Linnea: Automatic Generation of Efficient Linear Algebra Programs

Henrik Barthels
Introduction

• How to compute the following expressions?

\[b := (X^TM^{-1}X)^{-1}X^TM^{-1}y\]
\[x := W(A^T(AWA^T)^{-1}b - c)\]
\[x := (A^{-T}B^TBA^{-1} + R^T[\Lambda(Rz)]R)^{-1}A^{-T}B^TBA^{-1}y\]
\[X_{k+1} := S(S^TAS)^{-1}S^T + (I_n - S(S^TAS)^{-1}S^T A)X_k(I_n - AS(S^TAS)^{-1}S^T)\]

• High-level languages are easy to use, but performance is usually suboptimal.
• BLAS and LAPACK can be fast, but require a lot of expertise.
• Goal: Simplicity and performance.
• Dense, mid- to large-scale linear algebra.
### Introduction

How to compute...

\[ y' := H^\dagger y + (I_n - H^\dagger H)x \quad \text{[TG17]} \]

...with these operations?

- \( x := Ab \)
- \( 2n^2 \)
- \( X := AB \)
- \( 2n^3 \)
- \( x := a \pm b \)
- \( n \)
- \( X := A \pm B \)
- \( n^2 \)
### Introduction

How to compute...

\[ y' := H^\dagger y + (I_n - H^\dagger H)x \quad [\text{TG17}] \]

...with these operations?

\[ x := Ab \quad 2n^2 \]
\[ X := AB \quad 2n^3 \]
\[ x := a \pm b \quad n \]
\[ X := A \pm B \quad n^2 \]

\[
M_1 := H^\dagger H \\
M_2 := I_n - M_1 \\
m_3 := M_2 x \\
m_4 := H^\dagger y \\
y' := m_3 + m_4
\]

\[ \Rightarrow 2n^3 + 5n^2 + n \text{ FLOPs} \]
Introduction

How to compute...

\[ y' := H^\dagger y + (I_n - H^\dagger H)x \]  
\[ \iff y' := H^\dagger y + x - H^\dagger Hx \]  
\[ \iff y' := H^\dagger (y - Hx) + x \]

...with these operations?

\[ x := Ab \quad 2n^2 \]  
\[ X := AB \quad 2n^3 \]  
\[ x := a \pm b \quad n \]  
\[ X := A \pm B \quad n^2 \]

\[ M_1 := H^\dagger H \]  
\[ M_2 := I_n - M_1 \]  
\[ m_3 := M_2 x \]  
\[ m_4 := H^\dagger y \]  
\[ y' := m_3 + m_4 \]  
\[ \Rightarrow 2n^3 + 5n^2 + n \text{ FLOPs} \]
How to compute...

\[ y' := H^\dagger y + (I_n - H^\dagger H)x \quad \text{[TG17]} \]
\[ \iff y' := H^\dagger y + x - H^\dagger Hx \]
\[ \iff y' := H^\dagger (y - Hx) + x \]

...with these operations?

\[ x := Ab \quad 2n^2 \]
\[ X := AB \quad 2n^3 \]
\[ x := a \pm b \quad n \]
\[ X := A \pm B \quad n^2 \]

\[ M_1 := H^\dagger H \]
\[ M_2 := I_n - M_1 \]
\[ m_3 := M_2 x \]
\[ m_4 := H^\dagger y \]
\[ y' := m_3 + m_4 \]
\[ \Rightarrow 2n^3 + 5n^2 + n \text{ FLOPs} \]

\[ m_1 := Hx \]
\[ m_2 := y - m_1 \]
\[ m_3 := H^\dagger m_2 \]
\[ y' := m_3 + x \]
\[ \Rightarrow 2n^2 + 2n \text{ FLOPs} \]
Input

\[
n = 1500 \\
m = 1000 \\
\]

Matrix \( M(n, n) \) \(<\text{SPD}>\) \\
Matrix \( X(n, m) \) \(<\text{FullRank}>\) \\
ColumnVector \( y(n) \) \(<>\) \\
ColumnVector \( b(m) \) \(<>\) \\

\[ b = \text{inv}(X'*\text{inv}(M)*X)*X'*\text{inv}(M)*y \]
Instruction Set

**BLAS** [DDC$^+90$]
- $y \leftarrow Ax + y$
- $C \leftarrow AB + C$
- $B \leftarrow A^{-1}B$
- ...

**LAPACK** [AB$^+99$]
- Matrix factorizations.
- Eigensolvers.
- Solvers for linear systems with specific properties.
Linear Algebra Knowledge

- Properties: \texttt{trmm vs. gemm}
- Inference of properties: \( A \triangleright B \rightarrow AB \)
- Simplifications: \( A^T \rightarrow A \) if Symmetric(\( A \))
- Rewriting expressions:
  \[
  X := A^T A + A^T B + B^T A \quad \rightarrow \quad Y := B + A/2 \\
  X := A^T Y + Y^T A
  \]
- Common subexpressions:
  \[
  X := AB^{-T} C + B^{-1} A^T \quad \rightarrow \quad Z := AB^{-T} \\
  X := Z C + Z^T
  \]
- Matrix chains:
  \[
  (AB)c \quad \mathcal{O}(n^3) \\
  A(Bc) \quad \mathcal{O}(n^2)
  \]
\[ y' := H^\dagger y + (I_n - H^\dagger H)x \]
\[ y' := H^\dagger y + (I_n - H^\dagger H)x \]

\[ M_1 := H^\dagger H \]

\[ y' := H^\dagger y + (I_n - M_1)x \]
\begin{align*}
y' &:= H^\dagger y + x - H^\dagger Hx \\
M_1 &:= H^\dagger H \\
y' &:= H^\dagger y + (I_n - M_1)x
\end{align*}
Derivation Graph

\[ y' := H^\dagger (y - Hx) + x \]

\[ M_1 := H^\dagger H \]

\[ y' := H^\dagger y + (I_n - M_1)x \]
Derivation Graph

\[
\begin{align*}
y' &:= H^\dagger (y - Hx) + x \\
m_1 &:= Hx \\
y' &:= H^\dagger (y - m_1) + x \\
M_1 &:= H^\dagger H \\
y' &:= H^\dagger y + (I_n - M_1)x
\end{align*}
\]
Derivation Graph

\[ y' := H^\dagger(y - Hx) + x \]
\[ m_1 := Hx \]
\[ y' := H^\dagger y - H^\dagger m_1 + x \]
\[ m_2 := y - m_1 \]
\[ y' := H^\dagger m_2 + x \]
\[ M_1 := H^\dagger H \]
\[ y' := H^\dagger y + (I_n - M_1)x \]
\[ m_4 := H^\dagger m_1 \]
\[ \ldots \]
\[ y' := H^\dagger (y - Hx) + x \]

\[ m_1 := Hx \]

\[ y' := H^\dagger y - H^\dagger m_1 + x \]

\[ m_2 := y - m_1 \]

\[ y' := H^\dagger m_2 + x \]

\[ M_1 := H^\dagger H \]

\[ y' := H^\dagger y + (I_n - M_1)x \]

\[ m_4 := H^\dagger m_1 \]

\[ \ldots \]

\[ \ldots \]
Derivation Graph

Reducing Redundancy

\[ X := A BC + CDE \]
\[ T_1 := AB \]
\[ X := T_1C + CDE \]
\[ T_2 := T_1C \]
\[ X := T_2 + CDE \]
\[ T_3 := BC \]
\[ X := AT_3 + CDE \]
\[ T_4 := AT_3 \]
\[ X := T_4 + CDE \]
Reducing Redundancy

\[ X := ABC + CDE \]

- \[ T_1 := AB \]
- \[ X := T_1C + CDE \]
- \[ T_2 := T_1C \]
- \[ X := T_2 + CDE \]
- \[ T_3 := BC \]
- \[ X := AT_3 + CDE \]
- \[ T_2 := AT_3 \]
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]
\[ M_1 := AB \]
\[ X := M_1 + AC + AD \]
Derivation Graph

Reducing Redundancy

\[
X := AB + AC + AD
\]
\[
M_1 := AB
\]
\[
X := M_1 + AC + AD
\]
\[
M_2 := AC
\]
\[
X := M_1 + M_2 + AD
\]

<table>
<thead>
<tr>
<th>tmp</th>
<th>expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_1)</td>
<td>(AB)</td>
</tr>
<tr>
<td>(M_2)</td>
<td>(AC)</td>
</tr>
</tbody>
</table>
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]

\[ M_1 := AB \]

\[ X := M_1 + AC + AD \]

\[ M_2 := AC \]

\[ X := M_1 + M_2 + AD \]

\[ M_3 := M_1 + M_2 \]

\[ X := M_3 + AD \]
Reducing Redundancy

\[
X := AB + AC + AD \\
M_1 := AB \\
X := M_1 + AC + AD \\
M_2 := AC \\
X := M_1 + M_2 + AD \\
M_3 := M_1 + M_2 \\
X := M_3 + AD
\]
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]
\[ M_1 := AB \]
\[ X := M_1 + AC + AD \]
\[ M_2 := AC \]
\[ X := M_1 + M_2 + AD \]
\[ M_3 := M_1 + M_2 \]

\[ X := M_3 + AD \]

\[
\begin{array}{c|c}
\text{tmp} & \text{expr} \\
M_1 & AB \\
M_2 & AC \\
M_3 & AB + AC \\
\end{array}
\]

\[ M_1 + M_2 \Leftrightarrow AB + AC \]
Reducing Redundancy

\[
X := AB + AC + AD \\
M_1 := AB \\
X := M_1 + AC + AD \\
M_2 := AC \\
X := M_1 + M_2 + AD \\
M_3 := M_1 + M_2 \\
X := M_3 + AD
\]
Reducing Redundancy

\[ X := AB + AC + AD \]

\[ M_1 := AB \]

\[ X := M_1 + AC + AD \]

\[ M_2 := AC \]

\[ X := M_1 + M_2 + AD \]

\[ M_3 := M_1 + M_2 \]

\[ X := M_3 + AD \]

\[ \text{tmp expr} \]

\[ M_1 \quad AB \]

\[ M_2 \quad AC \]

\[ M_3 \quad AB + AC \]

\[ X := A(B + C + D) \]
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]
\[ M_1 := AB \]
\[ X := M_1 + AC + AD \]
\[ M_2 := AC \]
\[ X := M_1 + M_2 + AD \]
\[ M_3 := M_1 + M_2 \]
\[ X := M_3 + AD \]

<table>
<thead>
<tr>
<th>tmp</th>
<th>expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>( AB )</td>
</tr>
<tr>
<td>( M_2 )</td>
<td>( AC )</td>
</tr>
<tr>
<td>( M_3 )</td>
<td>( AB + AC )</td>
</tr>
<tr>
<td>( M_4 )</td>
<td>( B + C )</td>
</tr>
</tbody>
</table>

\[ X := A(B + C + D) \]
\[ M_4 := B + C \]
\[ X := A(M_4 + D) \]
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]
\[ M_1 := AB \]
\[ X := M_1 + AC + AD \]
\[ M_2 := AC \]
\[ X := M_1 + M_2 + AD \]
\[ M_3 := M_1 + M_2 \]
\[ X := M_3 + AD \]

<table>
<thead>
<tr>
<th>tmp</th>
<th>expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_1)</td>
<td>(AB)</td>
</tr>
<tr>
<td>(M_2)</td>
<td>(AC)</td>
</tr>
<tr>
<td>(M_3)</td>
<td>(AB + AC)</td>
</tr>
<tr>
<td>(M_4)</td>
<td>(B + C)</td>
</tr>
</tbody>
</table>

\[ X := A(B + C + D) \]
\[ M_4 := B + C \]
\[ X := AM_4 + AD \]
Reducing Redundancy

\[ X := AB + AC + AD \]
\[ M_1 := AB \]
\[ X := M_1 + AC + AD \]
\[ M_2 := AC \]
\[ X := M_1 + M_2 + AD \]
\[ M_3 := M_1 + M_2 \]
\[ X := M_3 + AD \]

\[ M_4 := B + C \]
\[ X := AM_4 + AD \]
\[ M_3 := AM_4 \]
\[ X := M_3 + AD \]

\[ X := A(B + C + D) \]
\[ M_4 := B + C \]
\[ X := AM_4 + AD \]
\[ M_3 := AM_4 \]
\[ X := M_3 + AD \]
Reducing Redundancy

\[
\begin{align*}
X &:= AB + AC + AD \\
M_1 &:= AB \\
X &:= M_1 + AC + AD \\
M_2 &:= AC \\
X &:= M_1 + M_2 + AD \\
M_3 &:= M_1 + M_2 \\
X &:= M_3 + AD
\end{align*}
\]

<table>
<thead>
<tr>
<th>tmp</th>
<th>expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>( AB )</td>
</tr>
<tr>
<td>( M_2 )</td>
<td>( AC )</td>
</tr>
<tr>
<td>( M_3 )</td>
<td>( AB + AC )</td>
</tr>
<tr>
<td>( M_4 )</td>
<td>( B + C )</td>
</tr>
</tbody>
</table>

\[
AM_4 \Leftrightarrow A(B + C)
\]

\[
\begin{align*}
X &:= A(B + C + D) \\
M_4 &:= B + C \\
X &:= AM_4 + AD \\
M_3 &:= AM_4 \\
X &:= M_3 + AD
\end{align*}
\]
Derivation Graph

Reducing Redundancy

\[ X := AB + AC + AD \]

\[ M_1 := AB \]

\[ X := M_1 + AC + AD \]

\[ M_2 := AC \]

\[ X := M_1 + M_2 + AD \]

\[ M_3 := M_1 + M_2 \]

\[ X := M_3 + AD \]

\[ \text{tmp expr} \]

| \( M_1 \) | \( AB \) |
| \( M_2 \) | \( AC \) |
| \( M_3 \) | \( AB + AC \) |
| \( M_4 \) | \( B + C \) |

\[ X := A(B + C + D) \]

\[ M_4 := B + C \]

\[ X := AM_4 + AD \]

\[ M_3 := AM_4 \]

\[ X := M_3 + AD \]
Exhaustive

\[ X := ABC + D \]
\[ T_1 := AB \]
\[ X := T_1 C + D \]
\[ T_3 := T_1 C + D \]
\[ X := T_3 \]
\[ T_4 := T_1 C \]
\[ X := T_4 + D \]
\[ T_5 := T_4 + D \]
\[ X := T_5 \]

Constructive

\[ X := ABC + D \]
\[ T_1 := AB \]
\[ T_2 := BC \]
\[ X := AT_2 + D \]
\[ X := T_2 + D \]
\[ T_3 := T_2 + D \]
\[ X := T_3 \]
Results

Example: \( w := AB^{-1}c \)

Naive
\[ w = A \times \text{inv}(B) \times c \]

Recommended
\[ w = A \times (B \setminus c) \]

Generated

```plaintext
ml0 = A; ml1 = B; ml2 = c;
potrf!('L', ml1)
trsv!('L', 'N', 'N', ml1, ml2)
trsv!('L', 'T', 'N', ml1, ml2)
ml3 = Array{Float64}(1000)
gemv!('N', 1.0, ml0, ml2, 0.0, ml3)
w = ml3
```
Results

1 Thread

![Graph showing speedup of Linnea](image_url)

Test problems:
- JL n
- JL r
- Arma n
- Arma r
- Eig n
- Eig r
- Mat n
- Mat r
Results

24 Threads

![Graph showing speedup of Linnea across various test problems with 24 threads. The x-axis represents test problems, and the y-axis represents speedup of Linnea.]
Results

Solutions over Time

![Chart showing solutions over time with relative cost and time in logarithmic scale.](chart.png)
Results

- 2x Intel Haswell E5-2680 v3.
  - 24 cores.
  - 64 GB RAM.
  - Turbo Boost is disabled.
- Setup:
  - 20 repetitions.
  - We compute the confidence interval of the median \([HB15]\).
  - Cold cache.
- 25 application problems.
  - Domains: statistics, signal processing, image processing, optimization, regularization, linear algebra algorithms.
- 100 randomly generated problems.
  - Between 4 and 7 operands.
  - Sizes between 50 and 2000.
  - Properties: diagonal, lower/upper triangular, symmetric, SPD.
  - No common subexpressions.
References


Linnea is available online: https://github.com/HPAC/linnea