

Math 4997-1

Lecture 9: Linear algebra with Blaze

Patrick Diehl 

<https://www.cct.lsu.edu/~pdiehl/teaching/2020/4997/>

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.



Reminder

Lecture 8

What you should know from last lecture

- ▶ Bond-based Peridynamics (Course project)

Vectors and Matrices

Vector space [1, 4]

A vector space (or a linear space) is a collection of so-called vectors $\mathbf{v} \in \mathbb{R}^n$. Vectors can be added or scaled (multiplied by a scalar value).

$$\begin{aligned}\mathbf{v} &= \{v_1, v_2, \dots, v_n\} \\ \mathbf{w} &= \{w_1, w_2, \dots, w_n\}\end{aligned}$$

Addition

$$\mathbf{v} + \mathbf{w} = \{v_1 + w_1, v_2 + w_2, \dots, v_n + w_n\}$$

Scaling

$$2\mathbf{v} = \{2v_1, 2v_2, \dots, 2v_n\}$$

Vector II

Column vector

$$\mathbf{v} = \begin{Bmatrix} v_1 \\ \vdots \\ v_n \end{Bmatrix}$$

Row vector

$$\mathbf{v}^T = \{v_1, \dots, v_n\}$$

Matrix

A matrix $\mathbf{A} \in \mathbb{R}^{n,m}$ has n rows and m columns

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & \dots & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{pmatrix}$$

Scaling

$$2\mathbf{A} = \begin{pmatrix} 2a_{1,1} & \dots & 2a_{1,m} \\ \vdots & \dots & \vdots \\ 2a_{n,1} & \dots & 2a_{n,m} \end{pmatrix}$$

Matrix II

Addition

$$\mathbf{A} + \mathbf{B} = \begin{pmatrix} a_{1,1} + b_{1,1} & \dots & a_{1,m} + b_{1,m} \\ \vdots & \dots & \vdots \\ a_{n,1} + b_{n,1} & \dots & a_{n,m} + b_{n,m} \end{pmatrix}$$

Matrix vector multiplication

$$\mathbf{A}\mathbf{v} = \begin{Bmatrix} a_{1,1} * b_1 + \dots + a_{1,m} * b_n \\ \vdots \\ a_{n,1} * b_1 + \dots + a_{n,m} * b_n \end{Bmatrix}$$

Applications

Communication

We have a group of people $P_1, \dots, P_n$, if person P_1 has contact with person P_2 , we can model this information by setting the matrix element $a_{1,2} = 1$. By doing this for all people in our group, we will get some matrix

$$\mathbf{M} = \begin{Bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{Bmatrix}$$

This matrix will tell us that P_1 has contact with P_2 and P_4 , P_2 with P_3 and so on.

Now we define

$$\mathbf{M}^4 = \mathbf{M} \cdot \mathbf{M} \cdot \mathbf{M} \cdot \mathbf{M},$$

which means for $\mathbf{M}^n$, we have to do n multiplications of $\mathbf{M}$.

Communication

We now can compute

$$M^2 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

and see that person P_3 can send some message to Person P_2 in two cycles.

Blaze

Blaze¹

Blaze is an open-source, high-performance C++ math library for dense and sparse arithmetic. With its state-of-the-art Smart Expression Template implementation Blaze combines the elegance and ease of use of a domain-specific language with HPC-grade performance, making it one of the most intuitive and fastest C++ math libraries available.

More details about the implementation details [2, 3].

¹ <https://bitbucket.org/blaze-lib/blaze/src/master/>

Installation²

CMake

```
tar -xvf blaze-3.6.tar.gz
cd blaze-3.6
cmake -DCMAKE_INSTALL_PREFIX=/home/patrick/blaze .
make install
```

Manual

```
tar -xvf blaze-3.6.tar.gz
cd blaze-3.6
cp -r ./blaze /home/patrick/blaze
```

²<https://bitbucket.org/blaze-lib/blaze/wiki/Configuration%20and%20Installation>

Compilation³

CMake

```
find_package( blaze )  
if( blaze_FOUND )  
    add_library( blaze_target INTERFACE )  
    target_link_libraries( blaze_target  
                           INTERFACE blaze::blaze )  
endif()
```

Compiler

```
g++ -I/home/diehlpk/blaze BlazeTest.cpp
```

³<https://bitbucket.org/blaze-lib/blaze/wiki/Configuration%20and%20Installation>

Parallelism⁴

C++11 Thread Setup

Add following arguments to the compiler

```
-std=c++11 -DBLAZE_USE_CPP_THREADS
```

and set the number of threads

```
export BLAZE_NUM_THREADS=4    // Unix systems
```

HPX

Add following arguments to the compiler

```
-DBLAZE_USE_HPX_THREADS
```

and set the number of threads

```
./a.out --hpx:threads=4
```

⁴<https://bitbucket.org/blaze-lib/blaze/wiki/Shared%20Memory%20Parallelization>

Blaze's API

Vector⁵

```
using blaze::DynamicVector;  
using blaze::columnVector;  
using blaze::rowVector;  
  
// Setup of the 3-dimensional dense column vector  
DynamicVector<int,columnVector> a{ 1, 2, 3 };  
  
// Setup of the 3-dimensional dense row vector  
DynamicVector<int,rowVector> b{ 4, 5, 6 };  
  
// Instantiation of a 3-dimensional column vector  
blaze::DynamicVector<int> c( 3UL );  
  
// Set all elements of the vector c to 5  
c = 5;
```

Vector operations⁶

```
// Get the size of the vector
auto size = c.size();

// Access the i-th element
auto value = c[i];

// Loop over the vector
for( size_t i=0UL; i< c.size(); ++i )
    std::cout << c[i] << std::endl;

// Iterate over a vector
blaze::CompressedVector<int> d{ 0, 2, 0, 0};
for( CompressedVector<int>::Iterator it=d.begin();
    it!=d.end(); ++it )
    std::cout << it->value() << std::endl;
```

Vector operations II⁷

```
blaze::DynamicVector<double> a, b;  
  
// Computes the sine of each element of the vector  
b = sin( a );  
// Computes the base e exponential of each element  
b = exp( a );  
// Computes the exponential value of each element  
b = pow( a, 1.2 );  
// Computes the absolute value of each element  
b = abs( a );  
// Complex numbers  
using blaze::StaticVector;  
using cplx = std::complex<double>;  
StaticVector<cplx,1UL> a{ cplx(-2.0,-1.0)};  
double b = imag( a ); //Get the imaginary part
```

Dense Matrix⁸

```
// Definition of a 3x4 matrix  
// Values are not initialized  
blaze::DynamicMatrix<int> A( 3UL, 4UL );  
  
// Definition of a 3x4 matrix  
// with 0 rows and columns  
blaze::StaticMatrix<int,3UL,4UL> A;  
  
// Definition of column-major matrix  
// with 0 rows and columns  
blaze::DynamicMatrix<double,blaze::columnMajor> C;
```

Remarks:

- ▶ Default is row-major matrices:
- ▶ Static Matrix are small and size known at compile time

⁸ <https://bitbucket.org/blaze-lib/blaze/wiki/Matrix%20Types>

Sparse matrix

```
// Definition of a 3x4 integral row-major matrix  
blaze::CompressedMatrix<int> A( 3UL, 4UL );  
  
// Definition of a 3x3 identity matrix  
blaze::IdentityMatrix<int> A( 3UL );  
  
// Definition of a 3x5 zero matrix  
blaze::ZeroMatrix<int> A( 3UL, 5UL );
```

Sparse matrices are used, if you have only few non-zero entries.

Matrix operation⁹

```
// Access elements
```

```
M1(0,0) = 1;
```

```
// Total amount of elements
```

```
size( M2 );
```

```
// Number of rows
```

```
M2.rows();
```

```
// Number of columns
```

```
M2.columns();
```

```
// Computes the element-wise absolute value
```

```
abs( A );
```


Matrix operation II¹⁰

```
// Traversing the matrix
blaze::CompressedMatrix<int> M1( 4UL, 4UL );

for( size_t i=0UL; i<M1.rows(); ++i ) {
    for( size_t j=0UL; j<M1.columns(); ++j ) {
        ... = M1(i,j);
    }
}

// Traversing the matrix by Iterator
for( size_t i=0UL; i<A.rows(); ++i ) {
    for( CompressedMatrix<int,rowMajor>::Iterator it=
        A.begin(i); it!=A.end(i); ++it ) {
        it->value() = ...;
    }
}
```

Arithmetic operation¹¹

```
blaze::StaticVector<int,3UL> v1{ 3, 2, 5, -4, 1, 6 };  
// Addition  
blaze::StaticVector<int,3UL> res = v1 + v1;  
// Subtraction  
blaze::StaticVector<int,3UL> res = v1 - 2 * v1;  
  
blaze::DynamicMatrix<float,rowMajor> M1( 7UL, 3UL );  
// Addition  
blaze::DynamicMatrix<float,rowMajor> res = M1 + M1;  
// Subtraction  
blaze::DynamicMatrix<float,rowMajor> res = 2*M1 - M1;
```

¹¹<https://bitbucket.org/blaze-lib/blaze/wiki/Arithmetic%20Operations>

Matrix decomposition¹²

```
blaze::DynamicMatrix<double,blaze::rowMajor> A;  
// ... Resizing and initialization  
  
blaze::DynamicMatrix<double,blaze::rowMajor> L, U, P;  
  
// LU decomposition of a row-major matrix  
lu( A, L, U, P );  
  
assert( A == L * U * P );
```

Decompositions

- ▶ Cholesky
- ▶ QR/RQ
- ▶ QL/LQ

¹²<https://bitbucket.org/blaze-lib/blaze/wiki/Matrix%20operations#!matrix-decomposition>

Eigen values^{13,14}

```
// The symmetric matrix A
SymmetricMatrix< DynamicMatrix<double,rowMajor>>
    A( 5UL, 5UL );
// ... Initialization

// The vector for the real eigenvalues
DynamicVector<double,columnVector> w( 5UL );
// The matrix for the left eigenvectors
DynamicMatrix<double,rowMajor>      V( 5UL, 5UL );

eigen( A, w, V );
```

Adapters may be more efficient and less memory consuming.

¹³ <https://bitbucket.org/blaze-lib/blaze/wiki/Matrix%20Operations#!eigenvalueseigenvectors>

¹⁴ <https://bitbucket.org/blaze-lib/blaze/wiki/Symmetric%20Matrices>

Summary

Summary

After this lecture, you should know

- ▶ Vectors and matrices
- ▶ How to use Blaze for matrix and vector operations
- ▶ How to compile a program using a external library

References

References I

- [1] Jim Hefferon.
Linear algebra, released under the gnu free documentation license.
- [2] K. Iglberger, G. Hager, J. Treibig, and U. Rde.
Expression templates revisited: A performance analysis of current methodologies.
SIAM Journal on Scientific Computing,
34(2):C42–C69, 2012.
- [3] K. Iglberger, G. Hager, J. Treibig, and U. Rde.
High performance smart expression template math libraries.
In *2012 International Conference on High Performance Computing Simulation (HPCS)*, pages 367–373, July 2012.

References II

[4] John T Scheick.

Linear algebra with applications, volume 81.

McGraw-Hill New York, 1997.