Programming languages and linear algebra computations

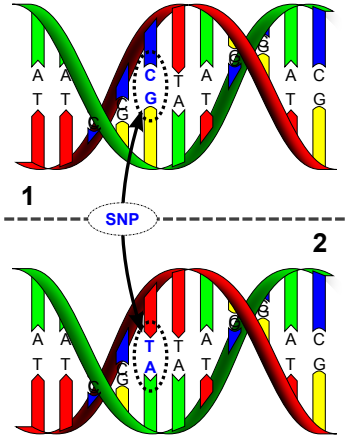
Christos Psarras, Henrik Barthels, **Paolo Bientinesi**

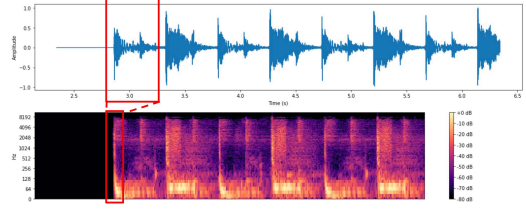
RWTH Aachen University **Umeå University**

April 22, 2020

Uppsala University (via Zoom)

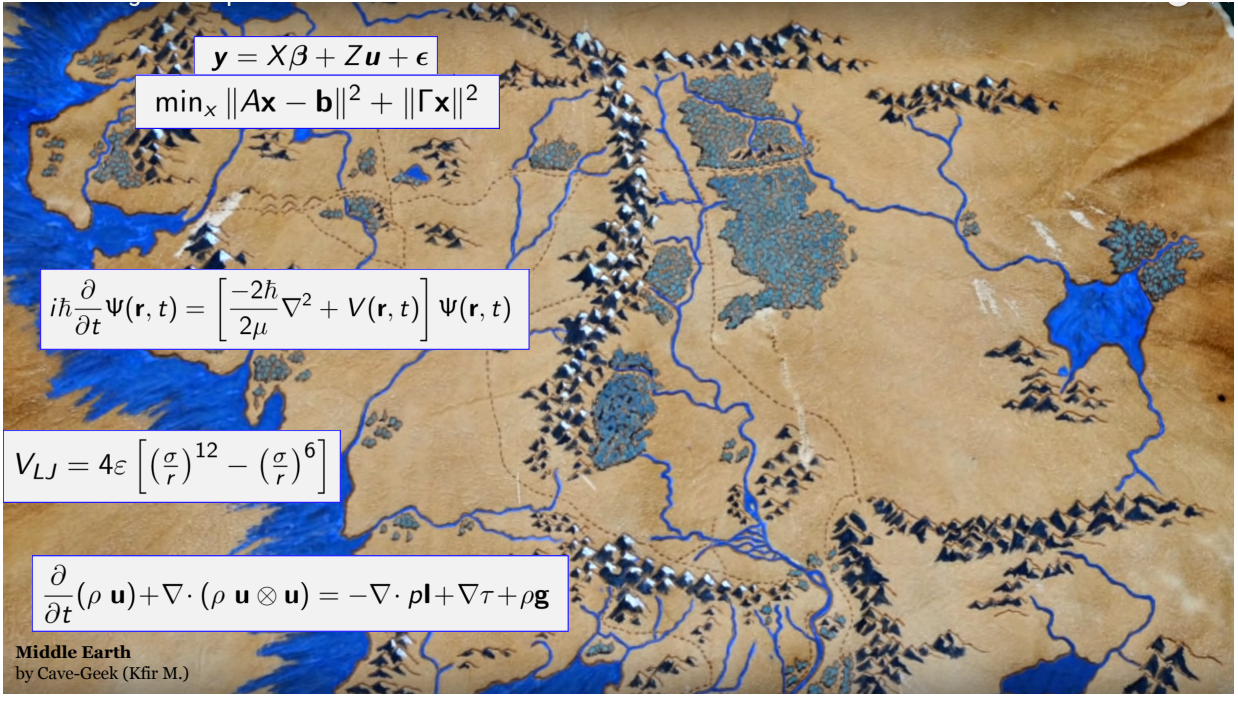
High-Performance & Scientific Computing





The computing world

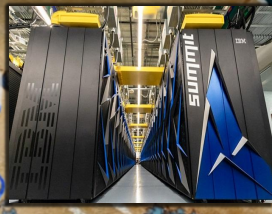
Middle Earth
by Cave-Geek (Kfir M.)



Middle Earth
by Cave-Geek (Kfir M.)

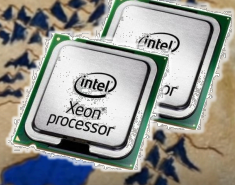
$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{u} + \boldsymbol{\epsilon}$$

$$\min_x \|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2$$



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

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



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

Middle Earth
by Cave-Geek (Kfir M.)

$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{u} + \boldsymbol{\epsilon}$$

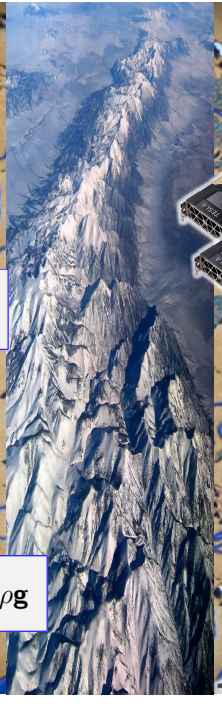
$$\min_x \|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2$$

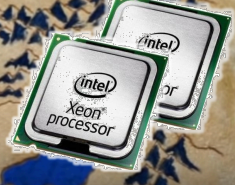
$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right]\Psi(\mathbf{r},t)$$

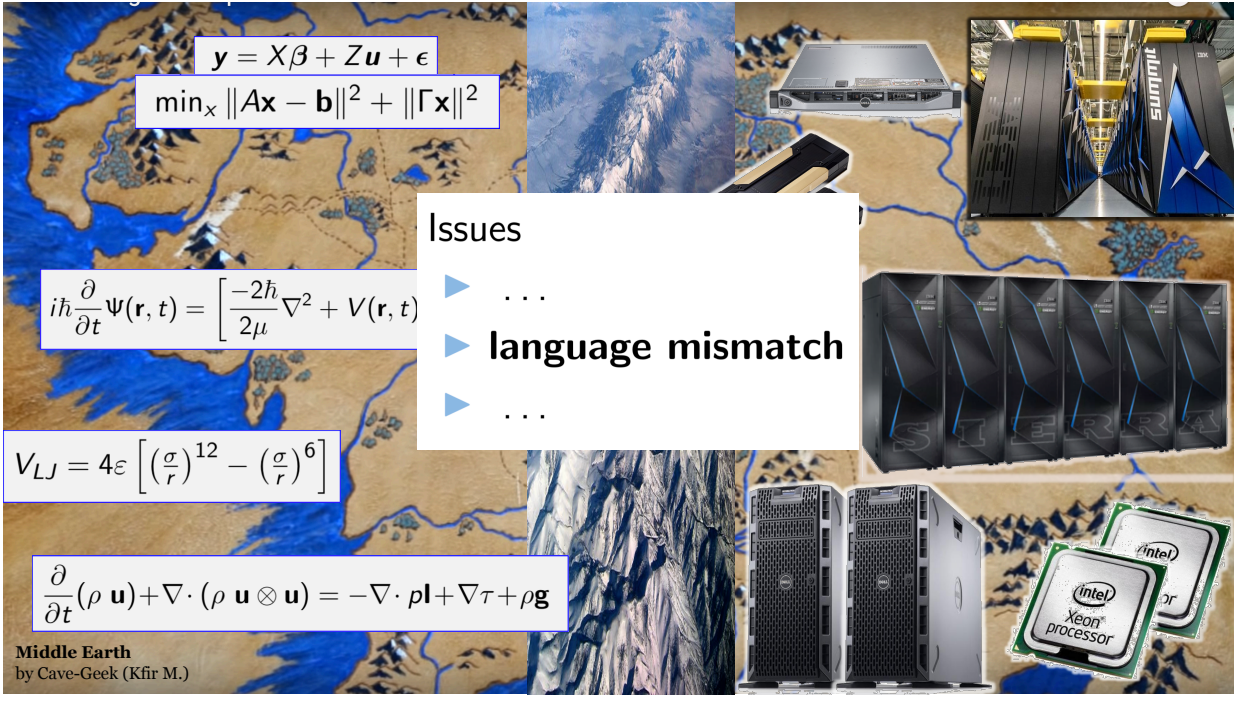
$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

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

Middle Earth
by Cave-Geek (Kfir M.)






$y = X\beta + Zu + \epsilon$
$\min_x \|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2$
$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \left[\frac{-2\hbar}{2\mu}\nabla^2 + V(\mathbf{r},t)\right.$
$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$
$\frac{\partial}{\partial t}(\rho\,\mathbf{u}) + \nabla\cdot(\rho\,\mathbf{u}\otimes\mathbf{u}) = -\nabla\cdot p\mathbf{I} + \nabla\tau + \rho\mathbf{g}$
Issues
...
language mismatch
...
Middle Earth
by Cave-Geek (Kfir M.)
Summit
SIERRA
intel
Xeon processor

This talk

matrix computations

- languages
- libraries
- compilers

MUL

VMUL

ADD

VADD

MOV

...

AND

...

FMADD

CMPSS

Middle Earth
by Cave-Geek (Kfir M.)

Matrix computations

Signal Processing	$x := \left(A^{-T}B^{T}BA^{-1} + R^{T}LR\right)^{-1} A^{-T}B^{T}BA^{-1}y$	$R \in \mathbb{R}^{n-1\times n}$, UT; $L \in \mathbb{R}^{n-1\times n-1}$, DI
Kalman Filter	$K_k := P_k^b H^T (HP_k^b H^T + R)^{-1};\ x_k^a := x_k^b + K_k(z_k - Hx_k^b);\ P_k^a := (I - K_K H)\, P_k^b$	
Ensemble Kalman Filter	$X^a := X^b + \left(B^{-1} + H^T R^{-1} H\right)^{-1} \left(Y - HX^b\right)$	$B \in \mathbb{R}^{N\times N}$ SSPD; $R \in \mathbb{R}^{m\times m}$, SSPD
Ensemble Kalman Filter	$\delta X := \left(B^{-1} + H^T R^{-1} H\right)^{-1} H^T R^{-1} \left(Y - HX^b\right)$	
Ensemble Kalman Filter	$\delta X := XV^T \left(R + HX(HX)^T\right)^{-1} \left(Y - HX^b\right)$	
Image Restoration	$x_k := (H^T H + \lambda\sigma^2 I_n)^{-1}(H^T y + \lambda\sigma^2(v_{k-1} - u_{k-1}))$	
Image Restoration	$H^\dagger := H^T(HH^T)^{-1};\ y_k := H^\dagger y + (I_n - H^\dagger H)x_k$	
Rand. Matrix Inversion	$X_{k+1} := S(S^TAS)^{-1}S^T + (I_n - S(S^TAS)^{-1}S^TA)X_k(I_n - AS(S^TAS)^{-1}S^T)$	
Rand. Matrix Inversion	$X_{k+1} := X_k + WA^TS(S^TAWA^TS)^{-1}S^T(I_n - AX_k)$	$W \in \mathbb{R}^{n\times n}$, SPD
Rand. Matrix Inversion	$X_{k+1} := X_k + (I_n - X_kA^T)S(S^TA^TWAS)^{-1}S^TA^TW$	
Rand. Matrix Inversion	$\Lambda := S(S^TAWAS)^{-1}S^T;\ \Theta := \Lambda AW;\ M_k := X_kA - I$ $X_{k+1} := X_k - M_k\Theta - (M_k\Theta)^T + \Theta^T(AX_kA - A)\Theta$	

Matrix computations (2)

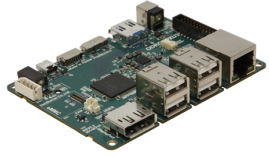
Generalized Least Squares	$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$	$n > m$; $M \in \mathbb{R}^{n \times n}$, SPD; $X \in \mathbb{R}^{n \times m}$; $y \in \mathbb{R}^{n \times 1}$
Stochastic Newton	$B_k := \frac{k}{k-1} B_{k-1}(I_n - A^T W_k((k-1)I_l + W_k^T A B_{k-1} A^T W_k)^{-1} W_k^T A B_{k-1})$	
Optimization	$x_f := WA^T(AWA^T)^{-1}(b - Ax)$; $x_o := W(A^T(AWA^T)^{-1}Ax - c)$	
Optimization	$x := W(A^T(AWA^T)^{-1}b - c)$	
Triangular Matrix Inv.	$X_{10} := L_{10}L_{00}^{-1}$; $X_{20} := L_{20} + L_{22}^{-1}L_{21}L_{11}^{-1}L_{10}$; $X_{11} := L_{11}^{-1}$; $X_{21} := -L_{22}^{-1}L_{21}$	
Tikhonov Regularization	$x := (A^T A + \Gamma^T \Gamma)^{-1} A^T b$	$A \in \mathbb{R}^{n \times m}$; $\Gamma \in \mathbb{R}^{m \times m}$; $b \in \mathbb{R}^{n \times 1}$
Tikhonov Regularization	$x := (A^T A + \alpha^2 I)^{-1} A^T b$	
Gen. Tikhonov Reg.	$x := (A^T PA + Q)^{-1}(A^T Pb + Qx_0)$	$P \in \mathbb{R}^{n \times n}$, SSPD; $Q \in \mathbb{R}^{m \times m}$, SSPD; $x_0 \in \mathbb{R}^{m \times 1}$
Gen. Tikhonov reg.	$x := x_0 + (A^T PA + Q)^{-1}(A^T P(b - Ax_0))$	
LMMSE estimator	$K_{t+1} := C_t A^T (AC_t A^T + C_z)^{-1}$; $x_{t+1} := x_t + K_{t+1}(y - Ax_t)$; $C_{t+1} := (I - K_{t+1}A)C_t$	
LMMSE estimator	$x_{\text{out}} = C_X A^T (AC_X A^T + C_Z)^{-1}(y - Ax) + x$	
LMMSE estimator	$x_{\text{out}} := (A^T C_Z^{-1} A + C_X^{-1})^{-1} A^T C_Z^{-1}(y - Ax) + x$	

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k (z_k - H x_k^b); \quad P_k^a := (I - K_K H)\, P_k^b$$

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

$$\Lambda := S(S^T A W A S)^{-1} S^T; \; \Theta := \Lambda A W; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$



MUL | ADD | MOV

MOVAPD

VFMADDPD ...

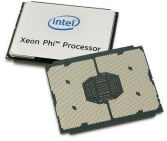
$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H)\, P_k^b$$

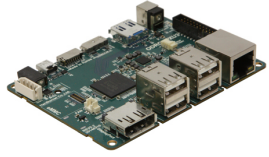
$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T A W A S)^{-1} S^T;\ \Theta := \Lambda A W;\ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A)\Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

?





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H)\, P_k^b$$

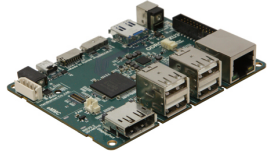
$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T A W A S)^{-1} S^T;\ \Theta := \Lambda A W;\ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

?

2-step solution



MUL | ADD | MOV

MOVAPD

VFMADDPD ...

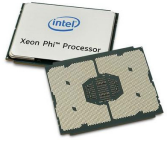
$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H)\, P_k^b$$

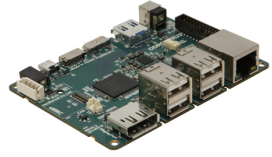
$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T A W A S)^{-1} S^T;\ \Theta := \Lambda A W;\ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A)\Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

building blocks (libraries)





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

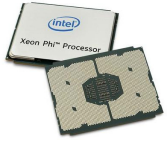
$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

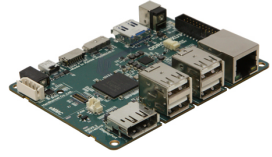
$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \ \Theta := \Lambda AW; \ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

building blocks (libraries)





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

back in the days

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

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

back in the days

- $\text{Cost}(\mathcal{A}lg) \equiv \#\text{operations}(\mathcal{A}lg)$
- Computers difficult to program
  Programs "easy" to optimize

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

back in the days

- Cost($\mathcal{A}lg$) $\equiv$ #operations($\mathcal{A}lg$)
- Computers difficult to program
  Programs "easy" to optimize
- **Libraries**: convenience, portability, separation of concerns

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

back in the days

- Cost($\mathcal{A}lg$) $\equiv$ #operations($\mathcal{A}lg$)
- Computers difficult to program
  Programs "easy" to optimize
- **Libraries**: convenience, portability, separation of concerns

since '90s

- Memory hierarchies
  Cost($\mathcal{A}lg$) $\not\equiv$ #operations($\mathcal{A}lg$)

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

back in the days

- Cost($\mathcal{A}lg$) $\equiv$ #operations($\mathcal{A}lg$)
- Computers difficult to program
  Programs "easy" to optimize
- **Libraries**: convenience, portability, separation of concerns

since '90s

- Memory hierarchies
  Cost($\mathcal{A}lg$) $\not\equiv$ #operations($\mathcal{A}lg$)
- Computers easy to program
  Programs difficult to optimize

Step 1: Libraries

EISPACK, LINPACK, BLAS, LAPACK, FFT, special funs, num.int., ..., NAG, MKL, ...

back in the days

- Cost($\mathcal{A}lg$) $\equiv$ #operations($\mathcal{A}lg$)
- Computers difficult to program
  Programs "easy" to optimize
- **Libraries**: convenience, portability, separation of concerns

since '90s

- Memory hierarchies
  Cost($\mathcal{A}lg$) $\not\equiv$ #operations($\mathcal{A}lg$)
- Computers easy to program
  Programs difficult to optimize
- **Libraries**: necessity (wrt performance)

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

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

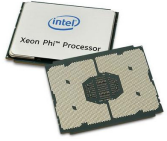
$$\Lambda := S(S^T A W A S)^{-1} S^T; \ \Theta := \Lambda A W; \ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

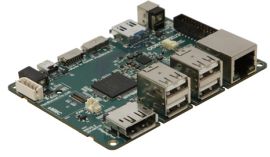
$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

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

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

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





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

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

$$\Lambda := S(S^T A W A S)^{-1} S^T; \ \Theta := \Lambda A W; \ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

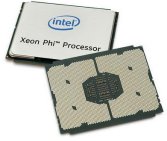
$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

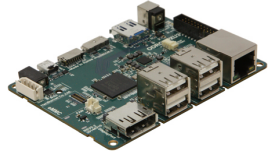
?

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

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

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





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

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

$$\Lambda := S(S^T A W A S)^{-1} S^T; \ \Theta := \Lambda A W; \ M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

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

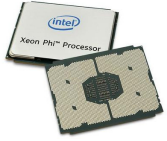
$$= Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

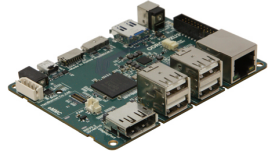
Step 2: "LAMP"
LINEAR ALGEBRA MAPPING PROBLEM

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

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

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





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

Linear Algebra Mapping Problem

Linear Algebra Mapping Problem

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

Linear Algebra Mapping Problem

- $\mathcal{E}$: a sequence of explicit assignments $var_i := EXP_i$
- $\mathcal{K}$: a set of available computational building blocks e.g., BLAS, LAPACK, ...

Linear Algebra Mapping Problem

- $\mathcal{E}$: a sequence of explicit assignments — $var_i := EXP_i$
- $\mathcal{K}$: a set of available computational building blocks — e.g., BLAS, LAPACK, ...
- $\mathcal{M}$: a cost function defined over $\mathcal{K}^+$ — FLOPs, data movement, stability, time

Linear Algebra Mapping Problem

- $\mathcal{E}$: a sequence of explicit assignments — $var_i := EXP_i$
- $\mathcal{K}$: a set of available computational building blocks — e.g., BLAS, LAPACK, ...
- $\mathcal{M}$: a cost function defined over $\mathcal{K}^+$ — FLOPs, data movement, stability, time

LAMP:

Find a sequence of calls to building blocks in $\mathcal{K}$, optimal according to $\mathcal{M}$, that computes all the assignments in $\mathcal{E}$.

Linear Algebra Mapping Problem

- $\mathcal{E}$: a sequence of explicit assignments — $var_i := EXP_i$
- $\mathcal{K}$: a set of available computational building blocks — e.g., BLAS, LAPACK, ...
- $\mathcal{M}$: a cost function defined over $\mathcal{K}^+$ — FLOPs, data movement, stability, time

LAMP:
Find a sequence of calls to building blocks in $\mathcal{K}$, optimal according to $\mathcal{M}$, that computes all the assignments in $\mathcal{E}$.

- Suboptimal solution — easy
- Optimality — NP complete — reduction from Ensemble Computation

Problem acknowledged, yet overlooked

"How do users go about solving their matrix computations?"

Problem acknowledged, yet overlooked

"How do users go about solving their matrix computations?"

1) By hand. C/Fortran. Need for expertize and patience.

computer efficiency! ... but human productivity?

Problem acknowledged, yet overlooked

"How do users go about solving their matrix computations?"

1) By hand. C/Fortran. Need for expertize and patience.

   computer efficiency! ... but human productivity?

2) Via a high-level language. Matlab, Julia, R, Armadillo (C++), NumPy, ...

   human productivity! ... but computer efficiency?

Problem acknowledged, yet overlooked

"How do users go about solving their matrix computations?"

1) By hand. C/Fortran. Need for expertize and patience.

computer efficiency! ... but human productivity?

2) Via a high-level language. Matlab, Julia, R, Armadillo (C++), NumPy, ...

human productivity! ... but computer efficiency?

Slow solutions → "2-language problem"

Investigation: How well do high-level languages solve LAMPs?

Investigation: How well do high-level languages solve LAMPs?

Seven languages/libraries. Not an exhaustive survey.

Main criterion: high level of **abstraction**.

Investigation: How well do high-level languages solve LAMPs?

Seven languages/libraries. Not an exhaustive survey.

Main criterion: high level of **abstraction**. Also: popularity, expressiveness, (performance...)

Investigation: How well do high-level languages solve LAMPs?

Seven languages/libraries. Not an exhaustive survey.

Main criterion: high level of **abstraction**. Also: popularity, expressiveness, (performance...)

```
C = A * B'+ B * A' + C;                          // Matlab, Octave

C = A * transpose(B) + B * transpose(A) + C      // Julia

C = A * trans(B) + B * trans(A) + C;             // Armadillo

C = A * B.transpose() + B * A.transpose() + C;   // Eigen

ct = at @ bt.T + bt @ at.T + ct                  // NumPy

ct <- at %*% t(bt) + bt %*% t(at) + ct           // R
```

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$C = AB$								

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$C = AB$	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$C = AB$	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27
`GEMM`	✓	✓	✓	✓	✓	✓	✓	

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
C = AB	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27
`GEMM`	✓	✓	✓	✓	✓	✓	✓	

C = C+AA'

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
C = AB	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27
`GEMM`	✓	✓	✓	✓	✓	✓	✓	
C = C+AA'	0.18	0.17	0.21	0.32	0.29	0.17	0.18	0.14

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
C = AB	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27
`GEMM`	✓	✓	✓	✓	✓	✓	✓	
C = C+AA'	0.18	0.17	0.21	0.32	0.29	0.17	0.18	0.14
`SYRK`	✓	✓	✓	×	×	✓	✓	

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
C = AB	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27
`GEMM`	✓	✓	✓	✓	✓	✓	✓	
C = C+AA'	0.18	0.17	0.21	0.32	0.29	0.17	0.18	0.14
`SYRK`	✓	✓	✓	×	×	✓	✓	
C = C+AB'+BA'	0.57	0.59	0.69	0.59	0.58	0.57	0.58	0.28

Do they map? matrix products

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
C = AB	0.29	0.28	0.30	0.31	0.29	0.29	0.29	0.27
`GEMM`	✓	✓	✓	✓	✓	✓	✓	
C = C+AA'	0.18	0.17	0.21	0.32	0.29	0.17	0.18	0.14
`SYRK`	✓	✓	✓	×	×	✓	✓	
C = C+AB'+BA'	0.57	0.59	0.69	0.59	0.58	0.57	0.58	0.28
`SYR2K`	×	×	×	×	×	×	×	

Do they map? $Ax = b \quad \equiv \quad x := A\backslash b \quad \not\equiv \quad \text{inv}(A) * b$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$x := A\backslash b$								

Do they map? $Ax = b \quad \equiv \quad x := A\backslash b \quad \not\equiv \quad \text{inv}(A) * b$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$x := A\backslash b$	0.71	0.72	0.63	0.68	0.64	0.63	0.72	0.61

Do they map? $Ax = b \quad \equiv \quad x := A\backslash b \quad \not\equiv \quad \text{inv}(A) * b$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$x := A\backslash b$	0.71	0.72	0.63	0.68	0.64	0.63	0.72	0.61
$\text{inv}(A) * b$	1.76	1.82	1.69	2.20	2.21	**0.63**	2.49	1.71

Do they map? $Ax = b \quad \equiv \quad x := A\backslash b \quad \not\equiv \quad \text{inv}(A) * b$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$x := A\backslash b$	0.71	0.72	0.63	0.68	0.64	0.63	0.72	0.61
$\text{inv}(A) * b$	1.76	1.82	1.69	2.20	2.21	**0.63**	2.49	1.71
LinSolve	-	-	-	-	-	✓	-	-

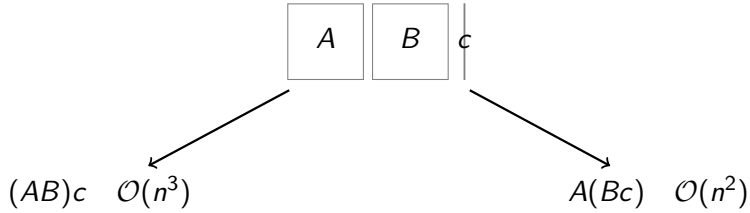
but ...

Do they map? $Ax = b \quad \equiv \quad x := A\backslash b \quad \not\equiv \quad \text{inv}(A) * b$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
$x := A\backslash b$	0.71	0.72	0.63	0.68	0.64	0.63	0.72	0.61
$\text{inv}(A) * b$	1.76	1.82	1.69	2.20	2.21	**0.63**	2.49	1.71
LinSolve	-	-	-	-	-	✓	-	-

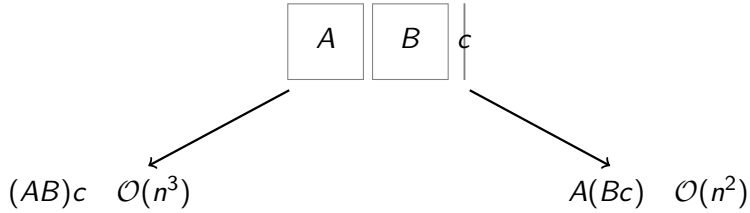
but . . . should they map?

Optimal parenthesisation



Product is associative, but its cost is not

Optimal parenthesisation



Product is associative, but its cost is not

Matrix Chain Algorithm $O(k \log k)$ Hu & Shing 1982; $O(k^3)$ dynamic programming

Matrix Chain?

Chain	**Optimal Evaluation**
1) "left-to-right" (LtR)	((A B) C)
2) "right-to-left" (RtL)	(A (B C))
3) "mixed" (Mix)	((A B) (C D))

Matrix Chain?

Chain	**Optimal Evaluation**
1) "left-to-right" (LtR)	((A B) C)
2) "right-to-left" (RtL)	(A (B C))
3) "mixed" (Mix)	((A B) (C D))

		Matlab	Octave	Julia	R	Eigen	Armad.	NumPy
1)	LtR no par.	0.056	0.056	0.055	0.061	0.058	0.056	0.055
	LtR guided	0.056	0.056	0.055	0.061	0.058	0.056	0.055

Matrix Chain?

Chain	**Optimal Evaluation**
1) "left-to-right" (LtR)	((A B) C)
2) "right-to-left" (RtL)	(A (B C))
3) "mixed" (Mix)	((A B) (C D))

		Matlab	Octave	Julia	R	Eigen	Armad.	NumPy
1)	LtR no par.	0.056	0.056	0.055	0.061	0.058	0.056	0.055
	LtR guided	0.056	0.056	0.055	0.061	0.058	0.056	0.055
2)	RtL no par.	0.42	0.43	0.42	0.44	0.42	**0.055**	0.42
	RtL guided	0.055	0.056	0.054	0.059	0.056	0.055	0.056

Matrix Chain?

Chain	**Optimal Evaluation**
1) "left-to-right" (LtR)	((A B) C)
2) "right-to-left" (RtL)	(A (B C))
3) "mixed" (Mix)	((A B) (C D))

		Matlab	Octave	Julia	R	Eigen	Armad.	NumPy
1)	LtR no par.	0.056	0.056	0.055	0.061	0.058	0.056	0.055
	LtR guided	0.056	0.056	0.055	0.061	0.058	0.056	0.055
2)	RtL no par.	0.42	0.43	0.42	0.44	0.42	**0.055**	0.42
	RtL guided	0.055	0.056	0.054	0.059	0.056	0.055	0.056
3)	Mix no par.	0.32	0.33	0.33	0.33	0.35	0.31	0.33
	Mix guided	0.21	0.22	0.22	0.22	0.23	0.20	0.22

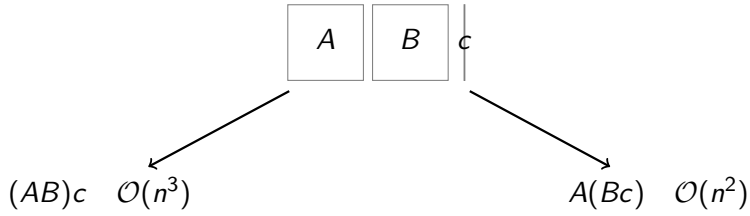
Matrix Chain?

Chain	**Optimal Evaluation**
1) "left-to-right" (LtR)	((A B) C)
2) "right-to-left" (RtL)	(A (B C))
3) "mixed" (Mix)	((A B) (C D))

		Matlab	Octave	Julia	R	Eigen	Armad.	NumPy
1)	LtR no par.	0.056	0.056	0.055	0.061	0.058	0.056	0.055
	LtR guided	0.056	0.056	0.055	0.061	0.058	0.056	0.055
2)	RtL no par.	0.42	0.43	0.42	0.44	0.42	**0.055**	0.42
	RtL guided	0.055	0.056	0.054	0.059	0.056	0.055	0.056
3)	Mix no par.	0.32	0.33	0.33	0.33	0.35	0.31	0.33
	Mix guided	0.21	0.22	0.22	0.22	0.23	0.20	0.22
	Matrix chains	×	×	×	×	×	≈	×

Challenge: Generalized Matrix Chain

[arXiv:1804.04021]



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., $A \rightarrow Q^T D Q$, $A \rightarrow LU$)
- Properties & specialized kernels (`GEMM`, `TRMM`, `SYMM`, ...)

Challenge: Not all flops were created equal

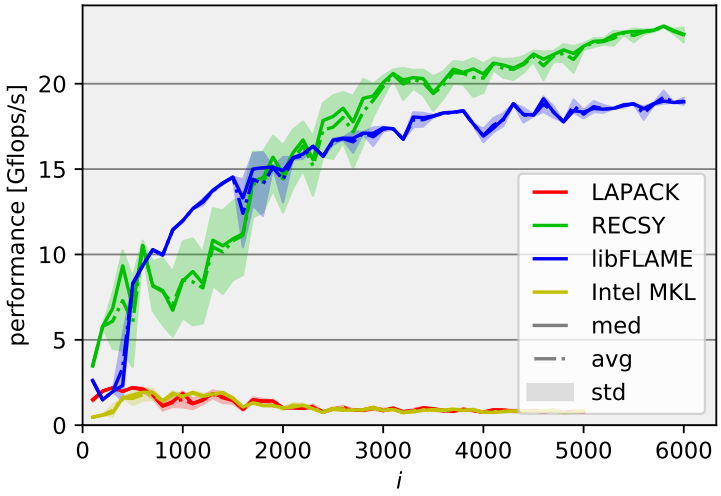
$$\#\text{FLOPs} \quad \text{vs.} \quad \text{execution time} \qquad (\text{vs. numerical stability})$$

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

Challenge: Not all flops were created equal

[arXiv:1504.08035]

Triangular Sylvester equation – all libraries perform the same # of flops



Challenge: Not all flops were created equal

[arXiv:1706.01341]

#FLOPs vs. execution time (vs. numerical stability)

$$\operatorname*{argmin}_{\mathcal{A}} \left(\text{FLOPs}(\mathcal{A}) \right) \neq \operatorname*{argmin}_{\mathcal{A}} \left(\text{time}(\mathcal{A}) \right)$$

$\Rightarrow$ **Performance prediction**: efficiency

Challenge: Not all flops were created equal [arXiv:1706.01341]

#FLOPs vs. execution time (vs. numerical stability)

$$\operatorname*{argmin}_{\mathcal{A}} \left(\text{FLOPs}(\mathcal{A}) \right) \neq \operatorname*{argmin}_{\mathcal{A}} \left(\text{time}(\mathcal{A}) \right)$$

$\Rightarrow$ **Performance prediction**: efficiency

Big Challenge: **Parallelism**

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

Properties?

Operation	Property	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
Linear System	-	0.71	0.74	0.62	0.67	0.63	0.62	0.65	0.61

Properties?

Operation	Property	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
Linear System	-	0.71	0.74	0.62	0.67	0.63	0.62	0.65	0.61
	Symmetric	0.71	0.73	0.62	0.69	N/A	0.62	0.65	0.46

Properties?

Operation	Property	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
Linear System	-	0.71	0.74	0.62	0.67	0.63	0.62	0.65	0.61
	Symmetric	0.71	0.73	0.62	0.69	N/A	0.62	0.65	0.46
	SPD	**0.41**	**0.40**	0.60	0.63	N/A	**0.34**	0.62	0.31

Properties?

Operation	Property	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
Linear System	-	0.71	0.74	0.62	0.67	0.63	0.62	0.65	0.61
	Symmetric	0.71	0.73	0.62	0.69	N/A	0.62	0.65	0.46
	SPD	**0.41**	**0.40**	0.60	0.63	N/A	**0.34**	0.62	0.31
	Triangular	**0.03**	**0.04**	**0.03**	0.63	N/A	0.62	0.65	0.03

Properties?

Operation	Property	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
Linear System	-	0.71	0.74	0.62	0.67	0.63	0.62	0.65	0.61
	Symmetric	0.71	0.73	0.62	0.69	N/A	0.62	0.65	0.46
	SPD	**0.41**	**0.40**	0.60	0.63	N/A	**0.34**	0.62	0.31
	Triangular	**0.03**	**0.04**	**0.03**	0.63	N/A	0.62	0.65	0.03
	Diagonal	0.03	0.05	**0.01**	0.63	N/A	**0.03**	0.62	0.001
		$\approx$	$\approx$	$\approx$	$\times$	$\times$	$\approx$	$\times$	

Properties?

Operation	Property	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy	C
Linear System	-	0.71	0.74	0.62	0.67	0.63	0.62	0.65	0.61
	Symmetric	0.71	0.73	0.62	0.69	N/A	0.62	0.65	0.46
	SPD	**0.41**	**0.40**	0.60	0.63	N/A	**0.34**	0.62	0.31
	Triangular	**0.03**	**0.04**	**0.03**	0.63	N/A	0.62	0.65	0.03
	Diagonal	0.03	0.05	**0.01**	0.63	N/A	**0.03**	0.62	0.001
		$\approx$	$\approx$	$\approx$	$\times$	$\times$	$\approx$	$\times$	
Multiplication	-	1.44	1.48	1.47	1.47	1.45	1.44	1.44	1.46
	Triangular	1.44	1.48	**0.75**	1.47	1.45	1.44	1.44	0.74
	Diagonal	1.44	1.48	**0.03**	1.47	1.45	1.42	1.44	0.06
		$\times$	$\times$	✓	$\times$	$\times$	$\times$	$\times$	

Challenge: Inference of properties

- **easy** $E := L_1 * U^T * L_2$ triangular(E) ?

Challenge: Inference of properties

- **easy** $E := L_1 * U^T * L_2$ triangular(E) ?
- **hard** $\lambda(L^{-T}AL^{-1})$ symmetric($L^{-T}AL^{-1}$) ?

Challenge: Inference of properties

- **easy** $E := L_1 * U^T * L_2$ triangular(E) ?
- **hard** $\lambda(L^{-T}AL^{-1})$ symmetric($L^{-T}AL^{-1}$) ?
- **impossible?** $E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T$ properties($I + U^TQ^{-1}U$) ?

Challenge: Inference of properties

- **easy** $E := L_1 * U^T * L_2$ triangular(E) ?
- **hard** $\lambda(L^{-T}AL^{-1})$ symmetric($L^{-T}AL^{-1}$) ?
- **impossible?** $E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T$ properties($I + U^TQ^{-1}U$) ?

$\Rightarrow$ **Symbolic analysis**: pattern matching

Challenge: Common Subexpressions

[arXiv:1710.06915]

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

Challenge: Common Subexpressions

[arXiv:1710.06915]

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

BUT

$$X := ABABv \quad \not\rightarrow \quad \begin{cases} Z := AB \\ X := ZZv \end{cases}$$

Common Subexpressions?

$$\begin{cases} X := AB \\ Y := AB \end{cases} \quad \rightarrow \quad \begin{cases} X := AB \\ Y := X \end{cases}$$

Common Subexpressions?

$$\begin{cases} X := AB \\ Y := AB \end{cases} \quad \rightarrow \quad \begin{cases} X := AB \\ Y := X \end{cases}$$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy
copy	0.27	0.31	0.36	0.30	0.30	0.26	0.30

Common Subexpressions?

$$\begin{cases} X := AB \\ Y := AB \end{cases} \quad \rightarrow \quad \begin{cases} X := AB \\ Y := X \end{cases}$$

	Matlab	Octave	Julia	R	Eigen	Armad.	NumPy
copy	0.27	0.31	0.36	0.30	0.30	0.26	0.30
direct	0.54	0.6	0.61	0.56	0.58	0.52	0.55
	×	×	×	×	×	×	×

Other features (challenges)

- Code motion

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow$?

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow$?

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow$?

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}$$

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow$?

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad \rightarrow ? \quad \begin{cases} M_1 x_T = y_T \\ M_2 x_B = y_B \end{cases}, \quad \begin{cases} y_T := M_1 x_T \\ y_B := M_2 x_B \end{cases}$$

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow ?$

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad \rightarrow ? \quad \begin{cases} M_1 x_T = y_T \\ M_2 x_B = y_B \end{cases}, \quad \begin{cases} y_T := M_1 x_T \\ y_B := M_2 x_B \end{cases} \quad \times, \times$$

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow ?$

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad \rightarrow ? \quad \begin{cases} M_1 x_T = y_T \\ M_2 x_B = y_B \end{cases}, \quad \begin{cases} y_T := M_1 x_T \\ y_B := M_2 x_B \end{cases} \quad \times, \times$$

- $\mathrm{diag}(A + B)$ vs. $\mathrm{diag}(A) + \mathrm{diag}(B)$

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow ?$

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad \rightarrow ? \quad \begin{cases} M_1 x_T = y_T \\ M_2 x_B = y_B \end{cases}, \quad \begin{cases} y_T := M_1 x_T \\ y_B := M_2 x_B \end{cases} \quad \times, \times$$

- $\mathrm{diag}(A + B)$ vs. $\mathrm{diag}(A) + \mathrm{diag}(B)$ $\rightarrow$ Armadillo

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow$?

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad \rightarrow ? \quad \begin{cases} M_1 x_T = y_T \\ M_2 x_B = y_B \end{cases}, \quad \begin{cases} y_T := M_1 x_T \\ y_B := M_2 x_B \end{cases} \quad \times, \times$$

- $\mathrm{diag}(A + B)$ vs. $\mathrm{diag}(A) + \mathrm{diag}(B)$ $\rightarrow$ Armadillo

  $\mathrm{diag}(AB)$ vs. ...

Other features (challenges)

- Code motion

```
for i = 1:n,
    X = A*B;
    d[i] = C[i,i];
end
```

$\rightarrow$?

```
X = A*B;
for i = 1:n,
    d[i] = C[i,i];
end
```

$\times$

- Blocked operands

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad \rightarrow ? \quad \begin{cases} M_1 x_T = y_T \\ M_2 x_B = y_B \end{cases}, \quad \begin{cases} y_T := M_1 x_T \\ y_B := M_2 x_B \end{cases} \quad \times, \times$$

- $\mathrm{diag}(A + B)$ vs. $\mathrm{diag}(A) + \mathrm{diag}(B)$ $\rightarrow$ Armadillo

  $\mathrm{diag}(AB)$ vs. ... $\rightarrow$ $\times$

Summary

- This evaluation is **NOT** a ranking of languages [arXiv:1911.09421]

Summary

- This evaluation is **NOT** a ranking of languages [arXiv:1911.09421]
- We expose optimizations that arise frequently in the solution of LAMPs
- Compilers & languages are great with scalars, not so much with matrices

Summary

- This evaluation is **NOT** a ranking of languages [arXiv:1911.09421]
- We expose optimizations that arise frequently in the solution of LAMPs
- Compilers & languages are great with scalars, not so much with matrices
- LAMPs are challenging — the optimal solution requires lots of interdisciplinary expertise

Summary

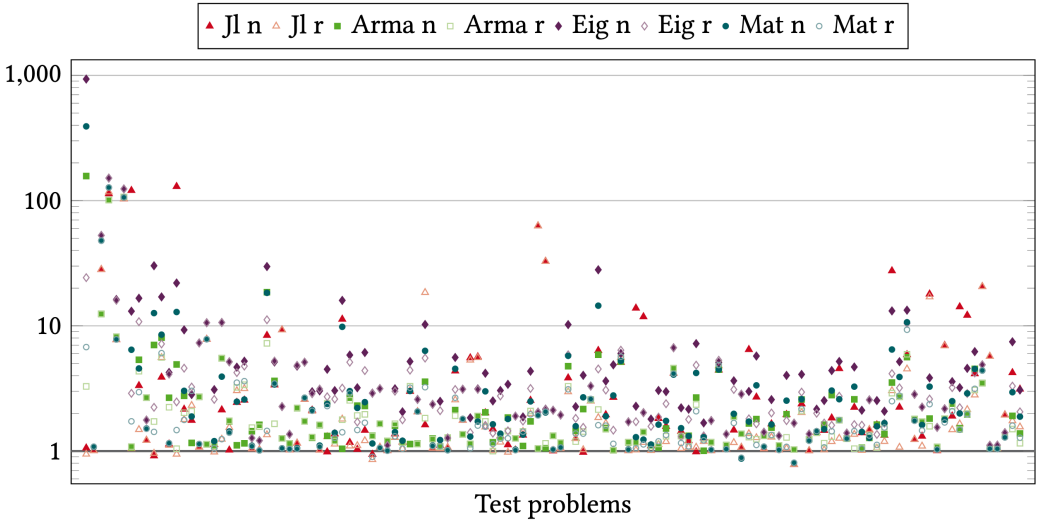
- This evaluation is **NOT** a ranking of languages [arXiv:1911.09421]
- We expose optimizations that arise frequently in the solution of LAMPs
- Compilers & languages are great with scalars, not so much with matrices
- LAMPs are challenging — the optimal solution requires lots of interdisciplinary expertise
- **Linnea**: A linear algebra compiler [arXiv:1912.12924]

Summary

- This evaluation is **NOT** a ranking of languages [arXiv:1911.09421]
- We expose optimizations that arise frequently in the solution of LAMPs
- Compilers & languages are great with scalars, not so much with matrices
- LAMPs are challenging — the optimal solution requires lots of interdisciplinary expertise
- **Linnea**: A linear algebra compiler [arXiv:1912.12924]

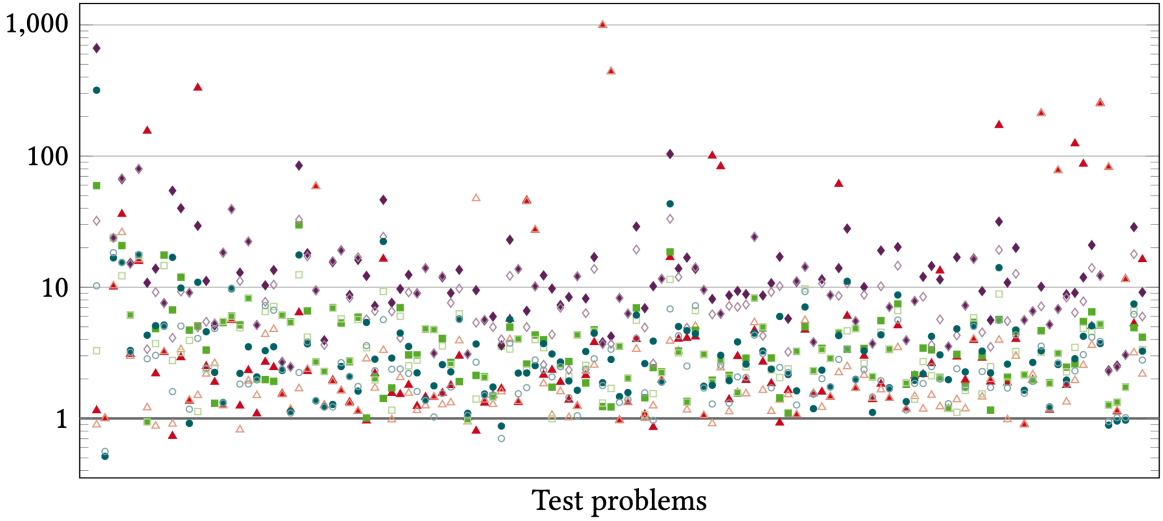
THANK YOU

Linnea speedup: 1 thread



Jl: Julia, **Arma**: Armadillo, **Eig**: Eigen, **Mat**: Matlab. **n**/**r**: naive/recommended implementation

Linnea speedup: 24 threads



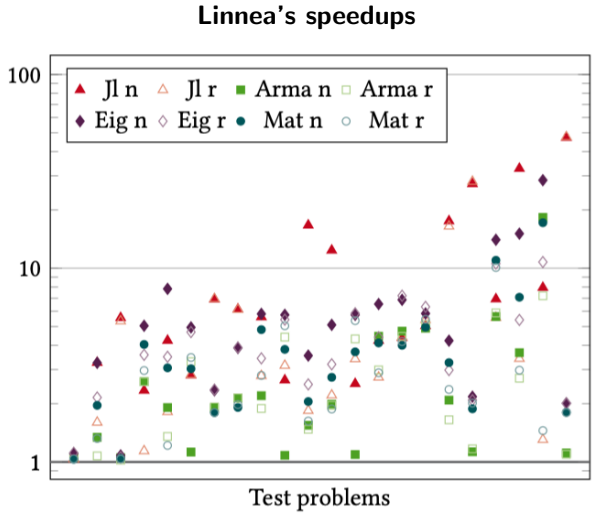
Jl: Julia, **Arma**: Armadillo, **Eig**: Eigen, **Mat**: Matlab. **n**/**r**: naive/recommended implementation

Linnea

Linear algebra knowledge: operators, identities, properties, theorems

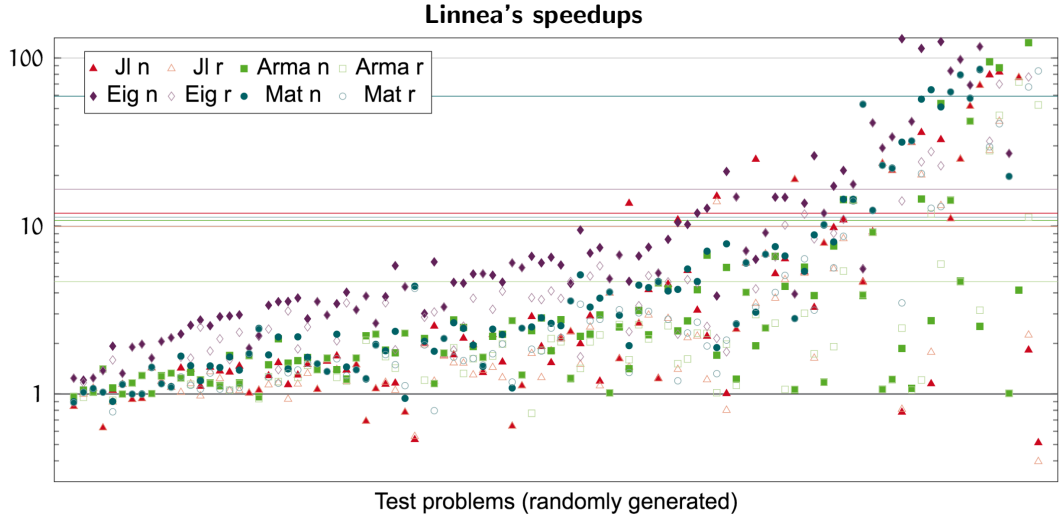
- Distributivity, commutativity, partitionings, ...
- $((QR)^T QR)^{-1}(QR)^T y \;\rightarrow\; (R^T Q^T QR)^{-1} R^T Q^T y \;\rightarrow\; R^{-1}R^{-T}R^T Q^T y \;\rightarrow\; R^{-1}Q^T y$
- $\mathrm{SPD}(A) \rightarrow \mathrm{SPD}(A_{BR} - A_{BL}A_{TL}^{-1}A_{BL}^T)$ Schur complement
- ...

Linnea: Results for real applications



Jl: Julia, **Arma**: Armadillo, **Eig**: Eigen, **Mat**: Matlab. **n**/**r**: naive/recommended implementation

Linnea: Results for random expressions



Jl: Julia, **Arma**: Armadillo, **Eig**: Eigen, **Mat**: Matlab. **n**/**r**: naive/recommended implementation

LAMPs are everywhere

- Kernels

  $\mathcal{E}$: $\{C := A * B + C\}$ $\mathcal{K}$: processor's ISA $\mathcal{M}$: execution time

LAMPs are everywhere

- Kernels

  $\mathcal{E}$: $\{C := A * B + C\}$ $\quad$ $\mathcal{K}$: processor's ISA $\quad$ $\mathcal{M}$: execution time

- Automatic code generation

  ATLAS [Whaley 2001], FLAME [Gunnels 2001], Build-To-Order BLAS [Siek 2008], LGEN [Spampinato 2014], ...

LAMPs are everywhere

- Kernels

  $\mathcal{E}$: $\{C := A * B + C\}$ $\mathcal{K}$: processor's ISA $\mathcal{M}$: execution time

- Automatic code generation

  ATLAS [Whaley 2001], FLAME [Gunnels 2001], Build-To-Order BLAS [Siek 2008], LGEN [Spampinato 2014], ...

- "Matrix Chain Problem"

  $\mathcal{E}$: $\{X := M_1 M_2 \cdots M_k\}$ $\mathcal{K}$: $\{C := A * B\}$ $\mathcal{M}$: #flops

LAMPs are everywhere

- Kernels

  $\mathcal{E}$: $\{C := A * B + C\}$ $\mathcal{K}$: processor's ISA $\mathcal{M}$: execution time

- Automatic code generation

  ATLAS [Whaley 2001], FLAME [Gunnels 2001], Build-To-Order BLAS [Siek 2008], LGEN [Spampinato 2014], ...

- "Matrix Chain Problem"

  $\mathcal{E}$: $\{X := M_1 M_2 \cdots M_k\}$ $\mathcal{K}$: $\{C := A * B\}$ $\mathcal{M}$: #flops

- Applications

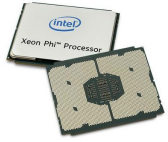
  $\mathcal{E}$: explicit assignments $\mathcal{K}$: libraries $\mathcal{M}$: execution time + ...

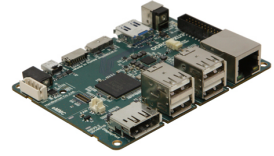
Quantum Chemistry | High-order FEM | DG | ... | Tensor App #N

Transposition | Contraction | ... | Alternating LS

Khatri-Rao | SpMTTKRP | ... | TTV, TTM

HPTT **TCL** **...** **BLAS**





MUL | ADD | MOV

MOVAPD

VFMADDPD ...

Linnea: Example

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

Naive

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

Recommended

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

Expert

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

Linnea: Example

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

Naive

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

Recommended

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

Expert

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

Generated

```
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
```