators: transposition, inversion $(X := AB^T C^{-T} D + \dots)$
  - Overlapping kernels (e.g., $L \leftarrow L^{-1}$, $X = A^{-1}B$)
  - Decompositions (e.g., $A \rightarrow Q^T DQ$, $A \rightarrow LU$)
  - Properties & specialized kernels (`GEMM`, `TRMM`, `SYMM`, ...)

Challenges

- **Parenthesisation**

In practice:

- Unary operators: transposition, inversion $(X := AB^T C^{-T} D + \dots)$
- Overlapping kernels (e.g., $L \leftarrow L^{-1}$, $X = A^{-1}B$)
- Decompositions (e.g., $A \rightarrow Q^T DQ$, $A \rightarrow LU$)
- Properties & specialized kernels (`GEMM`, `TRMM`, `SYMM`, `...`)

$\Rightarrow$ **Generalized** Matrix Chain Algorithm

Challenges – not all flops were created equal

- **Metric:** #FLOPs vs. execution time . . . vs. numerical stability

$$\underset{\mathcal{A}}{\text{argmin}}\,(\ \text{FLOPs}(\mathcal{A})\ ) \neq \underset{\mathcal{A}}{\text{argmin}}\,(\ \text{time}(\mathcal{A})\ )$$

Challenges – not all flops were created equal

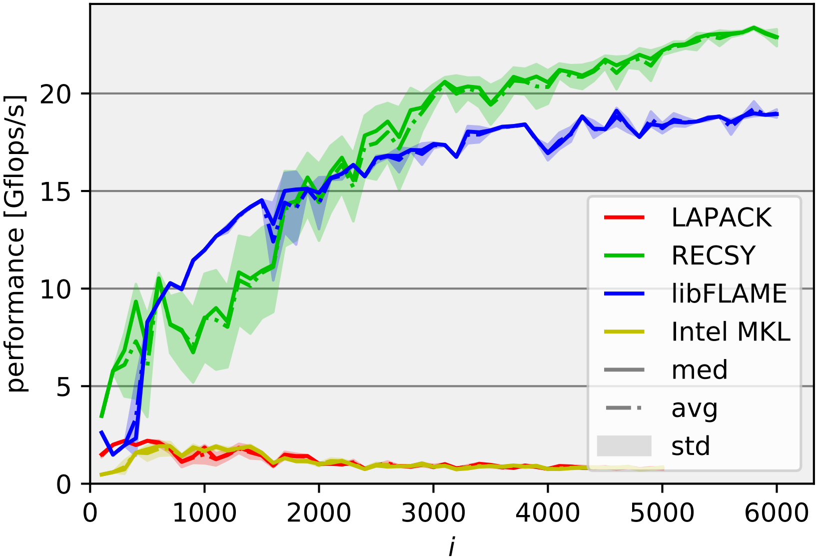
- **Metric:** #FLOPs vs. execution time ... vs. numerical stability

$$\underset{\mathcal{A}}{\text{argmin}} \left(\text{ FLOPs}(\mathcal{A}) \right) \neq \underset{\mathcal{A}}{\text{argmin}} \left(\text{ time}(\mathcal{A}) \right)$$

$\Rightarrow$ **Performance prediction** (efficiency)

Challenges – not all flops were created equal

▶



Challenges

- **Parallelism?** Libraries, explicit multi-threading, runtime, hybrid?

$$X := A((B^T C^{-T})D) \quad \text{vs.} \quad X := (AB^T)(C^{-T} D) \quad \text{vs.} \quad \ldots$$

Challenges

- **Parallelism?** Libraries, explicit multi-threading, runtime, hybrid?

$$X := A((B^T C^{-T})D) \quad \text{vs.} \quad X := (AB^T)(C^{-T}D) \quad \text{vs.} \quad \ldots$$

$\Rightarrow$ **Performance prediction** (efficiency, scalability)

Challenges

- **Linear algebra knowledge**: operators, identities, theorems

- Distributivity, commutativity, partitionings, ...
- $((QR)^T QR)^{-1}(QR)^T y \;\rightarrow\; (R^T Q^T QR)^{-1} R^T Q^T y \;\rightarrow\; R^{-1} R^{-T} R^T Q^T y \;\rightarrow\; R^{-1} Q^T y$
- $\mathrm{SPD}(A) \rightarrow \mathrm{SPD}(A_{BR} - A_{BL} A_{TL}^{-1} A_{BL}^T)$ Schur complement
- $\cdots$

Challenges

- **Linear algebra knowledge**: operators, identities, theorems

- Distributivity, commutativity, partitionings, ...
- $((QR)^T QR)^{-1}(QR)^T y \;\rightarrow\; (R^T Q^T QR)^{-1} R^T Q^T y \;\rightarrow\; R^{-1} R^{-T} R^T Q^T y \;\rightarrow\; R^{-1} Q^T y$
- $\mathrm{SPD}(A) \rightarrow \mathrm{SPD}(A_{BR} - A_{BL} A_{TL}^{-1} A_{BL}^T)$ Schur complement
- ...

$\Rightarrow$ "**Knowledge base**" – expert system – pattern matching

Challenges

- **Inference of properties**

$$E := L_1 * U^T * L_2 \qquad \text{triangular}(E)\ ?$$

$$E := Q^{-1}U(I + U^T Q^{-1} U)^{-1} U^T \qquad \text{properties}(I + U^T Q^{-1} U)\ ?$$

$$\lambda(A, B) \wedge \begin{cases} \text{symm}(A) \\ \text{SPD}(B) \end{cases} \rightarrow \lambda(L^{-T} A L^{-1}) \qquad \text{symmetric}(L^{-T} A L^{-1})\ ?$$

Challenges

- **Inference of properties**

$$E := L_1 * U^T * L_2 \qquad \text{triangular}(E)\ ?$$

$$E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T \qquad \text{properties}(I + U^TQ^{-1}U)\ ?$$

$$\lambda(A, B) \wedge \begin{cases} \text{symm}(A) \\ \text{SPD}(B) \end{cases} \rightarrow \lambda(L^{-T}AL^{-1}) \qquad \text{symmetric}(L^{-T}AL^{-1})\ ?$$

⇒ **Symbolic analysis** – pattern matching

Challenges

- **Common subexpressions**

$$\begin{cases} X := AB^{-T}C \\ Y := B^{-1}A^{T}D \end{cases} \rightarrow \begin{cases} Z := AB^{-T} \\ X := ZC \\ Y := Z^{T}D \end{cases}$$

Challenges

- **Common subexpressions**

$$\begin{cases} X := AB^{-T}C \\ Y := B^{-1}A^{T}D \end{cases} \rightarrow \begin{cases} Z := AB^{-T} \\ X := ZC \\ Y := Z^{T}D \end{cases}$$

$\Rightarrow$ **Pattern matching**

Example

$$w := AB^{-1}c, \quad \mathrm{SPD}(B)$$

Example

$$w := AB^{-1}c, \quad \mathsf{SPD}(B)$$

Naive $\leftarrow$ NEVER!!

```
w = A*inv(B)*c
```

Example

$$w := AB^{-1}c, \quad \text{SPD}(B)$$

Naive $\leftarrow$ NEVER!!

```
w = A*inv(B)*c
```

Recommended

```
w = A*(B\c)
```

Example

$$w := AB^{-1}c, \quad \mathrm{SPD}(B)$$

Naive $\leftarrow$ NEVER!!

```
w = A*inv(B)*c
```

Recommended

```
w = A*(B\c)
```

Expert

```
L = Chol(B)
w = A * (L'\(L\c))
```

Example

$$w := AB^{-1}c, \quad \mathsf{SPD}(B)$$

Naive $\leftarrow$ NEVER!!

```
w = A*inv(B)*c
```

Recommended

```
w = A*(B\c)
```

Expert

```
L = Chol(B)
w = A * (L'\(L\c))
```

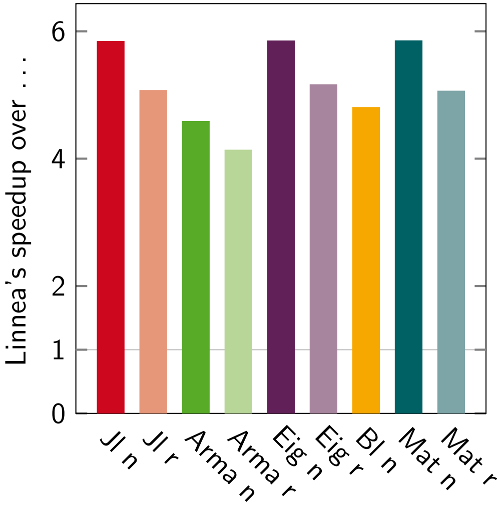
Generated – "Linnea" by H. Barthels

```
ml0 = A; ml1 = B; ml2 = c;
potrf!('L', ml1)
trsv!('L', 'N', 'N', ml1, ml2)
trsv!('L', 'T', 'N', ml1, ml2)
ml3 = Array{Float64}(10)
gemv!('N', 1.0, ml0, ml2, 0.0, ml3)
w = ml3
```

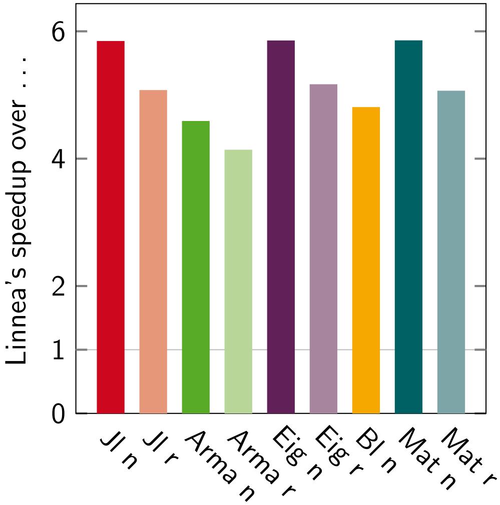
Experiments

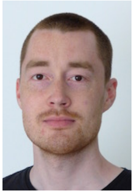
#	Example	
1	$b := (X^T X)^{-1} X^T y$	FullRank(X)
2	$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$	SPD(M), FullRank(X)
3	$W := A^{-1} B C D^{-T} E F$	LowTri(A), UppTri(D, E)
4	$\begin{cases} X := A B^{-1} C \\ Y := D B^{-1} A^T \end{cases}$	SPD(B)
5	$x := W(A^T (A W A^T)^{-1} b - c)$	FullRank(A, W) Diag(W), Pos(W)
⋮		

Linnea – Performance results



Linnea – Performance results





- Henrik Barthels
- `https://github.com/HPAC/linnea`
- "The Generalized Matrix Chain Algorithm" CGO'18
- "Program Synthesis for Algebraic Domains" (submitted)

Objectives

- **Linnea** as a compiler (off line) vs. **Linnea** as an interpreter (real time)
- Integration into languages and libraries
- Challenges: We only scraped the surface
  Memory usage, parallelism, common subexpressions, . . .
- Beyond matrices: tensors

Objectives

- **Linnea** as a compiler (off line) vs. **Linnea** as an interpreter (real time)
- Integration into languages and libraries
- Challenges: We only scraped the surface
  Memory usage, parallelism, common subexpressions, ...
- Beyond matrices: tensors
- Ultimately:

human productivity	&	computer efficiency