High-Performance & Automatic Computing

Paolo Bientinesi
pauldj@cs.umu.se

December 5, 2018
Umeå Universitet

High Performance and
Automatic Computing

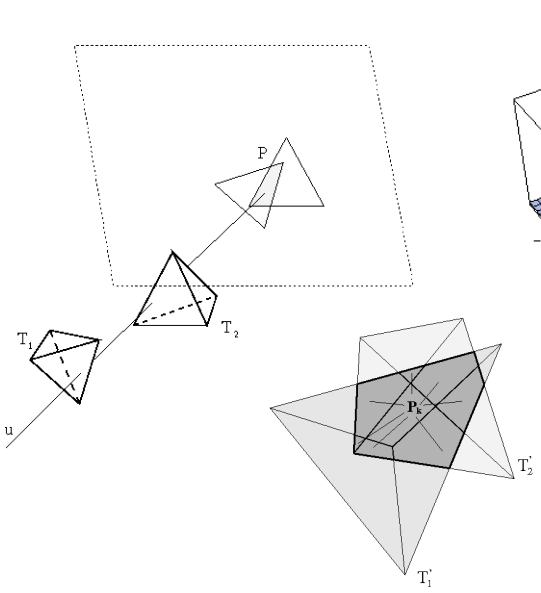
- About me
- Molecular dynamics
- Mixed precision calculations
- Accuracy & performance modeling
- Linear Algebra, applications
- Tensor operations

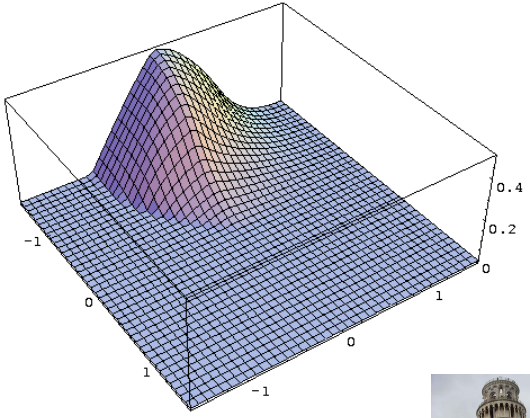
About me: Italy

Tuscany

University of Pisa

Thesis: "Computational Geometry Techniques for Approximating Electrostatic Forces"





USA


C A N A D A
MEXICO
GULF OF MEXICO
CUBA
POPULATED PLACES
Chicago
Seattle
Dallas
Miami
Washington
Atlanta
TRANSPORTATION
PHYSICAL FEATURES

University of Texas at Austin

Dissertation: "Mechanical Derivation and Systematic Analysis of Correct Linear Algebra Algorithms"

Algorithm $LU = A$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right), U \rightarrow \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right)$

where $A_{TL}, L_{TL}, U_{TL},$ are 0×0

While $m(A_{TL}) \leq m(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & 1 & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right),$$

$$\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} U_{00} & u_{01} & U_{02} \\ \hline 0 & \upsilon_{11} & u_{12}^T \\ \hline 0 & 0 & U_{22} \end{array}\right)$$

where $\alpha_{11}, 1, \upsilon_{11}$ are scalars

$\upsilon_{11} := \alpha_{11} - l_{10}^T u_{01}$

$u_{12}^T := a_{12}^T - l_{10}^T U_{02}$

$l_{21} := (a_{21} - L_{20} u_{01}) / \upsilon_{11}$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & 1 & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \cdots$$

endwhile

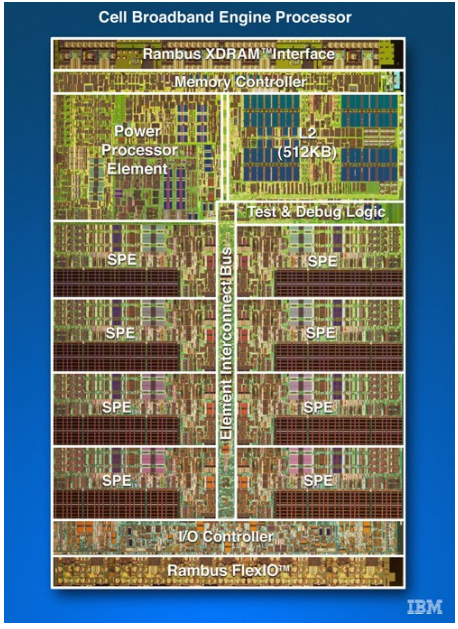
- Automatic backward analysis?

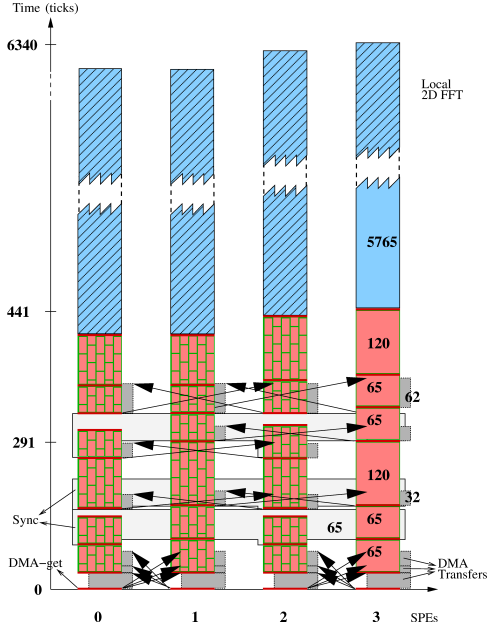
$v_{11} := \alpha_{11} - l_{10}^T u_{01}$ $u_{12}^T := a_{12}^T - l_{10}^T U_{02}$ $l_{21} := (a_{21} - L_{20} u_{01})/v_{11}$	$\check{v}_{11} + \delta\alpha_{11} = \alpha_{11} - l_{10}^T u_{01}$ $\wedge\ \lvert\delta\alpha_{11}\rvert \leq \gamma_{k+1}(\lvert l_{10}^T\rvert\lvert u_{01}\rvert + \lvert\check{v}_{11}\rvert)$ Th. 3.9 R2-F $\check{u}_{12}^T + \delta\alpha_{12}^T = a_{12}^T - l_{10}^T U_{02}$ $\wedge\ \lvert\delta\alpha_{12}^T\rvert \leq \gamma_{k+1}(\lvert l_{10}^T\rvert\lvert U_{02}\rvert + \lvert\check{u}_{12}^T\rvert)$ Th. 5.1 R2-F $\check{l}_{21} v_{11} + \delta\alpha_{21} = a_{21} - L_{20} u_{01}$ $\wedge\ \lvert\delta\alpha_{21}\rvert \leq \gamma_{k+1}(\lvert\check{L}_{20}\rvert\lvert\check{u}_{01}\rvert + \lvert\check{l}_{21}\rvert\lvert v_{11}\rvert)$ Th. 5.1 R4-F

- Symmetric eigenproblem $AX = X\Lambda$

  Multiple Relatively Robust Representations (MRRR)

Duke University – Cell Processor & FFTs





Germany – RWTH Aachen University

The world of scientific computing

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

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.

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

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

Linear mixed models

$\vdots$

The world of scientific computing

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

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.

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

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

Linear mixed models

$\vdots$

The world of scientific computing

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

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.

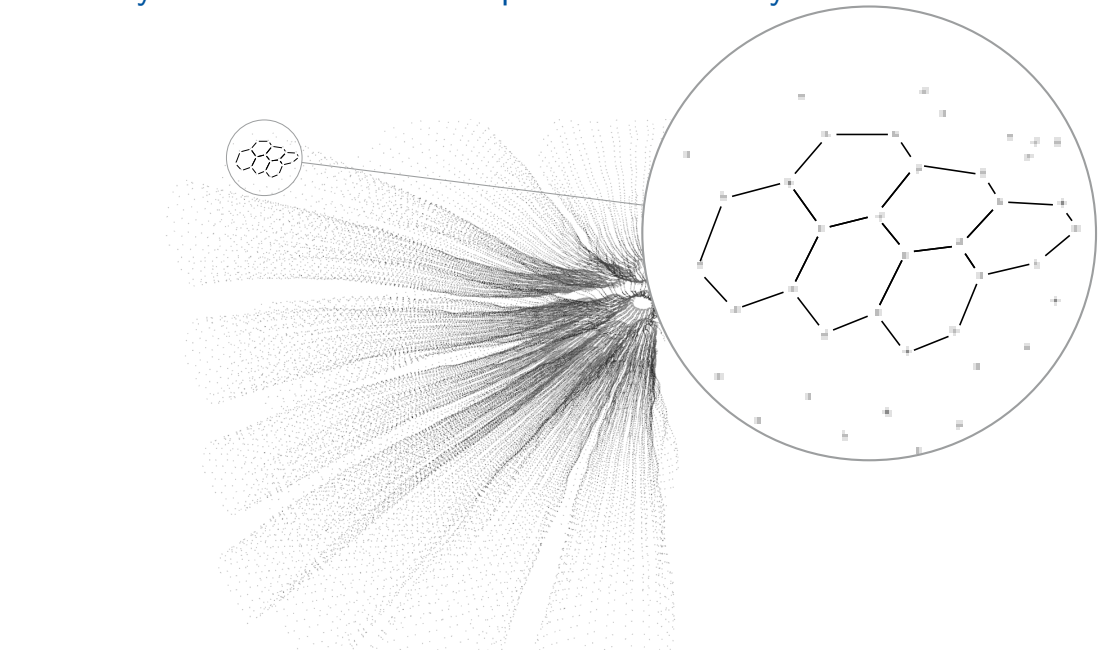
$$\boldsymbol{y} = X\boldsymbol{\beta} + Z\boldsymbol{u} + \boldsymbol{\epsilon}$$

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

Linear mixed models

$\vdots$

Domain Specific Languages, Compilers, Libraries



Pair vs Many-body potentials

$$V = \sum_i \sum_j V(i,j)$$

$$F_i = -\nabla_{x_i} V$$

```
for i in atoms:
    for j in neighbors(i):
        V += V(i, j)
        F[i] -= dVdi(i, j)
        F[j] -= dVdj(i, j)
```

$$V = \sum_i \sum_j V(i,j,\sum_k f(i,j,k))$$

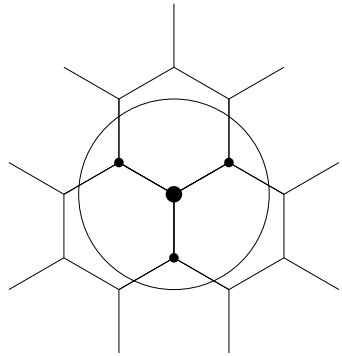
```
for i in atoms:
  for j in neighbors(i):
    tmp = 0
    for k in neighbors(i):
      tmp += f(i, j, k)
    V += V(i, j, tmp)
    F[i] -= dVdi(i, j, tmp)
    F[j] -= dVdj(i, j, tmp)
    tmp = dVdf(i, j, tmp)
    for k in neighbors(i):
      F[i] -= tmp * dfdi(i, j, k)
      F[j] -= tmp * dfdj(i, j, k)
      F[k] -= tmp * dfdk(i, j, k)
```

Vectorization

x86	SSE	128 bit
	AVX(2)	256 bit
	AVX-512 (IMCI)	512 bit
ARM	NEON	128 bit
	SVE	up to 2048 bit
POWER	AltiVec/VMX/VSX	128 bit
	QPX	256 bit
SPARC	HPC-ACE	128 bit
	HPC-ACE2	256 bit

Manual Optimization: Tersoff

$$V = \sum_i \sum_j V(i, j, \sum_k f(i, j, k))$$



The Vectorization of the Tersoff Multi-Body Potential: An Exercise in Performance Portability

Markus Höhnerbach
RWTH Aachen University

Ahmed E. Ismail
RWTH Aachen University,
West Virginia University

Paolo Bientinesi
RWTH Aachen University

Manual Optimization: AIREBO

$$
\begin{aligned}
V = & \sum_i \sum_j^i V(i,j,\sum_k^i f(i,k),\sum_l^j f(j,l),\sum_k^i \sum_l^j g(i,j,k,l)) \\
& + \sum_i \sum_j^i \sum_k^i \sum_l^j V'(i,j,k,l) \\
& + \sum_i \sum_j^{i'} V''(i,j,\max g(i,j,k,l) \forall k,l,\sum_k^i f(i,k),\sum_l^j f(j,l),\sum_k^i \sum_l^j g(i,j,k,l))
\end{aligned}
$$

Optimizing AIREBO:
Navigating the Journey from Complex Legacy Code to High Performance
Markus Höhnerbach, Paolo Bientinesi
submitted

PotC: A domain-specific language + compiler

Goal: Good –not best– performance on different platforms and arbitrary potentials

```
energy 1 / 2 * sum(i : all_atoms)
  sum(j : neighbors(i, R(i, j, j) + D(i, j, j))) V(i, j);
function V(i : atom; j : atom) =  f_C(i, j, r(i, j)) *
 (f_R(i, j, r(i, j)) + b(i, j) * f_A(i, j, r(i, j)));
function f_C(i : atom_type; j : atom_type; k : atom_type; r : distance) =
 implicit(i : i; j : j; k : k) piecewise(r <= R - D : 1;
 R - D < r < R + D: 1 / 2 - 1 / 2 * sin(pi / 2 * (r - R) / D); r >= R + D : 0);
# ...
function b(i : atom; j : atom) = (1 + beta(i, j) ^ n(i, j) *
  zeta(i, j) ^ n(i, j)) ^ (-1 / (2 * n(i, j)));
function zeta(i : atom; j : atom) =
 sum(k : neighbors(i, R(i, j, k) + D(i, j, k), j))
 implicit(i : i; j : j; k : k) f_C(r(i, k)) * g(theta(j, i, k)) *
   exp(lambda_3 ^ m * (r(i, j) - r(i, k)) ^ m);
parameter A(i : atom_type; j : atom_type) = file(1);
# ...
parameter gamma(i : atom_type; j : atom_type; k : atom_type) = file(5);
```

- About me
- Molecular dynamics
- **Mixed precision calculations**
- Accuracy & performance modeling
- Linear Algebra, applications
- Tensor operations

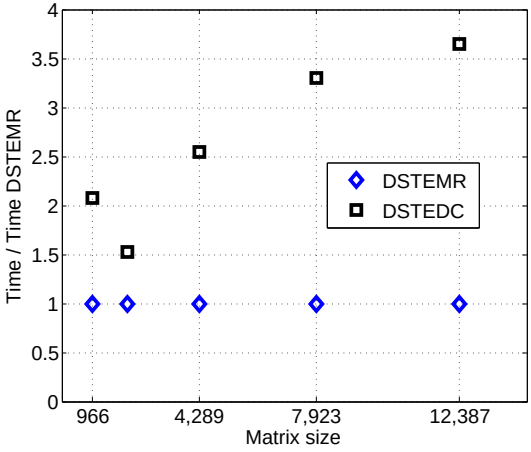
Tridiagonal Eigenproblem

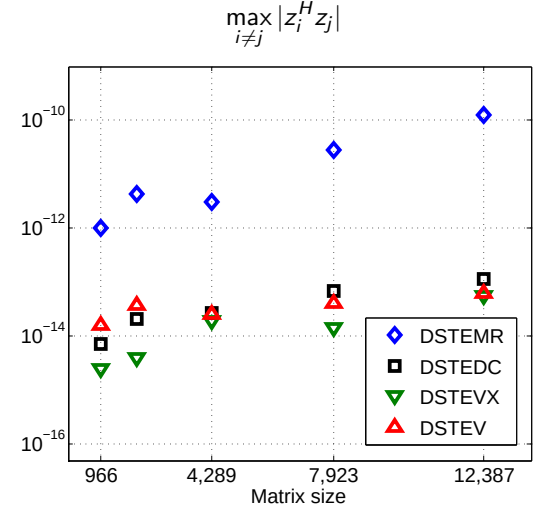
Algorithms			
1958	Bisection + Inverse Iteration (BI)	subsets	$O(kn^2)$
1961	QR	robust, accurate	$O(n^3)$
1981	Divide & Conquer (DC)	BLAS3, accurate	$O(n^3)$
1997	MR3 / MRRR	subsets, fast	$O(kn)$

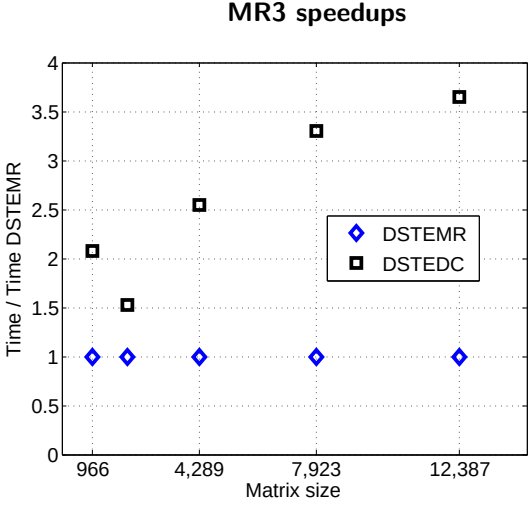
Tridiagonal Eigenproblem

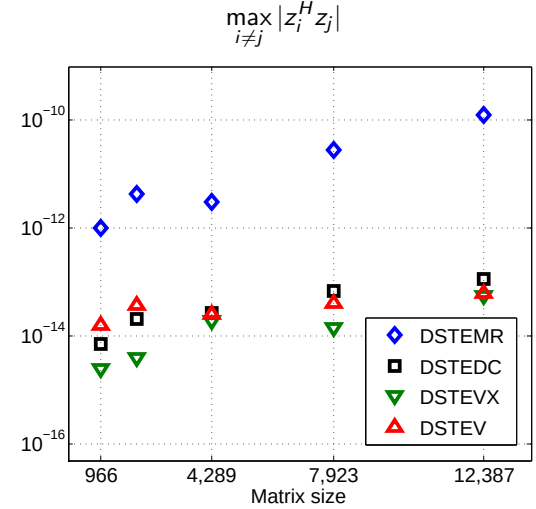
Algorithms			
1958	Bisection + Inverse Iteration (BI)	subsets	$O(kn^2)$
1961	QR	robust, accurate	$O(n^3)$
1981	Divide & Conquer (DC)	BLAS3, accurate	$O(n^3)$
1997	**MR3 / MRRR**	subsets, fast	$O(kn)$

- I.Dhillon & B.Parlett (U.C. Berkeley)
- no reorthogonalization $\Rightarrow$ k eigenpairs in $O(nk)$ operations
- 1) eigenvalues (*dqds*, *bisection*)
  2) eigenvectors + eigenvalues


MR3 speedups
Time / Time DSTEMR
Matrix size
DSTEMR
DSTEDC
0
0.5
1
1.5
2
2.5
3
3.5
4
966
4,289
7,923
12,387


$\max_{i \neq j} |z_i^H z_j|$
Matrix size
DSTEMR
DSTEDC
DSTEVX
DSTEV
10^{-10}
10^{-12}
10^{-14}
10^{-16}
966
4,289
7,923
12,387

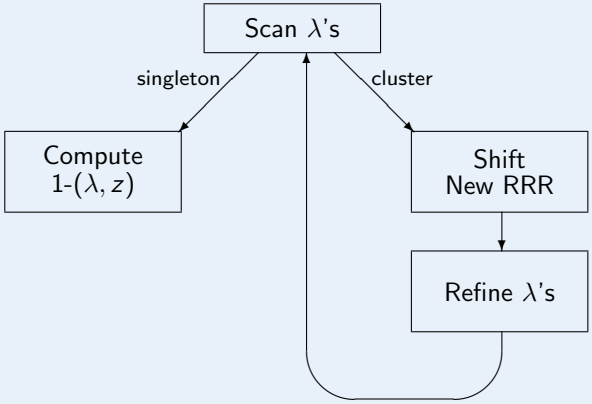




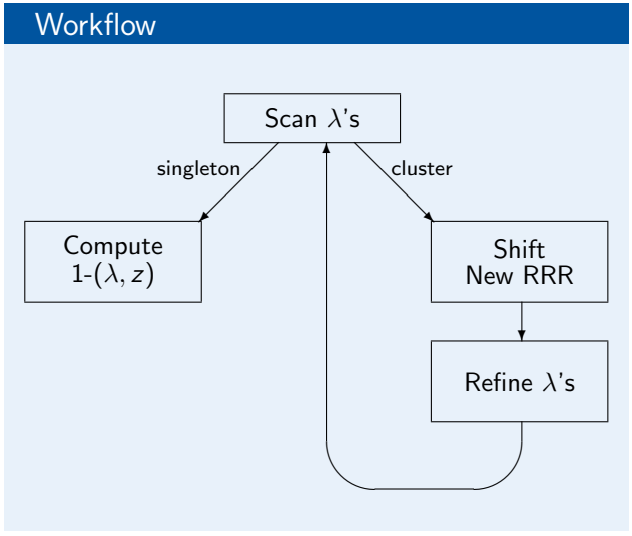
Idea: Trade speed for accuracy ...diametrically opposite approach wrt to current trends

MR3

Workflow



MR3



$$\texttt{if} \quad \min_{j \neq i} \frac{|\lambda_i - \lambda_j|}{|\lambda_i|} \geq \texttt{gaptol}$$

$$\texttt{then} \quad \lambda_i \text{ is a singleton}$$

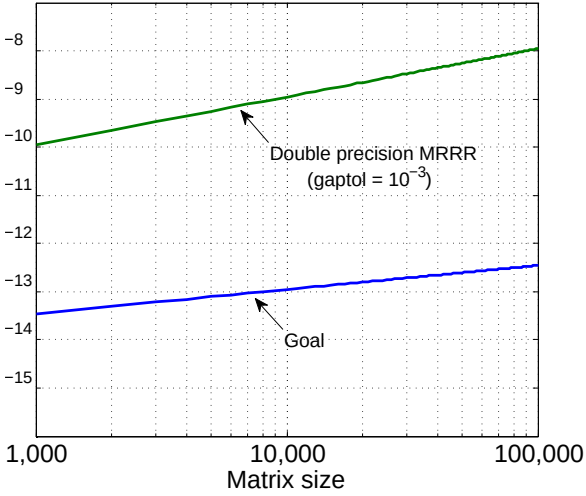
- `gaptol`: "small" (loose) $\Rightarrow$
  less clustering, better robustness,
  more parallelism, more BX
- `gaptol`: "large" (strict) $\Rightarrow$
  more work, deeper trees,
  better orthogonality (?),
  more failures

$$\text{orthogonality} \lesssim \frac{n\,\varepsilon}{gaptol}$$

A Mixed Precision Approach

$$\text{orthogonality} \lesssim \frac{n\,\varepsilon}{gaptol}$$

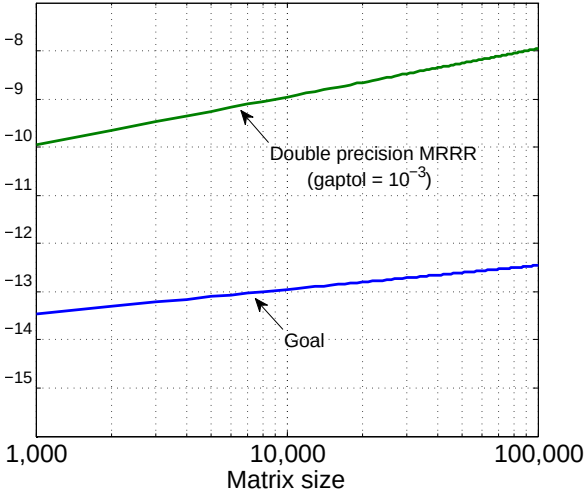
Double Precision



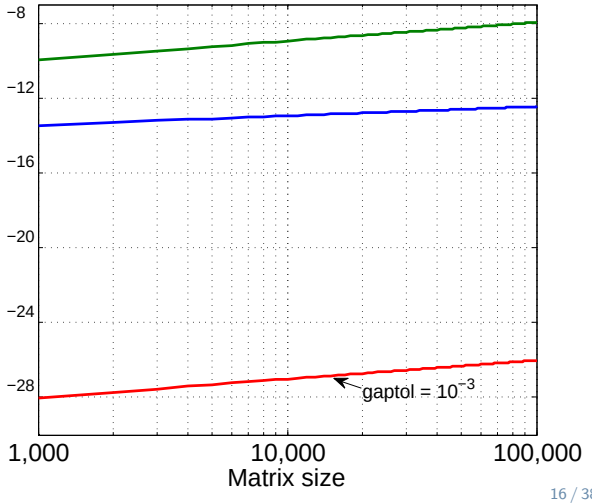
A Mixed Precision Approach

$$\text{orthogonality} \lesssim \frac{n\,\varepsilon}{gaptol}$$

Double Precision



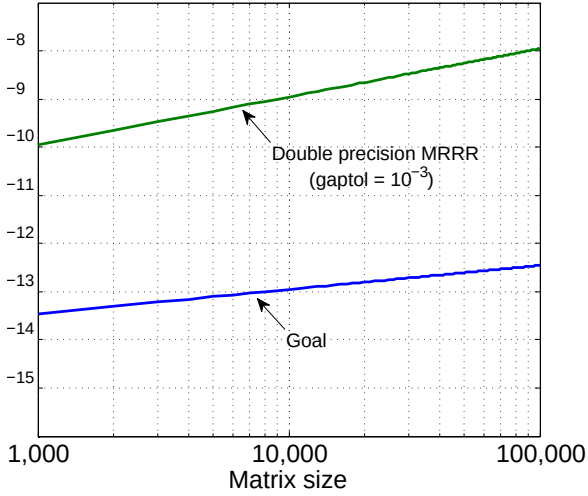
Quad Precision



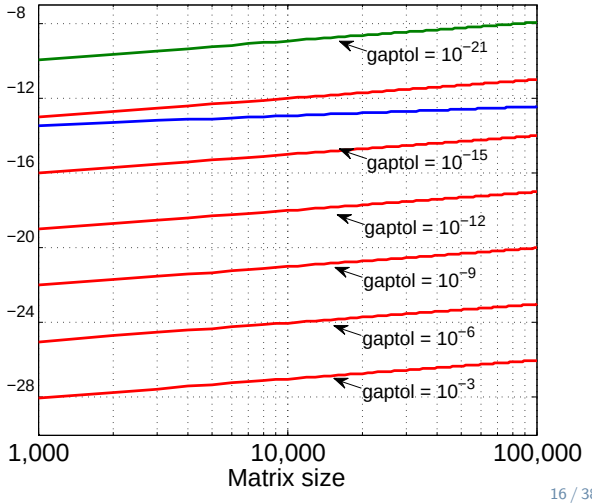
A Mixed Precision Approach

$$\text{orthogonality} \lesssim \frac{n\,\varepsilon}{gaptol}$$

Double Precision

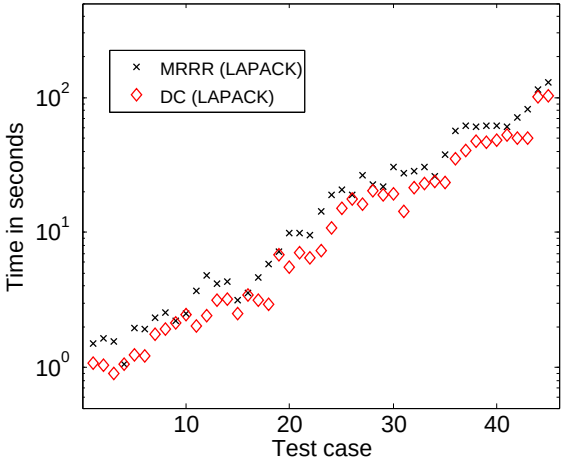


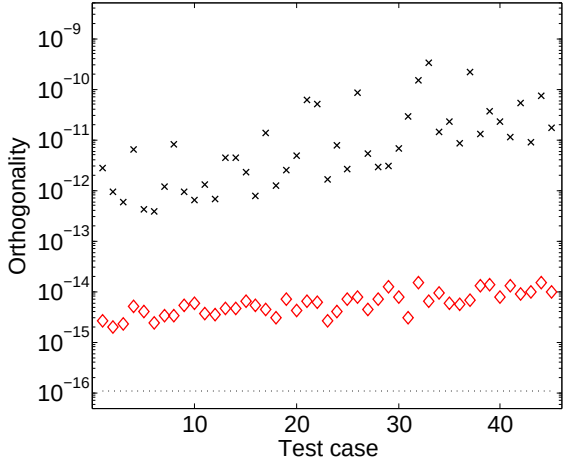
Quad Precision



Results: Dense Real Symmetric Eigenproblems

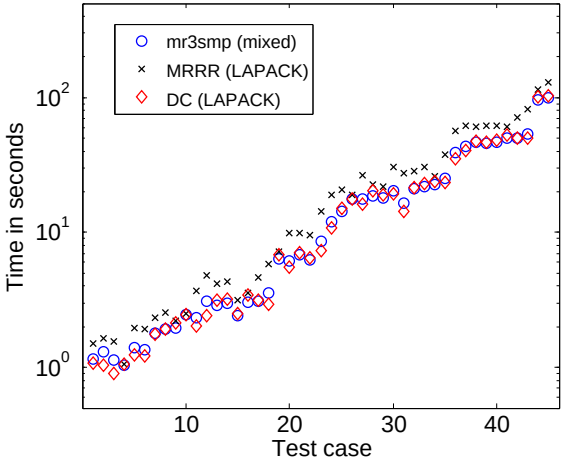
32 cores — 45 application matrices $n \in [1000..8000]$

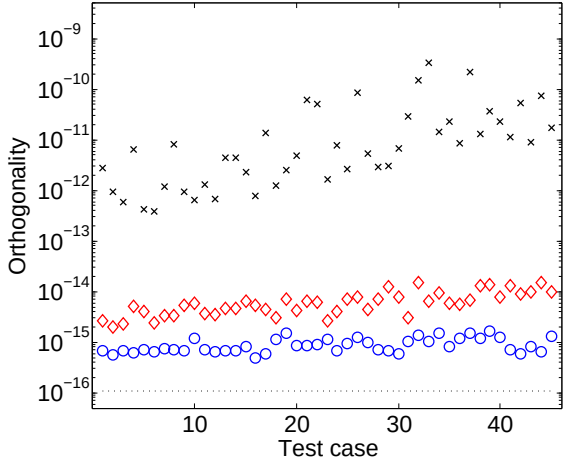




Results: Dense Real Symmetric Eigenproblems

32 cores — 45 application matrices $n \in [1000..8000]$

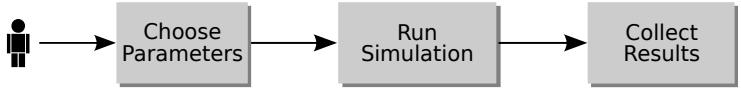




- About me
- Molecular dynamics
- Mixed precision calculations
- **Accuracy & performance modeling**
- Linear Algebra, applications
- Tensor operations

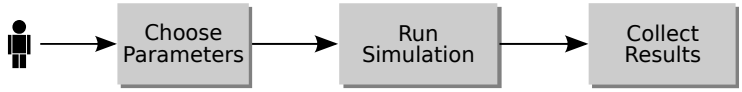
Numerical simulations

Typical workflow

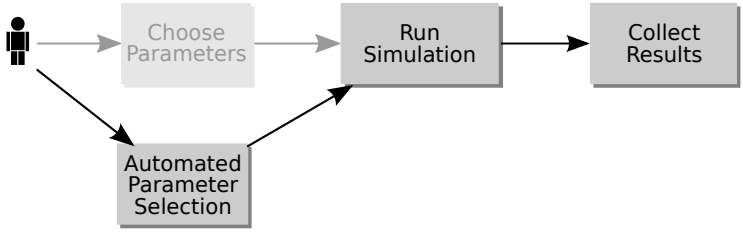


Numerical simulations

Typical workflow



Proposed workflow



Problem: Parameters selection

- Optimization problem — p: input parameters

$$\min_{p} \ \text{time}(\text{resources}, p) \quad \text{subject to} \quad \text{accurate}(p)$$

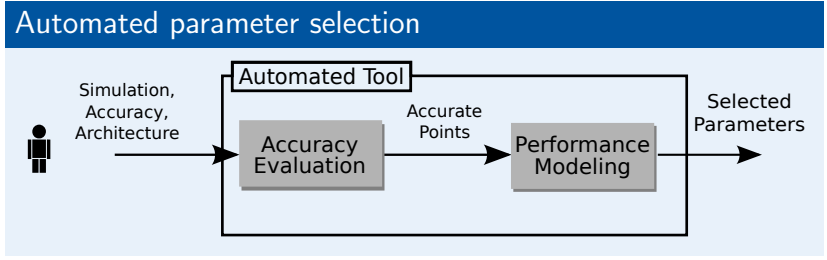
- Easily millions of combinations of parameters
- What is the "fastest" combination that leads to sufficiently accurate results?

Problem: Parameters selection

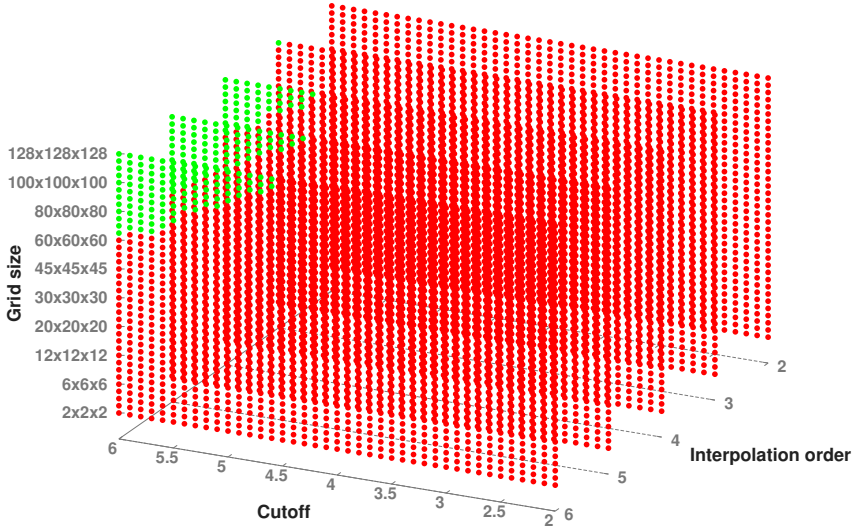
- Optimization problem — p: input parameters

$$\min_{p} \ \text{time}(\text{resources}, p) \quad \text{subject to} \quad \text{accurate}(p)$$

- Easily millions of combinations of parameters
- What is the "fastest" combination that leads to sufficiently accurate results?

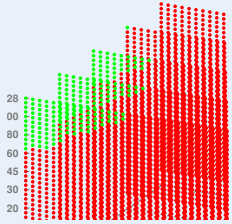


1) Accuracy boundaries



2) Performance models

Empirical



Accurate but **slow**

Analytical

$$52\frac{\frac{4}{3}\pi c^3}{L_x L_y L_z}N^2 + 5Nlog(N) + 8(G_x G_y G_z) + \dots$$

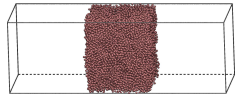
Fast but **inaccurate**

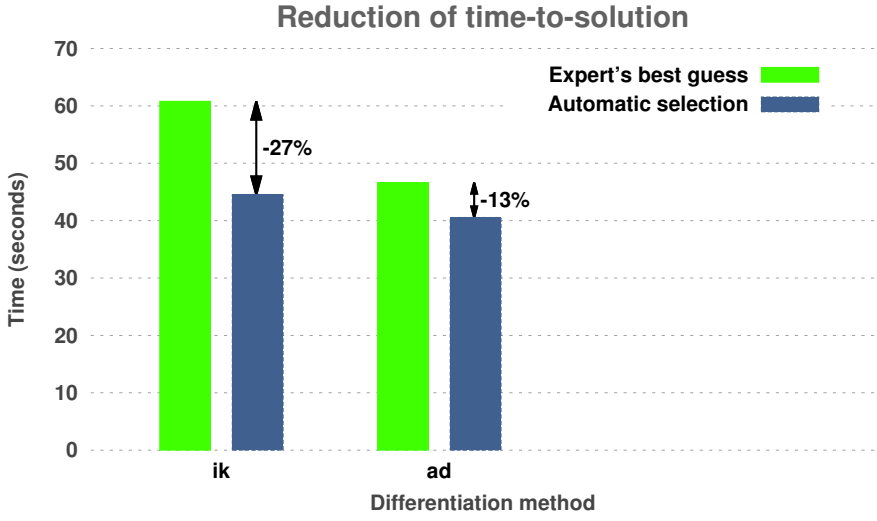
Hybrid

Few samples + simple models

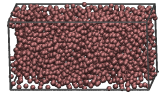
Both **Fast and Accurate**?

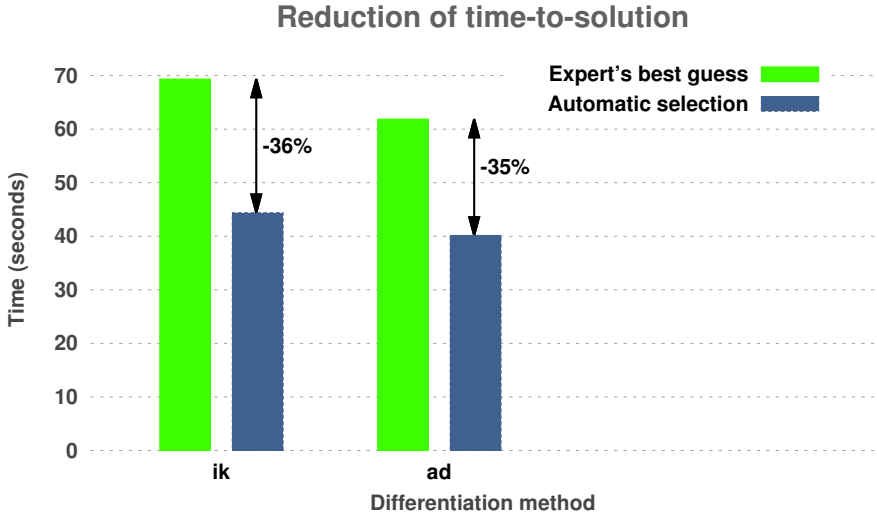
Results – Interfacial System





Results – Bulk System





- About me
- Molecular dynamics
- Mixed precision calculations
- Accuracy & performance modeling
- **Linear Algebra, applications**
- Tensor operations

Linear Algebra Applications

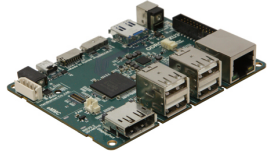
Generalized Least Squares	$b := (X^T M^{-1} X)^{-1} X^T M^{-1} y$	$n > m$; $M \in \mathbb{R}^{n \times n}$, SPD; $X \in \mathbb{R}^{n \times m}$; $y \in \mathbb{R}^{n \times 1}$
Signal Processing	$x := \left(A^{-T} B^T B A^{-1} + R^T L R\right)^{-1} A^{-T} B^T B A^{-1} y$	
Kalman Filter	$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}$; $x_k^a := x_k^b + K_k(z_k - H x_k^b)$; $P_k^a := (I - K_K H) P_k^b$	
Ensemble Kalman Filter	$X^a := X^b + \left(B^{-1} + H^T R^{-1} H\right)^{-1} \left(Y - H X^b\right)$	
Image Restoration	$x_k := (H^T H + \lambda \sigma^2 I_n)^{-1} (H^T y + \lambda \sigma^2 (v_{k-1} - u_{k-1}))$	
Rand. Matrix Inversion	$X_{k+1} := S(S^T A S)^{-1} S^T + (I_n - S(S^T A S)^{-1} S^T A) X_k (I_n - A S (S^T A S)^{-1} S^T)$	
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	$x_f := W A^T (A W A^T)^{-1} (b - Ax)$; $x_o := W (A^T (A W A^T)^{-1} A x - c)$	
Tikhonov Regularization	$x := (A^T A + \Gamma^T \Gamma)^{-1} A^T b$	$A \in \mathbb{R}^{n \times m}$; $\Gamma \in \mathbb{R}^{m \times m}$; $b \in \mathbb{R}^{n \times 1}$
Gen. Tikhonov Reg.	$x := (A^T P A + Q)^{-1} (A^T P b + Q x_0)$	$P \in \mathbb{R}^{n \times n}$, SSPD; $Q \in \mathbb{R}^{m \times m}$, SSPD; $x_0 \in \mathbb{R}^{m \times 1}$
LMMSE estimator	$K_{t+1} := C_t A^T (A C_t A^T + C_z)^{-1}$; $x_{t+1} := x_t + K_{t+1}(y - A x_t)$; $C_{t+1} := (I - K_{t+1} A) C_t$	

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

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

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

?



MUL ADD MOV

MOVAPD

VFMADDPD ...

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

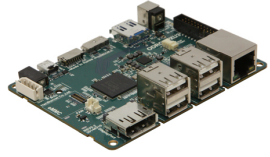
$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T(HC_\dagger H^T)^{-1} \end{cases}$$

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

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

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

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



MUL ADD MOV

MOVAPD

VFMADDPD ...

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

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

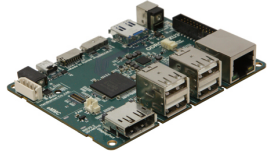
$$E := Q^{-1}U(I + U^TQ^{-1}U)^{-1}U^T \quad \ldots$$

?

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

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

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



MUL ADD MOV

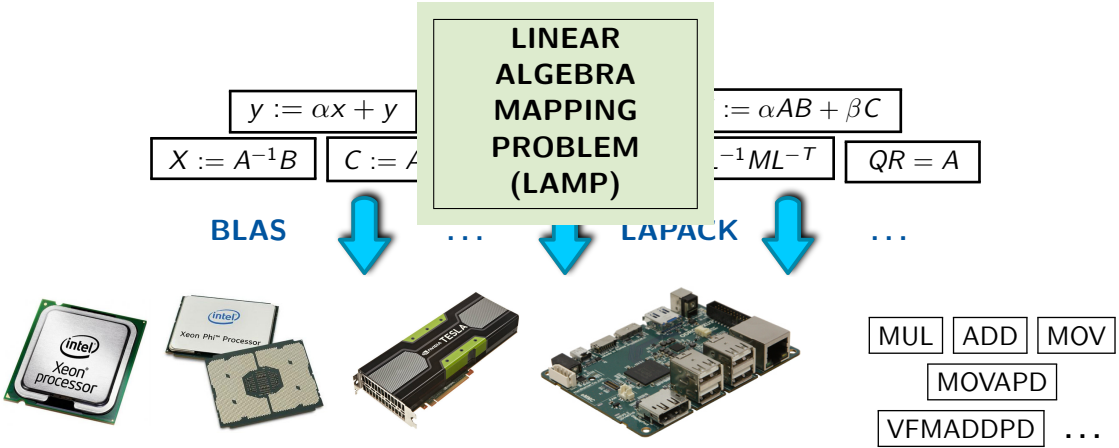
MOVAPD

VFMADDPD ...

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

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

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



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

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

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

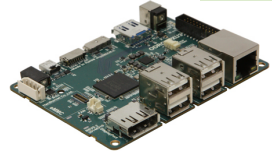
LINEAR ALGEBRA MAPPING PROBLEM (LAMP)

1. Associativity
   Generalized matrix chain
2. Specialized kernels
   Property propagation
3. Common subexpressions
4. Algebraic identities

$y := \alpha x + y$

$X := A^{-1}B$ $\quad C := A$ $\quad \ldots$

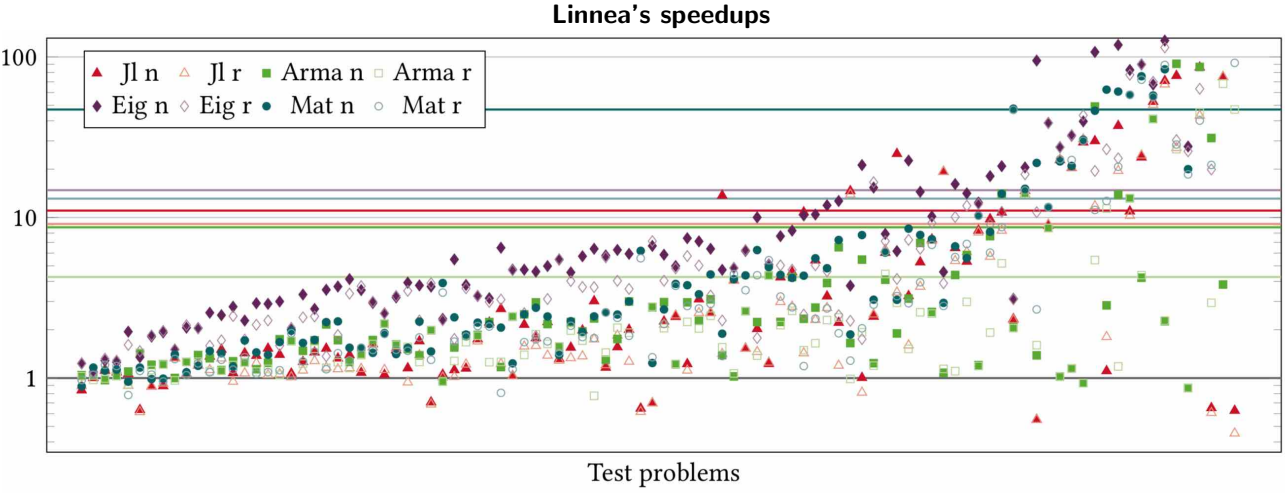
BLAS $\quad \ldots \quad$ **LAPACK**



MUL | ADD | MOV

MOVAPD

VFMADDPD $\quad \ldots$



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

- About me
- Molecular dynamics
- Mixed precision calculations
- Accuracy & performance modeling
- Linear Algebra, applications
- **Tensor operations**

Tensor Operations

Coupled-Cluster methods

$$
\begin{aligned}
\tau_{ij}^{ab} &= t_{ij}^{ab} + \frac{1}{2} P_b^a P_j^i t_i^a t_j^b, \\
\tilde{F}_e^m &= f_e^m + \sum_{fn} v_{ef}^{mn} t_n^f, \\
\tilde{F}_e^a &= (1-\delta_{ae}) f_e^a - \sum_m \tilde{F}_e^m t_m^a - \frac{1}{2} \sum_{mnf} v_{ef}^{mn} t_{mn}^{af} + \sum_{fn} v_{ef}^{an} t_n^f, \\
\tilde{F}_i^m &= (1-\delta_{mi}) f_i^m + \sum_e \tilde{F}_e^m t_i^e + \frac{1}{2} \sum_{nef} v_{ef}^{mn} t_{in}^{ef} + \sum_{fn} v_{if}^{mn} t_n^f,
\end{aligned}
$$

$$
\begin{aligned}
\tilde{W}_{ei}^{mn} &= v_{ei}^{mn} + \sum_f v_{ef}^{mn} t_i^f, \\
\tilde{W}_{ij}^{mn} &= v_{ij}^{mn} + P_j^i \sum_e v_{ie}^{mn} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{mn} \tau_{ij}^{ef}, \\
\tilde{W}_{ie}^{am} &= v_{ie}^{am} - \sum_n \tilde{W}_{ei}^{mn} t_n^a + \sum_f v_{ef}^{ma} t_i^f + \frac{1}{2} \sum_{nf} v_{ef}^{mn} t_{in}^{af}, \\
\tilde{W}_{ij}^{am} &= v_{ij}^{am} + P_j^i \sum_e v_{ie}^{am} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{am} \tau_{ij}^{ef}, \\
z_i^a &= f_i^a - \sum_m \tilde{F}_i^m t_m^a + \sum_e f_e^a t_i^e + \sum_{em} v_{ei}^{ma} t_m^e + \sum_{em} v_{im}^{ae} \tilde{F}_e^m + \frac{1}{2} \sum_{efm} \\
z_{ij}^{ab} &= v_{ij}^{ab} + P_j^i \sum_e v_{ie}^{ab} t_j^e + P_b^a P_j^i \sum_{me} \tilde{W}_{ie}^{am} t_{mj}^{eb} - P_b^a \sum_m \tilde{W}_{ij}^{am} t_m^b + P_b^a
\end{aligned}
$$

credits to D. Matthews, E. Solomonik, J. Stanton, and J. Gauss

Tensor Operations

Coupled-Cluster methods

$$
\begin{aligned}
\tau_{ij}^{ab} &= t_{ij}^{ab} + \frac{1}{2} P_b^a P_j^i t_i^a t_j^b, \\
\tilde{F}_e^m &= f_e^m + \sum_{fn} v_{ef}^{mn} t_n^f, \\
\tilde{F}_e^a &= (1-\delta_{ae}) f_e^a - \sum_m \tilde{F}_e^m t_m^a - \frac{1}{2} \sum_{mnf} v_{ef}^{mn} t_{mn}^{af} + \sum_{fn} v_{ef}^{an} t_n^f, \\
\tilde{F}_i^m &= (1-\delta_{mi}) f_i^m + \sum_e \tilde{F}_e^m t_i^e + \frac{1}{2} \sum_{nef} v_{ef}^{mn} t_{in}^{ef} + \sum_{fn} v_{if}^{mn} t_n^f, \\
\tilde{W}_{ei}^{mn} &= v_{ei}^{mn} + \sum_f v_{ef}^{mn} t_i^f, \\
\tilde{W}_{ij}^{mn} &= v_{ij}^{mn} + P_j^i \sum_e v_{ie}^{mn} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{mn} \tau_{ij}^{ef}, \\
\tilde{W}_{ie}^{am} &= v_{ie}^{am} - \sum_n \tilde{W}_{ei}^{mn} t_n^a + \sum_f v_{ef}^{ma} t_i^f + \frac{1}{2} \sum_{nf} v_{ef}^{mn} t_{in}^{af}, \\
\tilde{W}_{ij}^{am} &= v_{ij}^{am} + P_j^i \sum_e v_{ie}^{am} t_j^e + \frac{1}{2} \sum_{ef} v_{ef}^{am} \tau_{ij}^{ef}, \\
z_i^a &= f_i^a - \sum_m \tilde{F}_i^m t_m^a + \sum_e f_e^a t_i^e + \sum_{em} v_{ei}^{ma} t_m^e + \sum_{em} v_{im}^{ae} \tilde{F}_e^m + \frac{1}{2} \sum_{efm} v \\
z_{ij}^{ab} &= v_{ij}^{ab} + P_j^i \sum_e v_{ie}^{ab} t_j^e + P_b^a P_j^i \sum_{me} \tilde{W}_{ie}^{am} t_{mj}^{eb} - P_b^a \sum_m \tilde{W}_{ij}^{am} t_m^b + P_b^a
\end{aligned}
$$

credits to D. Matthews, E. Solomonik, J. Stanton, and J. Gauss

- **1)** Kernels
  Identification, standardization
- **2)** Decomposition / Mapping

- Tensor Transpositions Compiler
  TTC – `github.com/HPAC/TTC`
- High-Perf. Tensor Transp. Library
  HPTT – `github.com/HPAC/HPTT`
- Tensor Contraction Code Generator
  TCCG – `github.com/HPAC/TCCG`
- Tensor Contraction Library
  TCL – `github.com/springer13/tcl`

- **Tensor Transpositions**

$$\mathcal{B}_{i_1,i_2,\ldots,i_N} \leftarrow \alpha \cdot \mathcal{A}_{\pi(i_1,i_2,\ldots,i_N)} + \beta \cdot \mathcal{B}_{i_1,i_2,\ldots,i_N}$$

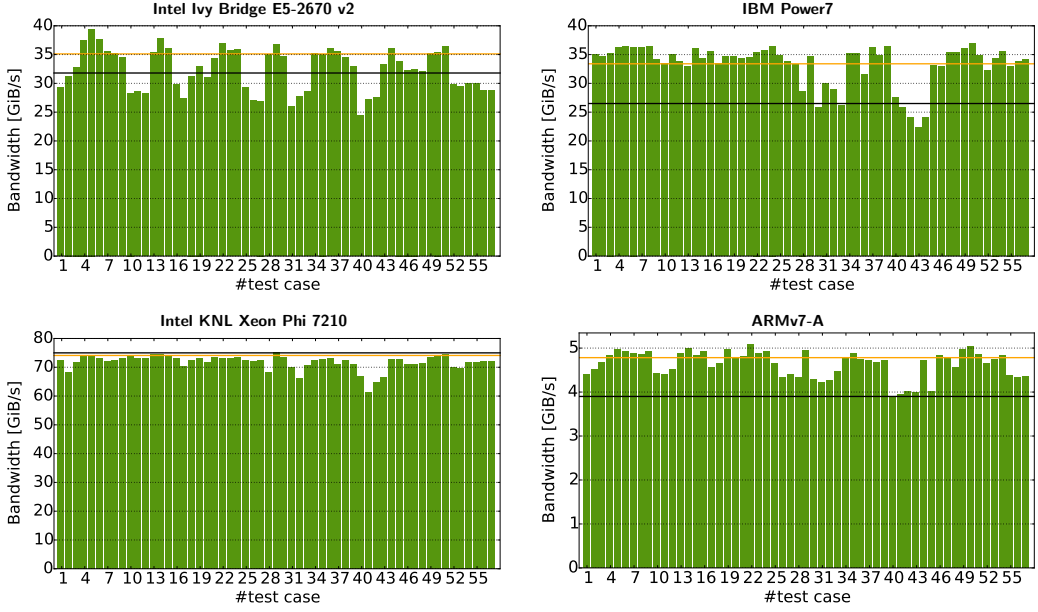
- **Summations** — linear summation over tensor transpositions

$$\begin{aligned}
\mathcal{B}_{i_0 i_1 i_2} &\leftarrow 2\mathcal{A}_{i_0 i_1 i_2} - \mathcal{A}_{i_2 i_1 i_0} - \mathcal{A}_{i_0 i_2 i_1} \\
\mathcal{B}_{i_0 i_1 i_2} &\leftarrow 4\mathcal{A}_{i_0 i_1 i_2} - 2\mathcal{A}_{i_1 i_0 i_2} - 2\mathcal{A}_{i_2 i_1 i_0} + \mathcal{A}_{i_1 i_2 i_0} - 2\mathcal{A}_{i_0 i_2 i_1} + \mathcal{A}_{i_2 i_0 i_1} \\
\mathcal{B}_{i_0 i_1 i_2 i_3} &\leftarrow 2\mathcal{A}_{i_0 i_1 i_2 i_3} - \mathcal{A}_{i_2 i_1 i_0 i_3} - \mathcal{A}_{i_0 i_2 i_1 i_3} - \mathcal{A}_{i_0 i_1 i_3 i_2}
\end{aligned}$$

- **Tensor Contractions**

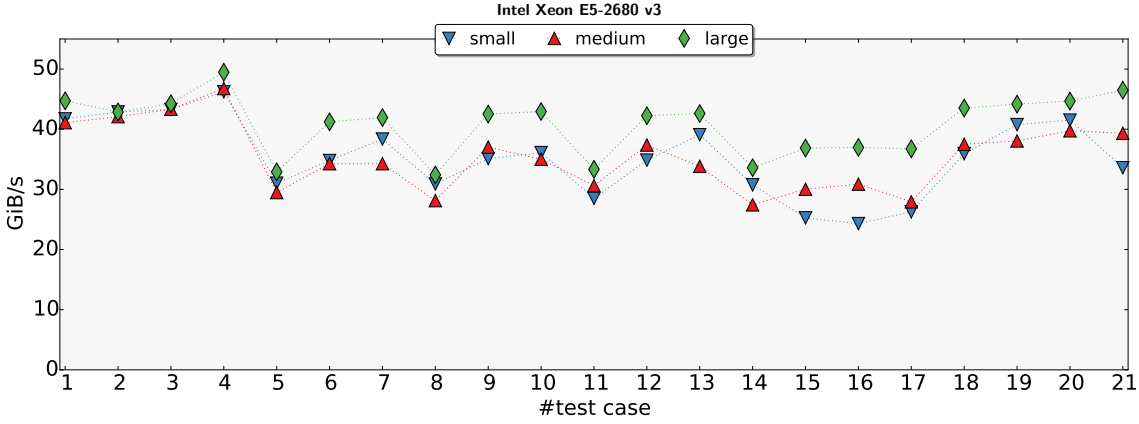
$$\mathcal{C}_{\pi^{\mathcal{C}}(I_m \cup I_n)} \leftarrow \alpha \cdot \mathcal{A}_{\pi^{\mathcal{A}}(I_m \cup I_k)} \times \mathcal{B}_{\pi^{\mathcal{B}}(I_n \cup I_k)} + \beta \cdot \mathcal{C}_{\pi^{\mathcal{C}}(I_m \cup I_n)}$$

HPTT – Transposition



Black and orange lines denote STREAM and AXPY bandwidth, respectively.

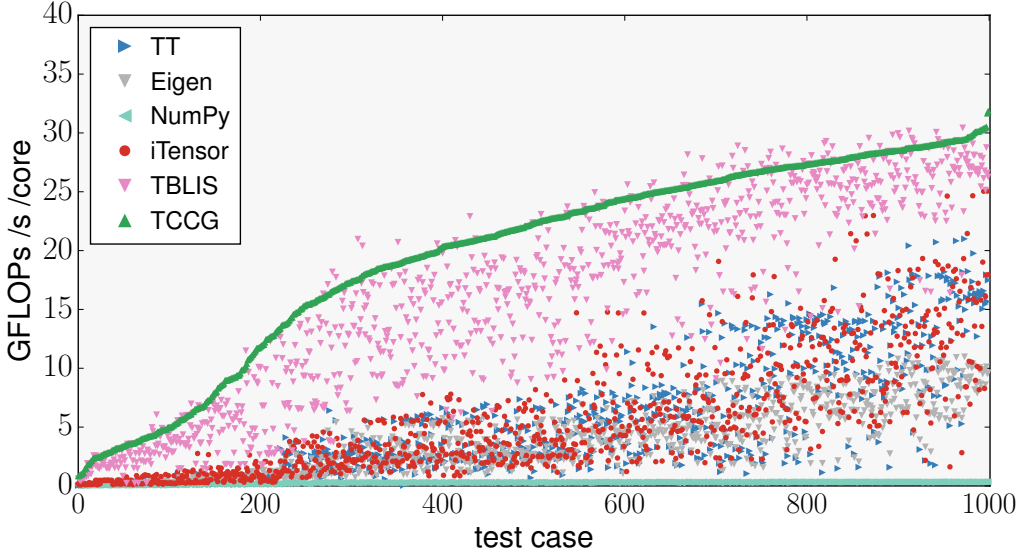
Summation



$$\text{BW} = \frac{2 \times N^d \times \texttt{sizeof(double)}}{2^{30} \times \text{Time}} \text{ GiB/s}, \qquad \text{TRIAD} : 107.7 \text{ GiB/s}$$

Problem size: Small (70MBs), medium (320Mbs), large (1200MBs).

TCCG – Contraction



DP tensor contractions. Host: 2×Haswell-EP E5-2680 v3 @ 24 threads

Mapping – LAMP (again)

Generation of Hamiltonian and Overlap Matrices

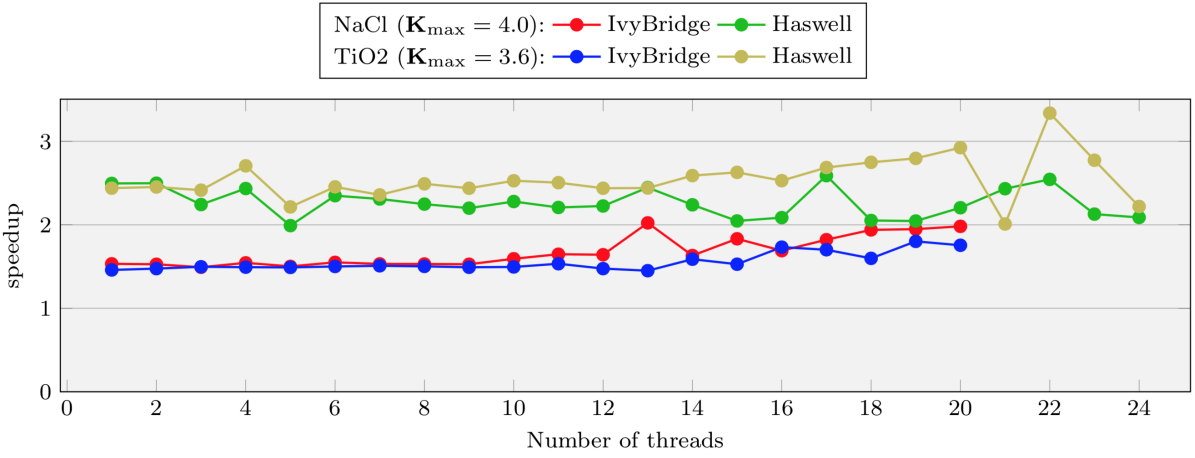
$$(S)_{G',G} = \sum_a \sum_{L=(l,m)} \left(A_L^{a,G'}\right)^* A_L^{a,G} + \left(B_L^{a,G'}\right)^* B_L^{a,G} \left\|\dot{u}_{l,a}\right\|^2$$

$$\begin{aligned}(H)_{G',G} = \sum_a \sum_{L',L} &\left(A^*_{L',a,t'}\, T^{[AA]}_{L',L;a}\, A_{L,a,t}\right) + \left(A^*_{L',a,t'}\, T^{[AB]}_{L',L;a}\, B_{L,a,t}\right) \\ &+ \left(B^*_{L',a,t'}\, T^{[BA]}_{L',L;a}\, A_{L,a,t}\right) + \left(B^*_{L',a,t'}\, T^{[BB]}_{L',L;a}\, B_{L,a,t}\right)\end{aligned}$$

"mappable" to BLAS and LAPACK kernels

Mapping – LAMP (again)

Generation of Hamiltonian and Overlap Matrices



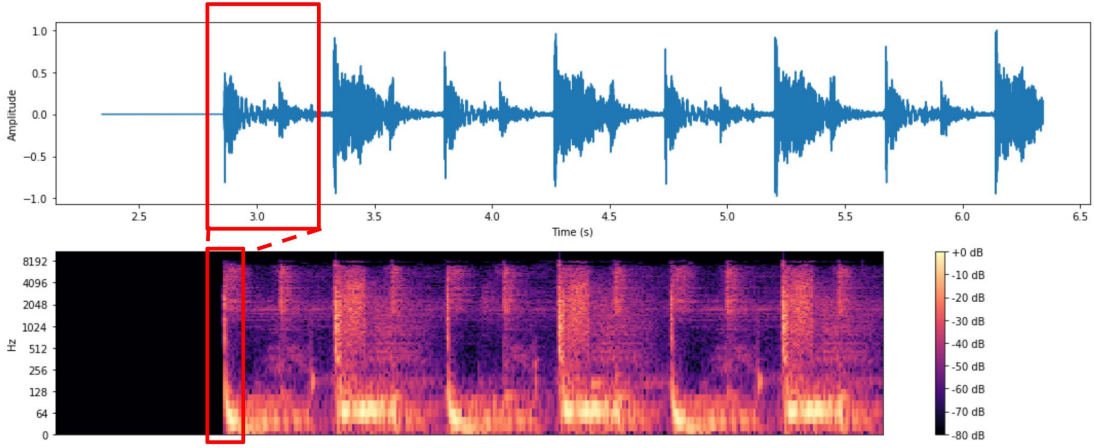
More on Tensors?

- The "other" operations?
  Data science — Decompositions, rank, compression, completion, ...
- Kernels?
- Common language? Interface?
- Mapping onto kernels

Two passions: mountains & music

Two passions: mountains & music

Two passions: mountains & music



Talk: "Automatic Dj-ing", next Wednesday, 9am