Efficiently Mapping Linear Algebra to High-Performance Code

Christos Psarras[1] Henrik Barthels[1] Paolo Bientinesi[2]

[1]Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen University
[2]Department of Computing Science, Umeå Universitet

The Linear Algebra Mapping Problem

In the domain of numerical linear algebra, significant effort is put into optimizing low-level libraries such as BLAS and LAPACK. However, we observe a decrease in the number of users that actually go through the tedious, error-prone and time consuming process of using directly said libraries by writing their code in C or Fortran; instead, languages and libraries such as Matlab, Julia, Eigen and Armadillo, which offer a higher level of abstraction, are becoming more and more popular. These languages and libraries allow users to input a linear algebra problem as an expression which closely resembles the mathematical description, for example:

Stochastic Newton [2] $B_k := \frac{k}{k-1} B_{k-1}(I_n - A^T W_k((k-1)I_l + W_k^T A B_{k-1} A^T W_k)^{-1} W_k^T A B_{k-1})$

Signal Processing [3] $x := \left(A^{-T} B^T B A^{-1} + R^T L R\right)^{-1} A^{-T} B^T B A^{-1} y$

These expressions are then internally mapped to lower level building blocks such as BLAS and LAPACK. Unfortunately, our experience suggests that this translation frequently results in **suboptimal code**. We investigate how well popular high-level languages and libraries translate expressions to code [5].

Example of a LAMP

Least Squares $b := (X^T X)^{-1} X^T y$, where $X \in \mathbb{R}^{n \times m}$, $y \in \mathbb{R}^{n \times 1}$, $n = 2500$, $m = 500$

Given the Least Squares expression, what sequence of BLAS/LAPACK calls would one use to compute it?

Solution 1

1. K := GEMM(X^T, X)
2. K := GETRF(K)
3. K := GETRI(K)
4. C := GEMM(K, X^T)
5. b := GEMV(C, y)

⊖⊖ Performance
⊖⊖ Stability

Solution 2

1. K := SYRK(X^T)
2. b := GEMV(X^T, y)
3. K := POTRF(K)
4. b := POTRS(K, b)

⊕⊕ Performance
⊖ Stability

Solution 3

1. Q, R := GEQRF(K)
2. b := ORMQR(Q, y)
3. b := TRSM(R, b)

⊕ Performance
⊕⊕ Stability

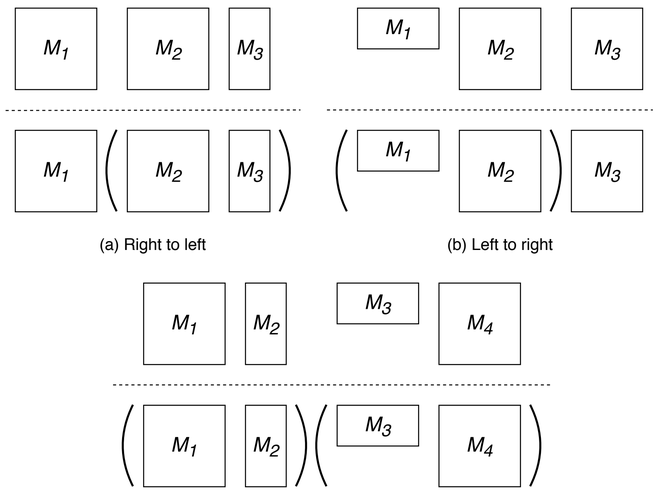
Kernel Invocation

We investigate the ability of modern linear algebra languages and libraries to identify and make proper use of BLAS kernels.
The use of the appropriate BLAS kernel for an operation can

- Significantly reduce number of FLOPS
- Reduce total amount of memory
- Increase efficiency on target machine
- Make use of matrix properties

Name	Expression	C	Armadillo	Eigen	Julia	Matlab	Numpy	R
GEMM	$C = AB$	0.27	0.29 ✓	0.29 ✓	0.30 ✓	0.29 ✓	0.29 ✓	0.31 ✓
SYRK	$C = AA^T$	0.14	0.17 ✓	0.29 –	0.21 ✓	0.18 ✓	0.18 ✓	0.32 –
SYR2K	$C = AB^T + BA^T$	0.28	0.57 –	0.58 –	0.69 –	0.57 –	0.58 –	0.59 –

Matrix Chain [1]



(a) Right to left (b) Left to right

(c) mixed

Computation	Armadillo	Eigen	Julia	Matlab	Numpy	R
Right-to-Left	✓	–	–	–	–	–
Left-to-Right	✓	✓	✓	✓	✓	✓
Mixed	–	–	–	–	–	–

Matrix Properties

Operation	Property	C	Arma	Eigen	Julia	Matlab	NumPy	R
Linear System	Symmetric	0.463	–	*n.a.*	–	–	–	–
	SPD	0.316	✓	*n.a.*	–	✓	–	–
	Triangular	0.031	–	*n.a.*	✓	✓	–	–
	Diagonal	0.001	♣	*n.a.*	✓	♣	–	–
Multiplication	General	1.461						
	Triangular	0.748	–	–	◇	–	–	–
	Diagonal	0.064	–	–	◇	–	–	–

Depending on the domain, expressions might contain operands with specific shapes (ex. triangular, diagonal) or properties (ex. positive, symmetric). These properties are particularly important during the evaluation of said expressions, as they enable techniques that often yield better **performance** and **numerical accuracy**.
Solution 2 of the LAMP example takes advantage of the Symmetric Positive Definite (SPD) property of the intermediate matrix K and uses the Cholesky factorization, which is twice as fast as a general case LU factorization.

Common Subexpression Elimination

$b := A^{-T} B^T B A^{-1} y$, where $A, B \in \mathbb{R}^{n \times n}$, $y \in \mathbb{R}^{n \times 1}$, $n = 2000$

Given the expression above, what sequence of BLAS/LAPACK calls would one use to compute it?

Solution 1

1. K := GESV(A^T, B^T)
2. C := SYRK(K)
3. b := GEMV(C, y)

Cost: $\frac{11}{3}n^3 + n^2$

Solution 2

1. K := GESV(A^T, B^T)
2. b := GEMV(K, y)
3. b := GEMV(K^T, b)

Cost: $\frac{8}{3}n^3 + 4n^2$

Solution 3

1. b := GESV(A, y)
2. b := GEMV(B, b)
3. b := GEMV(B^T, b)
4. b := GESV(A^T, b)

Cost: $\frac{4}{3}n^3 + 8n^2$

$b := A^T B^T B A y$, where $A, B \in \mathbb{R}^{n \times n}$, $y \in \mathbb{R}^{n \times 1}$, $n = 2000$

Given the expression above, what sequence of BLAS/LAPACK calls would one use to compute it?

Solution 1

1. K := GEMM(A^T, B^T)
2. C := SYRK(K)
3. b := GEMV(C, y)

Cost: $3n^3 + 2n^2$

Solution 2

1. K := GEMM(A^T, B^T)
2. b := GEMV(K^T, y)
3. b := GEMV(K, b)

Cost: $2n^3 + 4n^2$

Solution 3

1. b := GEMV(A, y)
2. b := GEMV(B, b)
3. b := GEMV(B^T, b)
4. b := GEMV(A^T, b)

Cost: $8n^2$

Matrix Inversion

Should languages substitute inversion with solving a linear system?

```
inv(A) * b    vs    A \ b
```

Explicitly inverting a matrix is **slow** and **unstable** [4, p. 260].

	Armadillo	Eigen	Julia	Matlab	Numpy	R
Inversion Substitution	✓	–	–	–	–	–

Blocked Matrices

$$\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}^{-1} B = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}^{-1} \begin{bmatrix} B_T \\ B_B \end{bmatrix} = \begin{bmatrix} A_1^{-1} B_T \\ A_2^{-1} B_B \end{bmatrix}$$

While all languages support methods for instantiating such operands, they don't take advantage of the structure when it comes to computation, yielding poor performance.

	Armadillo	Eigen	Julia	Matlab	Numpy	R
Blocked Operands	–	–	–	–	–	–

References

[1] H. Barthels et al. The Generalized Matrix Chain Algorithm. In *Proceeding of International Symposium on Code Generation and Optimization*, Feb 2018.

[2] J. Chung et al. Stochastic Newton and Quasi-Newton Methods for Large Linear Least-squares Problems. *CoRR*, math.NA, 2017.

[3] Y. Ding and I. W. Selesnick. Sparsity-Based Correction of Exponential Artifacts. *Signal Processing*, 120:236–248, 2016.

[4] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, 2002.

[5] Christos Psarras, Henrik Barthels, and Paolo Bientinesi. The Linear Algebra Mapping Problem. *CoRR*, abs/1911.09421, 2019.

Acknowledgment

Financial support from the **Deutsche Forschungsgemeinschaft (DFG)** through grant IRTG-2379 and the use of Lonestar from TACC are gratefully acknowledged.