The Matrix Chain Algorithm to Compile Linear Algebra Expressions

Henrik Barthels

HPAC High Performance and Automatic Computing

RWTH AACHEN UNIVERSITY

Introduction

Plain C

```
for (j = 0; j < m; j++)
   for (p = 0; p < k; p++)
      for (i = 0; i < n; i++)
         C[i][j] = A[i][p]*B[p][j];
```

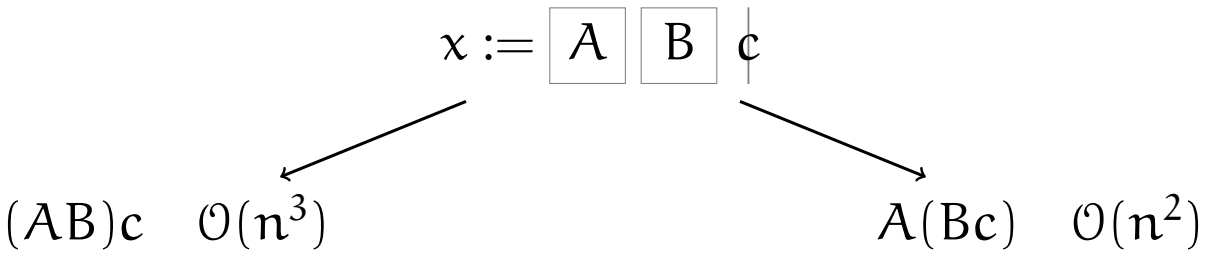
Libraries: BLAS [DDC+90]

```
dgemm("N", "N", &M, &N, &K, &alpha, &A, &ldA, &B, &ldB, &beta, &C,
      &ldC);
```

Matlab

```
C = A*B
```

The Matrix Chain Algorithm



- $X := ABCDE$
- Dynamic programming $\mathcal{O}(n^3)$ algorithm [CRL90].
- In practice:

$$X := AB^TC^{-T}D$$

Goals

- Simplicity of Matlab.
- High performance:
  - $y \leftarrow Ax$
  - $C \leftarrow AB$
  - $C \leftarrow A^{-1}B$

The Generalized Matrix Chain Algorithm

Input Grammar

$$
\begin{aligned}
\text{chain} &\rightarrow \text{factor} \cdot \text{chain} \mid \text{factor} \\
\text{factor} &\rightarrow \text{op} \mid \text{op}^{\mathrm{T}} \mid \text{op}^{-1} \mid \text{op}^{-\mathrm{T}} \\
\text{op} &\rightarrow \textit{symbol} \mid \textit{symbol}_{\text{indices}} \\
\text{indices} &\rightarrow \textit{index}\ \text{indices} \mid \textit{index}
\end{aligned}
$$

$$
\begin{aligned}
\text{definitions} &\rightarrow \text{definition definitions} \mid \text{definition} \\
\text{definition} &\rightarrow \textbf{Matrix}\ \textit{name}\ \text{size}\ \langle \text{property}^{*} \rangle \\
\text{definition} &\rightarrow \textbf{Index}\ \textit{name range} \\
\text{size} &\rightarrow (\textit{rows}, \textit{columns}) \\
\text{property} &\rightarrow \textbf{LowerTriangular} \mid \textbf{Orthogonal} \mid \ldots
\end{aligned}
$$

The Generalized Matrix Chain Algorithm

Algorithm

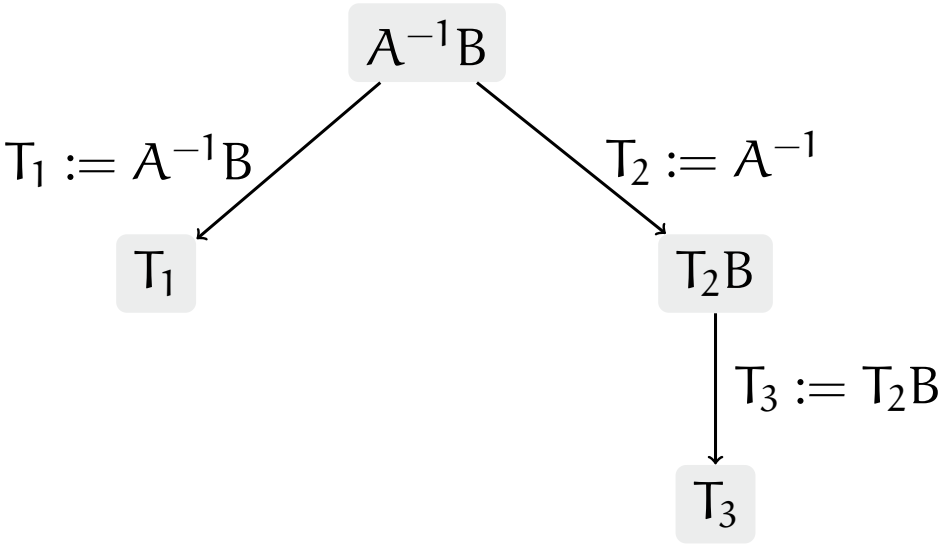
$$\text{expr1} \cdot \text{expr2}$$

- Standard: expr $= A$.
- Generalized: expr $= A, A^T, A^{-1}, A^{-T}$.
- With unary operators:

$$AB^T \qquad A^{-1}B \qquad A^{-1}B^{-T} \qquad \dots$$

The Generalized Matrix Chain Algorithm

CLAK [FTB13]

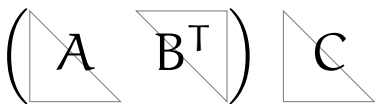


The Generalized Matrix Chain Algorithm

Properties

Operation	Cost
A^{-1} B	n^3
A^{-1} B	$2.7n^3$

Inference of Properties



$$W = AB^T$$

Operation	Cost
W B	n^3
W B	$\frac{1}{3}n^3$

HPAC High Performance and Automatic Computing | RWTH AACHEN UNIVERSITY

Conclusion

Results

$$W := A^{-1}BCD^{-T}EF$$

Naive implementation

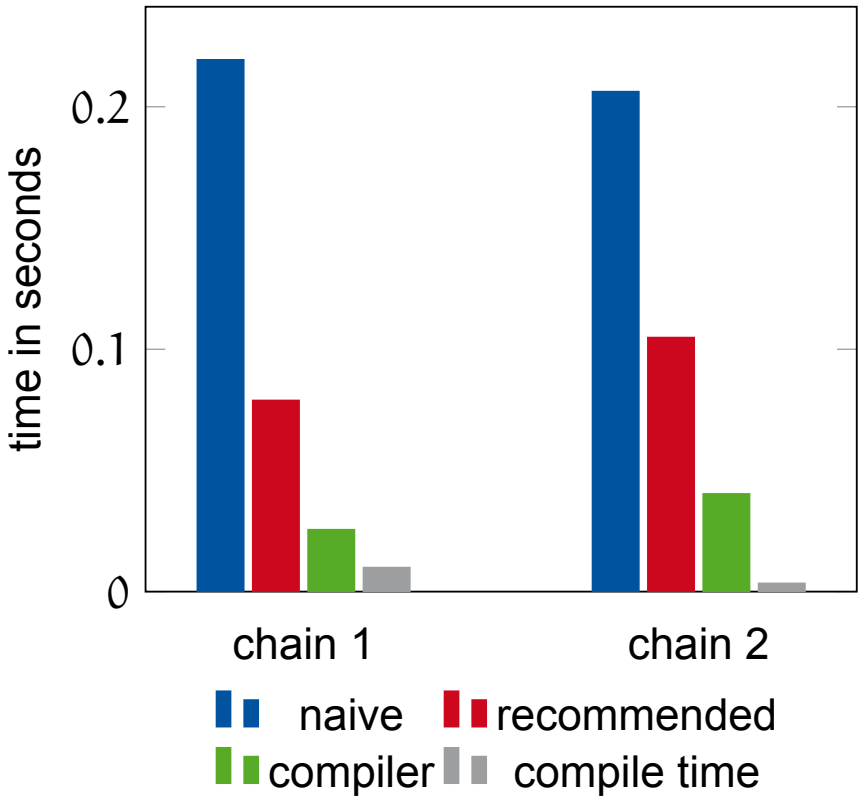
```
W = inv(A)*B*C*inv(D)'*E*F;
```

Recommended implementation

```
W = (((A\B)*C)/D')*E*F;
```

Compiler implementation

```
t1 = linsolve(A, B, opts1);
t3 = linsolve(D, C', opts2)';
t8 = t3*E;
t12 = t8*F;
W = t1*t12;
```



Conclusion

> There should be one—and preferably only one—obvious way to do it.
>
> The Zen of Python

How to compute $A^{-1}B$?

Code	Cost
`inv(A)*B`	$4n^3$
`A\B`	$2.7n^3$

HPAC High Performance and Automatic Computing | RWTH AACHEN UNIVERSITY

References

[CRL90] Thomas H. Cormen, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill, Inc., 1990.

[DDC+90] Jack J. Dongarra, Jeremy Du Croz, et al. A set of Level 3 Basic Linear Algebra Subprograms. *ACM TOMS*, 16(1):1–17, 1990.

[FTB13] Diego Fabregat-Traver and Paolo Bientinesi. A Domain-Specific Compiler for Linear Algebra Operations. In *VECPAR 2010*, volume 7851 of *LNCS*, pages 346–361. Springer, 2013.