

Software libraries: Eigen

Victor Eijkhout, Susan Lindsey

Fall 2024

last formatted: November 20, 2024

1. Linear algebra

- Linear algebra is very important in science
- You're not the first person to need linear algebra software
- \Rightarrow many libraries in existence
- many old libraries in Fortran, C

2.

```
1 // eigen/matvec.cpp  
2 #include <Eigen/Dense>  
3 using namespace Eigen;
```

- Eigen library: idiomatic C++
- Many common operations
- High performance

Linear algebra objects

3. Dense matrices and vectors

- Matrix size as template parameter:

```
1 // eigen/matvec.cpp
2 const int siz=5;
3 Matrix<double,siz,siz> A;
4 Vector<double,siz> sol,rhs,tmp;
```

- Dynamic definition:

```
1 // eigen/matvec.cpp
2 Matrix<float,Dynamic,Dynamic> Af(20,20);
3 // or: MatrixXf Af(20,20);
```

(use 'i' for `int` and 'd' for `double`)

4. Elementary operations

- Indexing with parentheses: $A(i,j)*v(j)$
- Methods `rows`, `cols`, `size`
- Range over vector, output:

```
1 // eigen/matvec.cpp
2 for ( auto& v : rhs )
3     v = 1.;
4 cout << rhs << '\n';
```

- $seq(f,t)$ is the sequence $f, f+1, \dots, t$:
`A(seq(0,5), seq(3,10));`
- Subblock of size $p \times q$ starting at (i,j) :
`A.block(i,j,p,q);`

Exercise 1: Matrix vector product

Code the matrix-vector product with a matrix A and vector x .
Compare the result to writing

$$y = A*x;$$

5. Jacobi method

$$Ax = f \Leftrightarrow \forall_i: \sum_j a_{ij}x_j = f_i$$

repeatedly solve:

$$x_i^{(n+1)} = (f_i - \sum_{j \neq i} a_{ij}x_j^{(n)}) / a_{ii}$$

- Code this algorithm; iterate a few steps
- Write a second version using *seq* or *block* (see above); compare to the first for correctness
- Can you write a version using ranges?

Operations

6. Solving $Ax = b$

Multiply by inverse: $x = A^{-1}b$:

```
1 // eigen/invert.cpp
2 auto sol1 = A.inverse() * rhs;
```

Use Gaussian elimination:

$$A = LU, \quad Ly = f, \quad Ux = y$$

```
1 // eigen/invert.cpp
2 PartialPivLU<MatrixXd> LU(A);
3 auto sol2 = LU.solve(rhs);
```

Exercise 2: System solving

Diagonal dominance:

$$\forall_i: a_{ii} > \sum_{j \neq i} |a_{ij}|$$

- Solve the linear system by both methods above. How accurate is the solution?
- Experiment with different matrix sizes and amounts of diagonal dominance.

Exercise 3: Vandermonde matrix

$$a_{ij} = x_i^j$$

where $\{x_i\}_i$ sequence without duplicates

- Again solve the system both ways. Observations?

7. Singular values

```
1 // eigen/invert.cpp
2 JacobiSVD<MatrixXd> svd(A);
3 VectorXd sigmas = svd.singularValues();
```

- Singular values are a little like eigenvalues
- The spread in singular values indicates how 'difficult' the matrix is
- \Rightarrow 'condition number': largest divided by smallest eigenvalue

Exercise 4: Commandline options

Use commandline options to:

- set the size of the matrix
- switch between diagonal dominant and vandermonde matrix
- for the former case set the diagonal dominance