When $1 + 1 > 2$
The Power of Interdisciplinary Research

Paolo Bientinesi + Edoardo di Napoli

Aachen Institute for Advanced Study in Computational Engineering Science
RWTH Aachen University

SLQM Opening workshop, JSC
April 4, 2017
Outline

1 Historical perspective
2 Eigensolvers
3 Tensors
4 Conclusions
Good old times
Present

Computational Physics  Computational Chemistry  Computational Biology  …

Applied Mathematics / Numerics

Computer science / Algorithms

Implementation / HPC

When $1 + 1 > 2$ The Power of Interdisciplinary Research
Pros

- Separation of concerns
- Specialization
- High-performance
- Standardization
- Layering
But...

\[ y = X\beta + Zu + \epsilon \]

\[ \min_x ||Ax - b||^2 + ||\Gamma x||^2 \]

**LINEAR MIXED MODELS**

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

**LENNARD-JONES POTENTIAL**

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

**SCHRÖDINGER EQN.**

\[ \vdots \]
But...

\[ y = X\beta + Zu + \epsilon \]

\[
\min_x \|Ax - b\|^2 + \|\Gamma x\|^2
\]

**LINEAR MIXED MODELS**

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

**LENNARD-JONES POTENTIAL**

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

**SCHRÖDINGER EQN.**
But...

\[ y = X\beta + Zu + \epsilon \]

\[
\min_x ||Ax - b||^2 + ||\Gamma x||^2
\]

**LINEAR MIXED MODELS**

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

**LENNARD-JONES POTENTIAL**

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

**SCHRÖDINGER EQN.**

:::

**HPAC**

:::
A successful collaboration

Computer Science
Duke University

Physics
U. of North Carolina
A successful collaboration

Computer Science
Duke University

Physics
U. of North Carolina

- Eigensolvers
- Tensor contractions
- Tensor calculations
Eigensolvers – compartmental approach

- Development in isolation
- “There is a matrix at the door”
- Expert in numerics, expert in performance
- Disconnect between HPC and users
Accuracy vs. Time

Mixed-precision MR3-based symmetric eigensolver

application matrices, size ∈ [1.000, ..., 8.000]

Orthogonality

Execution time: 32 threads

Eigensolvers – collaborative approach

- Eigensolver as part of a **process**
- Where are the matrices coming from?
- What are the known properties?
- What are the eigenvectors used for?
“Correlations in Sequences of Generalized Eigenproblems Arising in Density Functional Theory”
A practical Density Functional Theory example

Sequences of eigenvalue problems arising in FLAPW

Initial guess for charge density $n_{\text{start}}(r)$

Compute discretized Kohn-Sham equations

Solve a set of eigenproblems $P^{(\ell)}_{k_1} \ldots P^{(\ell)}_{k_N}$

Converged?

Output
Electronic structure, ...

Yes

$|n^{(\ell)} - n^{(\ell-1)}| < \eta$

Compute new charge density $n^{(\ell)}(r)$

No

OUTPUT
Electronic structure, ...

$P^{(\ell)}_{k_i}: \begin{align*}
H^{(\ell)}_{k_i} x^{(\ell)}_j &= \lambda^{(\ell)}_j S^{(\ell)}_{k_i} x^{(\ell)}_j \\
&= 1, \ldots, \text{nev.}
\end{align*}$
Sequences of Eigenproblems

Adjacent iteration cycles

\[
\begin{align*}
\text{ITERATION } (\ell) & : \\
\mathbf{P}^{(\ell)}_{k_1} & \xrightarrow{\text{direct solver}} (X^{(\ell)}_{k_1}, \Lambda^{(\ell)}_{k_1}) \\
\mathbf{P}^{(\ell)}_{k_2} & \xrightarrow{\text{direct solver}} (X^{(\ell)}_{k_2}, \Lambda^{(\ell)}_{k_2}) \\
\mathbf{P}^{(\ell)}_{k_N} & \xrightarrow{\text{direct solver}} (X^{(\ell)}_{k_N}, \Lambda^{(\ell)}_{k_N}) \\
X & \equiv \{x_1, \ldots, x_{\text{nev}}\} \\
\Lambda & \equiv \text{diag}(\lambda_1, \ldots, \lambda_{\text{nev}})
\end{align*}
\]

Next cycle
Sequences of Eigenproblems

Adjacent iteration cycles

\[ \begin{align*}
\text{ITERATION } (\ell) & \\
\rightarrow & \\
P_{k_1}^{(\ell)} & \xrightarrow{\text{direct solver}} (X_{k_1}^{(\ell)}, \Lambda_{k_1}^{(\ell)}) \\
P_{k_2}^{(\ell)} & \xrightarrow{\text{direct solver}} (X_{k_2}^{(\ell)}, \Lambda_{k_2}^{(\ell)}) \\
P_{k_N}^{(\ell)} & \xrightarrow{\text{direct solver}} (X_{k_N}^{(\ell)}, \Lambda_{k_N}^{(\ell)}) \\
\rightarrow & \\
\text{ITERATION } (\ell + 1) & \\
\rightarrow & \\
P_{k_1}^{(\ell+1)} & \xrightarrow{\text{direct solver}} (X_{k_1}^{(\ell+1)}, \Lambda_{k_1}^{(\ell+1)}) \\
P_{k_2}^{(\ell+1)} & \xrightarrow{\text{direct solver}} (X_{k_2}^{(\ell+1)}, \Lambda_{k_2}^{(\ell+1)}) \\
P_{k_N}^{(\ell+1)} & \xrightarrow{\text{direct solver}} (X_{k_N}^{(\ell+1)}, \Lambda_{k_N}^{(\ell+1)}) \\
\rightarrow & \\
X & \equiv \{x_1, \ldots, x_{\text{nev}}\} \\
\Lambda & \equiv \text{diag}(\lambda_1, \ldots, \lambda_{\text{nev}}) \\
\end{align*} \]
An alternative solving strategy: the ChASE algorithm

Adjacent cycles

\[ P_{k_1}^{(\ell)} \xrightarrow{\text{iterative solver}} (X_{k_1}^{(\ell)}, \Lambda_{k_1}^{(\ell)}) \]

\[ P_{k_2}^{(\ell)} \xrightarrow{\text{iterative solver}} (X_{k_2}^{(\ell)}, \Lambda_{k_2}^{(\ell)}) \]

\[ P_{k_N}^{(\ell)} \xrightarrow{\text{iterative solver}} (X_{k_N}^{(\ell)}, \Lambda_{k_N}^{(\ell)}) \]

\[ X \equiv \{x_1, \ldots, x_{\text{nev}}\} \]

\[ \Lambda \equiv \text{diag}(\lambda_1, \ldots, \lambda_{\text{nev}}) \]

Next cycle
Speed-up

\[
\text{Speed-up} = \frac{\text{CPU time (input random vectors)}}{\text{CPU time (input approximate eigenvectors)}}
\]

\(\text{Au}_{98}\text{Ag}_{10} - n = 13,379 - 128 \text{ cores.}\)
The core of the algorithm: Chebyshev filter

In practice

Three-terms recurrence relation

\[ C_{m+1}(t) = 2x C_m(t) - C_{m-1}(t); \quad m \in \mathbb{N}, \quad C_0(t) = 1, \quad C_1(t) = x \]

\[ Z_m \overset{\text{def}}{=} p_m(\tilde{H}) Z_0 \quad \text{with} \quad \tilde{H} = H - cI_n \]

**FOR:** \( i = 1 \rightarrow \text{DEG} - 1 \)

\[ Z_{i+1} \leftarrow \frac{2 \sigma_{i+1}}{e} \tilde{H} \times Z_i - \sigma_{i+1} \sigma_i Z_{i-1} \]

**END FOR.**
Efficiency and portability

\[\text{Au}_{98}\text{Ag}_{10} - n = 8,970 - 32\text{ cores.}\]

Lanczos $<0.2\%$

Chebyshev filter $\approx 88\%$

Rayleigh-Ritz $\approx 3.2\%$

Residuals Convergence $\approx 8.7\%$

"An optimized and scalable eigensolver for sequences of eigenvalue problems"
Tensors

Hot, emerging topic in HPC

Lost in translation: tensor vs. multi-dimensional array

Tensor contraction = generalization of matrix product

⇒ can I use GEMM? Yes/no? When?

Paolo Bientinesi + Edoardo di Napoli

When $1 + 1 > 2$ The Power of Interdisciplinary Research
Tensors

- Hot, emerging topic in HPC
- Lost in translation: tensor vs. multi-dimensional array
- Tensor contraction = generalization of matrix product
  \[ \Rightarrow \text{can I use GEMM? Yes/no? When?} \]


**Taxonomy**

\[ V_{h_1 h_2 \ldots} := S_{i_1 i_2 \ldots} T_{j_1 j_2 \ldots} \]

**Definition:** \( \Delta(X) = \# \text{ of free indices of } X \)

**Class 1:** \( \Delta(S) = 0 \land \Delta(T) = 0 \)
- \( \times \) BLAS3
- \( \times \) BLAS2
- \( \checkmark \) BLAS1

**Class 2:** \( \Delta(S) \geq 1 \land \Delta(T) = 0 \) or \( \Delta(S) = 0 \land \Delta(T) \geq 1 \)
- \( \times \) BLAS3
- \( \checkmark \) BLAS2 (+ transp)
- \( \checkmark \) BLAS1

**Class 3:** \( \Delta(S) \geq 1 \land \Delta(T) \geq 1 \)
- \( \checkmark \) BLAS3 (+ transp)
- \( \checkmark \) BLAS2
- \( \checkmark \) BLAS1

“Towards an Efficient Use of the BLAS Library for Multilinear Tensor Contractions”,
Tensors (continued)

- Hot, emerging topic in HPC
- Lost in translation: tensor vs. multi-dimensional array
- Tensor contraction = generalization of matrix product
  ⇒ can I use GEMM? Yes/no? When?
- Contractions implemented as explicit loops
  
  **Pros**: Symmetries, simplifications
  **Cons**: Efficiency, parallelism, portability!!

Paolo Bientinesi + Edoardo di Napoli | When 1 + 1 > 2 The Power of Interdisciplinary Research
Tensors (continued)

- Hot, emerging topic in HPC
- Lost in translation: tensor vs. multi-dimensional array
- Tensor contraction = generalization of matrix product
  ⇒ can I use GEMM? Yes/no? When?

- Contractions implemented as explicit loops
  **Pros**: Symmetries, simplifications
  **Cons**: Efficiency, parallelism, portability!!

- Our approach:
  - Expose properties
  - Map to linear algebra libraries! (and develop tensor libraries)
  **Pros**: Inherited efficiency, parallelism, portability
  **Cons**: Not all symmetries are exploited
Tensor libraries

Tensor transpositions

Tensor contractions


Hamiltonian and Overlap matrices in FLAPW methods

Operatorial form

\[
(H)_{G',G} = \sum_a \int \int \varphi^*_G(r) \hat{H}_{KS} \varphi_G(r) \, dr,
\]

\[
(S)_{G',G} = \sum_a \int \int \varphi^*_G(r) \varphi_G(r) \, dr.
\]

Entrywise form

\[
(S)_{G',G} = \sum_a \sum_{L=(l,m)} (A^{a,G'}_L)^* A^{a,G}_L + (B^{a,G'}_L)^* B^{a,G}_L \| \hat{u}_{l,a} \|^2
\]

\[
(H)_{G',G} = \sum_a \sum_{L',L} \left( (A^{a,G'}_{L'})^* T^{[AA]}_{L',L;a} A^{a,G}_L \right) + \left( (A^{a,G'}_{L'})^* T^{[AB]}_{L',L;a} B^{a,G}_L \right)
\]

\[
+ \left( (B^{a,G'}_{L'})^* T^{[BA]}_{L',L;a} A^{a,G}_L \right) + \left( (B^{a,G'}_{L'})^* T^{[BB]}_{L',L;a} B^{a,G}_L \right).
\]
Hamiltonian and Overlap matrices

Tensor form

\[ H = \sum_{a=1}^{N_A} \left( A_a^H T_a^{[AA]} A_a \right) + \sum_{a=1}^{N_A} \left( A_a^H T_a^{[AB]} B_a + B_a^H T_a^{[BA]} A_a + B_a^H T_a^{[BB]} B_a \right) \]

\[ S = \sum_{a=1}^{N_A} A_a^H A_a + \sum_{a=1}^{N_A} B_a^H \dot{U}_a^H \dot{U}_a B_a \]
Constructing $S_{AA}$

An example of memory layout re-structuring

$$S_{AA} = \sum_{a=1}^{N_A} A^H_a A_a.$$ 

1: for $a := 1 \rightarrow N_A$ do
2: \hspace{2em} $S_{AA} = A^H_a A_a$
3: end for

\[\triangleright (\text{zherk: } 4N_LN_G^2 \text{ Flops})\]

1: $S_{AA} = A^H_* A_*$

\[\triangleright (\text{zherk: } 4N_A N_LN_G^2 \text{ Flops})\]
Constructing $H_{AB + BA + BB}$

An example of algorithm re-structuring

$$H_{AB + BA + BB} = \sum_{a=1}^{N_A} B_a^H (T_a^{[BA]} A_a) + (A_a^H T_a^{[AB]}) B_a +$$

$$\frac{1}{2} B_a^H (T_a^{[BB]} B_a) + \frac{1}{2} (B_a^H T_a^{[BB]}) B_a$$

$$= \sum_{a=1}^{N_A} B_a^H (T_a^{[BA]} A_a + \frac{1}{2} T_a^{[BB]} B_a) +$$

$$(A_a^H T_a^{[AB]} + \frac{1}{2} B_a^H T_a^{[BB]}) B_a$$

1: for $a := 1 \rightarrow N_A$ do
2: $Z_a = T_a^{[BA]} A_a$ \hfill $\triangleright$ (zgemm: $8N_L^2N_G$ Flops)
3: $Z_a = Z_a + \frac{1}{2} T_a^{[BB]} B_a$ \hfill $\triangleright$ (zhemm: $8N_L^2N_G$ Flops)
4: Stack $Z_a$ to $Z_*$ and $B_a$ to $B_*$
5: end for
6: $H = Z_*^H B_* + B_*^H Z_*$ \hfill $\triangleright$ (zher2k: $8N_A N_L N_G^2$ Flops)
### Experimental multi-core results

<table>
<thead>
<tr>
<th></th>
<th>NaCl ($K_{\text{max}} = 4.0$)</th>
<th></th>
<th>TiO$<em>2$ ($K</em>{\text{max}} = 3.6$)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IvyBridge</td>
<td>Haswell</td>
<td>IvyBridge</td>
<td>Haswell</td>
</tr>
<tr>
<td></td>
<td>HSDL A FLEUR</td>
<td>×</td>
<td>HDSL A FLEUR</td>
<td>×</td>
</tr>
<tr>
<td>1 core</td>
<td>31.53</td>
<td>48.31</td>
<td>1.53</td>
<td>19.00</td>
</tr>
<tr>
<td>2 cores</td>
<td>16.10</td>
<td>24.58</td>
<td>1.53</td>
<td>9.98</td>
</tr>
<tr>
<td>1 CPU</td>
<td>3.90</td>
<td>6.21</td>
<td>1.59</td>
<td>2.25</td>
</tr>
<tr>
<td>2 CPUs</td>
<td>2.61</td>
<td>5.20</td>
<td>1.99</td>
<td>1.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Scalability of HSDL A and FLEUR: execution times in minutes on Haswell (12 cores / CPU) and IvyBridge (10 cores / CPU); speedups of HSDL A over FLEUR in **bold.**

“High-performance generation of the Hamiltonian and Overlap matrices in FLAPW methods.”
Porting to heterogeneous architectures

Back-of-the-envelope analysis

- 5 lines of the algorithm constitute 97% of flops
- Correspond to BLAS-3 operations \((\text{gemm}, \text{herk}, \text{her2k})\)
- High arithmetic intensity and should fit GPUs well

- First step: offload these routine calls
- All BLAS kernels. Which library?
  - cuBLAS
  - cuBLAS-XT
  - MAGMA
  - BLASX

- 3x wrappers \((\text{zgemm}, \text{zherk}, \text{zher2k})\)
- Init and cleanup of cuda runtime and devices
- Allocate data in page-locked memory

Only around 100 lines of additional code
Experimental multi-GPU results

Test case: AuAg \((N_A = 108, N_L = 121, N_G = [3275 - 13379])\)

- CPU: E5-2650, 2 x 8core, 2.0GHz, 64GBs RAM (Sandy Bridge)
- 2 Nvidia Tesla K20X
- Peak performance: 256 GFs/s + 2 x 1.3 TFs/s

“Hybrid CPU-GPU generation of the Hamiltonian and Overlap matrices in FLAPW methods.”
Conclusions

Interdisciplinary research

- Close collaboration & interactions
- Common language!
- New questions, new challenges
- Lots of opportunities
Conclusions

Interdisciplinary research

- Close collaboration & interactions
- Common language!
- New questions, new challenges
- Lots of opportunities
- ...funding schemes ...
- Education
Thank you for your attention!