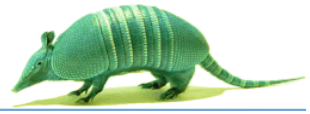
RWTH AACHEN UNIVERSITY

Armadillo
C++ linear algebra library

Umair Muslim
341318

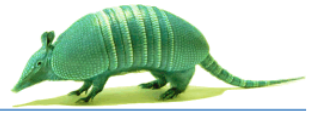
Overview



- Introduction
- Libraries comparison
- Features
- History
- API
  - Structural Representation
  - Functionality overview
- Evaluation
- Conclusion

Armadillo by Umair Muslim

Introduction

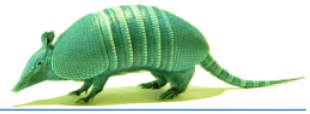


- A linear algebra library (matrix math's) with high quality syntax
- Well balanced between speed and syntax simplicity.
- Similar syntax as Matlab
- Why not Matlab?
  - Proprietary program
  - Weak dynamically typed
  - Cross platform inconsistency

Comparison with other Libraries

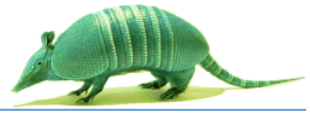
- Newmat
  - Unclear license
  - Reimplementation instead of reuse of LAPACK
  - Lack high-performance
- uBLAS (Boost C++ lib)
  - Less availability of basic functionality (e.g. Matrix inversion)
- IT++ (ITPP)
  - Use restrictive GNU General Public License (GPL)
    - Any library under it becomes infected with GPL
    - Software must come with source code

Features



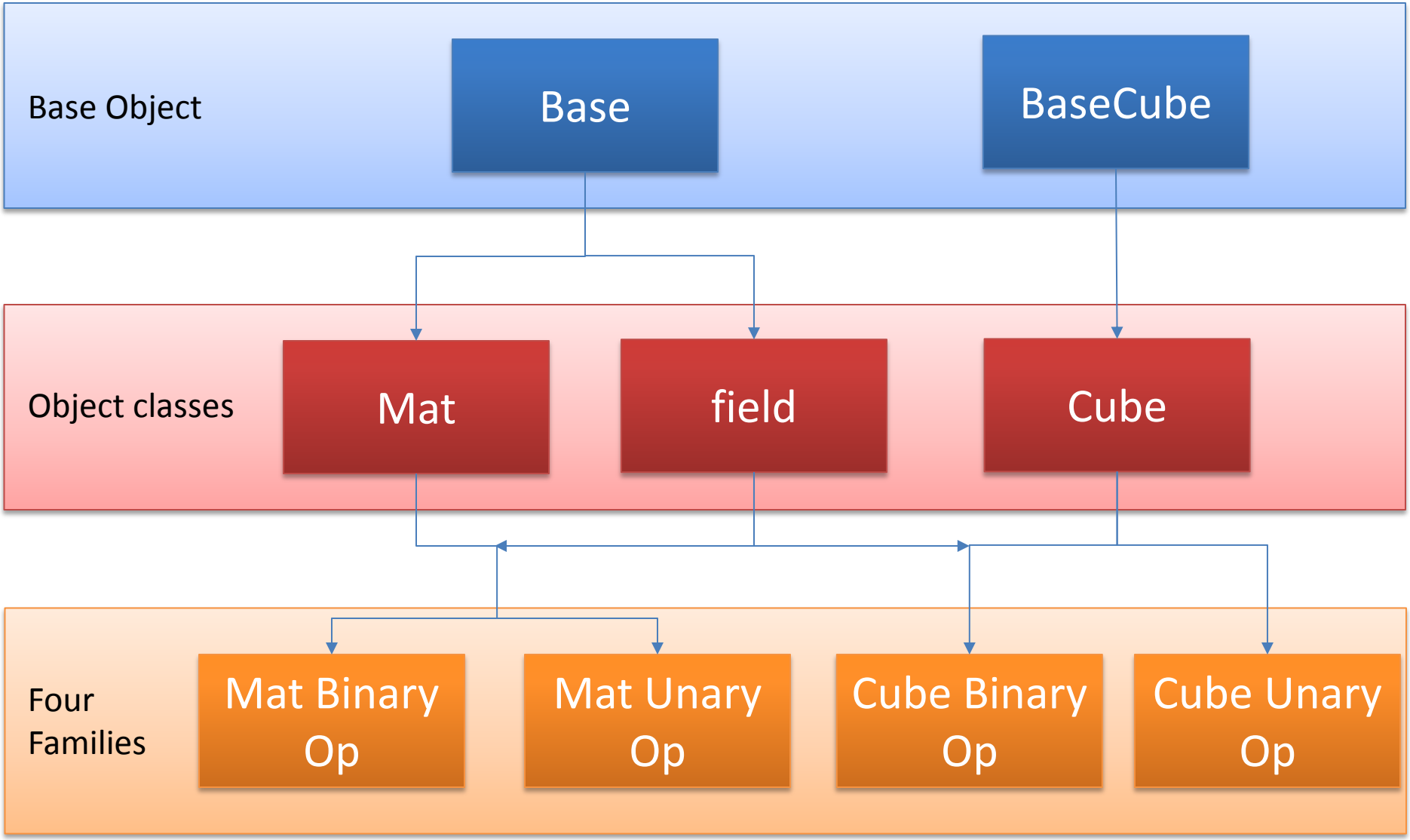
- Simple syntax: Matlab or Octave.
- Numerical type support.
- Based on LAPACK library.
- Fast matrix manipulation using template meta-programing.
- Matrix save and load in file as Matlab
- Interfacing with other libraries: via STL-iterators
- Open source development
- Cross-platform usability

History



- Version 1.x
  - Added some basic functions: .min(), .max(), .floor(), save/load
- Version 2.x
  - Support for C++11
  - Exception handling: solve(), svd(), pinv(), syl() etc, std::runtime_error
- Version 3.x
  - Added Constants: datum class.
  - Template meta-programming, matrix addition etc.
- Version 5.x
  - Auto handling of 64 bit Integers
- Version 6.x
  - Improved functionality while using Intel MKL, ATLAS & OpenBLAS, like norm(), accu() etc.

Structural Representation

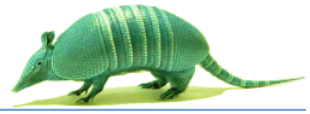


Armadillo by Umair Muslim

Functionality overview

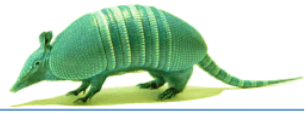
- Base datatypes
  - Mat<*type*> : mat
  - Col<*type*> : colvec, vec
  - Row<*type*> : rowvec
  - Cube<*type*> : cube
  - Field<*type*> : field
- Operators: + - * / % == != <= >= < >
- Functions: .transform(), .fill(), .diag(), .is_empty(), .print(), .save()/.load() . . . etc.
- Generators: zeros, eye, ones, rand, randi, randu, . . . etc.

Functionality overview



- Optimizations:
  - SIMD vectorization
    - SSE2
    - Elementary expressions (matrix addition, multiplication by scaler etc.)
    - Using GCC 4.7+ with -03.
    - SSE3, SSE4 or AVX, -march=native
  - Lazy evaluation
    - Template meta-programming
    - Evaluating mathematical expressions
    - trans (X) => Op< Mat, op_trans >
    - trans (square (X)) => Op< eOp< Mat, eop_square >, op_trans >

Example 1 - PageRank



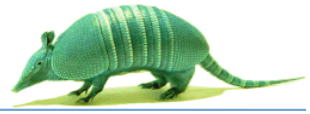
```
//creating Adjency matrix
mat A = zeros(size(graph));

for (; i<n; i++)
{
    //get the out degree of node i
    double out_degree = sum(graph.row(i));
    if(out_degree == 0) {
        A.row(i) = rowvec(n).fill( 1/(double)n );
    }
    else
        A.row(i) = graph.row(i)/out_degree;
}
```

```
mat A;

A.load("A.mat", arma_ascii);
calculatePageRank(A, 0.85, 0.00000001);
```

```
void calculatePageRank(mat graph, double d, double epsilon)
{
  //iterative PageRank calculation
  do {
      if(!firstTime)
          rank_old = rank;
      rank = (1-d) * e + d * A.t() * rank_old;
      i++;
      firstTime = false;
  } while( !(norm(rank-rank_old,1) < epsilon) );

  cout << "Graph: \n" << graph << "\n\n";
```

```
root@ubuntu:~/workspace/PageRank# ./page_rank
Graph:
         0         0    1.0000    1.0000
    1.0000         0         0    1.0000
         0    1.0000    1.0000    1.0000
         0         0         0         0


Adjency Matrix:
         0         0    0.5000    0.5000
    0.5000         0         0    0.5000
         0    0.3333    0.3333    0.3333
    0.2500    0.2500    0.2500    0.2500


After 110 iterations, page ranks :
   0.7676
   0.7533
   1.0795
   1.3996
```

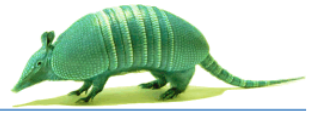
Evaluation



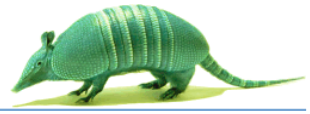
- Speedup factor of Armadillo relative to other software
- A, B, C, D and Q are NxN matrices, where N=500.
- g++ example.cpp –o example –O2 –fwhole-program –l lib

Operation	Formula	Matlab	Newmat	IT++
Add & scalar mult.	Q=0.1*A + 0.2*B + 0.3*C;	2.9	5.6	4.5
Traspose, matrix mult	Q = Q + 0.1*A' * 0.2 * B;	1.0	3.2	1.1
Submatrix copy	A(2:N, 2:N) = B(1:N-1, 1:N-1);	3.6	7.8	8.5
Direct element access	for c = 1:N for r = 1:N Q(r,c) = A(N+1-r, c)+B(r, N+1-c) + C(N+1-r, N+1-c); end End	14.7	5.1	2.0

Conclusion



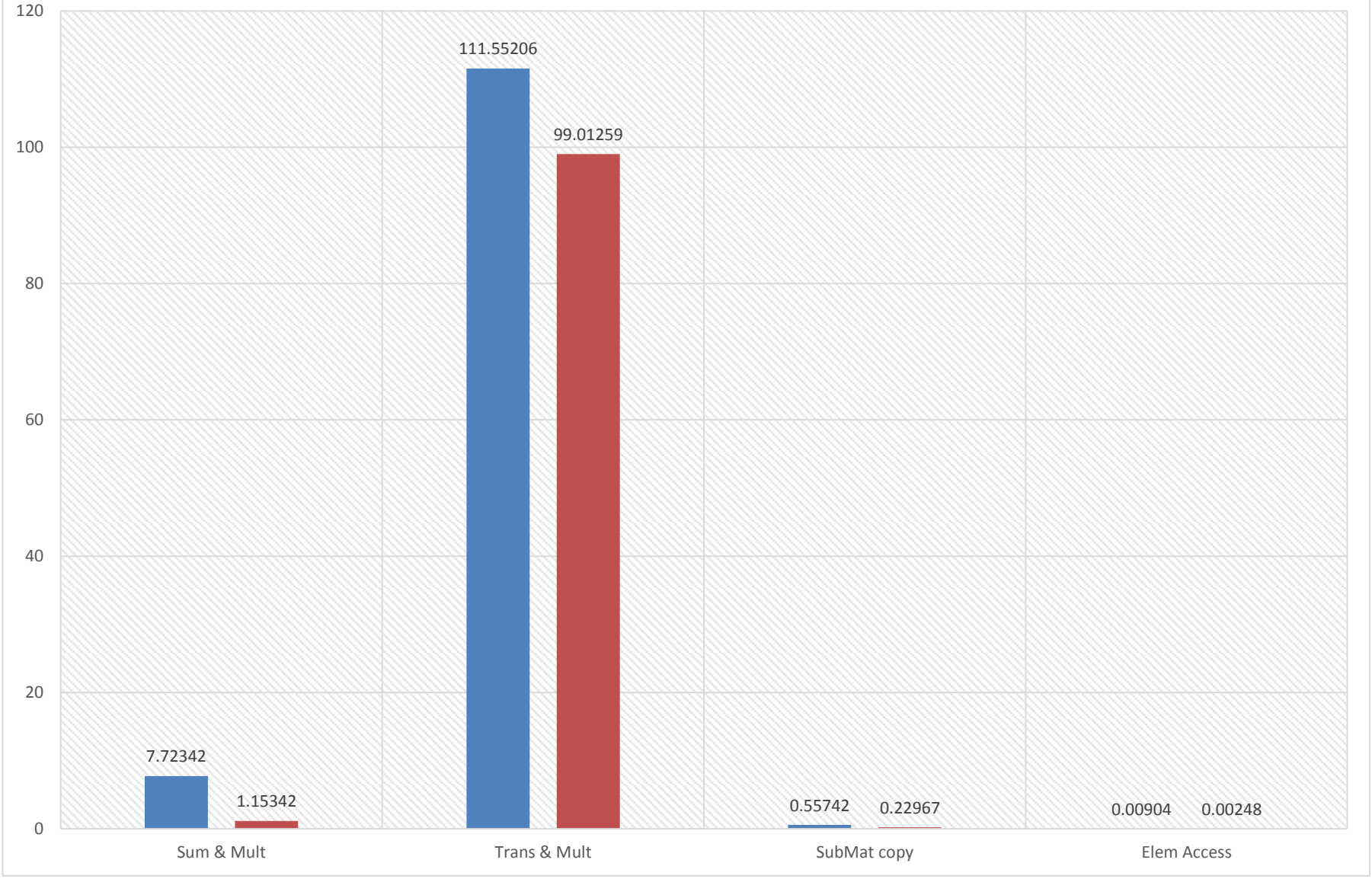
- Problems with Matlab and other linear algebra libraries.
- Provides balance between speed and syntax simplicity.
- Template meta-programming.
- Perform faster than Matlab, IT++ and Newmat libraries.



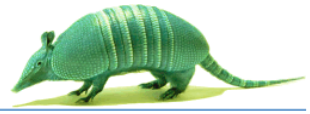
Question please.

THANK YOU.

Armadillo Optimization

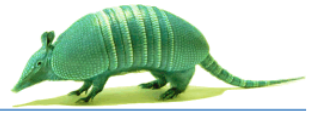


Armadillo Evaluation Imp



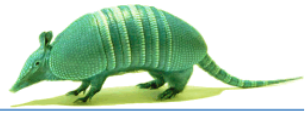
```
1  void add_scale_mul(mat &A, mat &B, mat &C, mat &D) {
2      mat Q = 0.1 * A + 0.2 * B + 0.3 * C;
3  }
4
5  void tras_mult(mat &A, mat &B, mat &C, mat &D) {
6      mat Q(size(A)); Q.fill(1.0);
7      Q = Q + 0.1 * A.t() * 0.2 * B;
8  }
9
10 void sub_copy(mat &A, mat &B, mat &C, mat &D) {
11     unsigned int N = A.n_cols;
12     A(span(2,N-1),span(2,N-1)) = B(span(1,N-2),span(1,N-2));
13 }
14
15 void elem_access(mat &A, mat &B, mat &C, mat &D) {
16     unsigned int N = A.n_cols;
17     mat Q(N,N);
18     int c=0, r=0;
19     for(; c<N; c++){
20         for(; r<N; r++){
21             Q.at(r,c) = A.at(N-1-r,c) + B.at(r,N-1-c) + C.at(N-1-r,N-1-c);
22         }
23     }
24 }
```

IT++ Evaluation Imp



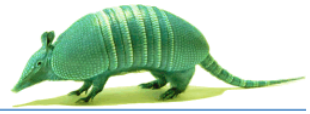
```
1  void add_scale_mul(mat &A, mat &B, mat &C, mat &D) {
2      mat Q = 0.1 * A + 0.2 * B + 0.3 * C;
3  }
4
5  void tras_mult(mat &A, mat &B, mat &C, mat &D) {
6      mat Q(A.rows(),A.rows()); Q = 1.0;
7      Q = Q + 0.1 * transpose(A) * 0.2 * B;
8  }
9
10 void sub_copy(mat &A, mat &B, mat &C, mat &D) {
11     unsigned int N = A.rows();
12     A(2,N-1,2,N-1) = B(1,N-2,1,N-2);
13 }
14
15 void elem_access(mat &A, mat &B, mat &C, mat &D) {
16     unsigned int N = A.rows();
17     mat Q(A.rows(),A.rows());
18     int c=0, r=0;
19     for(; c<N; c++){
20         for(; r<N; r++){
21             Q(r,c) = A(N-1-r,c) + B(r,N-1-c) + C(N-1-r,N-1-c);
22         }
23     }
24 }
```

Newmat Evaluation Imp



```
1  void add_scale_mul(Matrix &A, Matrix &B, Matrix &C, Matrix &D) {
2      Matrix Q = 0.1 * A + 0.2 * B + 0.3 * C;
3  }
4
5  void tras_mult(Matrix &A, Matrix &B, Matrix &C, Matrix &D) {
6      Matrix Q(A.nrows(),A.nrows()); Q = 1.0;
7      Q = Q + 0.1 * A.t() * 0.2 * B;
8  }
9
10 void sub_copy(Matrix &A, Matrix &B, Matrix &C, Matrix &D) {
11     unsigned int N = A.nrows();
12     A.submatrix(2,N-1,2,N-1) = B.submatrix(1,N-2,1,N-2);
13 }
14
15 void elem_access(Matrix &A, Matrix &B, Matrix &C, Matrix &D) {
16     unsigned int N = 500;
17     Matrix Q(N,N); Q = 0.0;
18     int c=1, r=1;
19     for(; c<=N; c++){
20         for(; r<=N; r++){
21             Q.submatrix(r,r,c,c) = A.submatrix((N+1-r),(N+1-r),c,c) +
22             B.submatrix(r,r,(N+1-c),(N+1-c)) +
23             C.submatrix((N+1-r),(N+1-r),(N+1-c),(N+1-c));
24         }
25     }
26 }
```

File loading Implementaion



- Newmat

```
Matrix loadMatrix(string name) {
    ifstream myAFile(name.c_str(), std::ifstream::in);
    Matrix A(500,500);
    vector<Real> seglist;
    int i=0, index = 1;

    std::string line;
    while (std::getline(myAFile, line, ','))
    {
        if(++i >= 500) {
            A.column(index++) << &seglist[0];
            seglist.clear();
            i = 0;
        }
        else {
            seglist.push_back(atof(line.c_str()));
        }
    }
    return A;
}
```

IT++

```
mat loadMatrix(string name) {
    ifstream myAFile(name.c_str(), std::ifstream::in);
    string matrix;
    int i=0;

    std::string line;
    while (std::getline(myAFile, line))
    {
        std::replace( line.begin(), line.end(), ',', ' ');
        if(++i < 500) {
            matrix += line + ";";
        }
        else {
            matrix += line;
        }
    }

    mat A = matrix.c_str();
    return A;
}
```