Numerical Methods for Computational Science and Engineering

Computing with Matrices and Vectors

Machine Arithmetic

As machines can't compute in real numbers but compute with machine numbers, there are some constraints. Over- and underflow exist, the amount of numbers is finite. Thus, instead of checking for equality, we check if the difference of two numbers is smaller than some maximal relative error.

Cancellation

Cancellation occurs whenSubstrachting we attempt to:

- Substract numbers of the same size.
- Divide by a small number close to zero.

To avoid cancellation, we simply avoid such kinds of substractions and divisions by recastion expressions or use tricks like taylor approximations.

Gaussian Elimination

Theory

A is *invertible* or *regular* if and only if there exists an unique solution x for $Ax = b$, that is $x = A^{-1}b$.

Code and Complexity

In Eigen, we use the following code to solve $Ax = b$:

```
MatrixXd A;
VectorXd b, x;
...
// A has no special form
x = A.lu().solve(b);
// A is lower triangular
x = A.triangularView<Eigen::Lower>().solve(b);
// A is upper triangular
x = A.triangularView<Eigen::Upper>().solve(b);
// A is hermitian and positive definite
x = A.selfadjointView<Eigen::Upper>().llt().solve(b);
// A is hermition and positive or negative definite
x = A.selfadjointView<Eigen::Upper>().ldlt().solve(b);
// Solving based on householder transformation
x = A.HouseholderQR<MatrixXd>().solve(b);
// Solving based on singular value decomposition
```

```
x = A.jacobiSvd<MatrixXd>().solve(b);
```

The generic asymptotic complexity is $O(n^3)$, but triangular linear systems can be solved in $O(n^2)$.

Solving $Ax = b$ by computing A^{-1} and calculating $x = A^{-1}b$ is unstable. Instead we use gaussian elimination, which is stable and not affected by roundoff in practice.

Low-Rank Modifications

We assume a generic rank-one midification is applied to A:

$$\tilde{A} = A + uv^H$$

By using *block elimination*, we get:

$$\begin{bmatrix} A & u \\ v^H & -1 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \zeta \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

From this follows that:

$$\begin{gathered} A\tilde{x} + u\zeta = b \\ \tilde{x} = A^{-1}(b - u\zeta) \\ \ldots \\ (1 + v^H A^{-1} u)\zeta = v^H A^{-1} b \\ \ldots \\ A\tilde{x} = b - \frac{uv^H A^{-1}}{1 + v^H A^{-1} u} b \end{gathered}$$

Sparse Matrices

A matrix A is called *sparse*, if the number of non-zero entries $nnz(A)$ is much smaller than the size of A.

Storage Formats

Sparse matrices should be solved with `x = A.SparseLU<SparseMatrixType>().solve(b)` to further reduce the asymptotic complexity.

Also note that the product of sparse matrices isn't necessarily sparse.

Triplet/Coordinate List (COO) Format

Stores triplets with row indices, column indices and the associated values.

Compressed Row-Storage (CRS) Format

Stores the values, the corresponding column indices and row pointers that indicate the start of a new row.

```
SparseMatrix<double, RowMajor> A(rows, cols);
```

Compressed Column-Storage (CCS) Format

Same as CRS, but with column pointers and row indices instead.

```
SparseMatrix<double, ColMajor> A(rows, cols);
```

Linear Least Square Problems

Example: Linear Parameter Estimation

Assume we take m measurements and find the pairs (x_i, y_i). We can express these measurements in the following overdetermined linear system:

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Solution Concepts

For given A, b, the vector x is a least squares solution of $Ax = b$, if:

$$x \in argmin_y \|Ay - b\|_2^2$$

For a least squares solution x, the vector Ax is the unique orthogonal projection of b onto the image of $Image(A)$. From this follows that:

$$\begin{gathered} b - Ax \perp Image(A) \\ A^T(b - Ax) = 0 \\ A^TAx = A^Tb \end{gathered}$$

If $m \geq n$ and $Kernel(A) = 0$, then the linear system of equations $Ax = b$ has a unique least squares solution $x = (A^TA)^{-1}A^Tb$.

Moore-Penrose Pseudoinverse

As there are many potential least square solutions, we define the *generalized solution* x^+ of a linear system of equations as:

$$x^+ = argmin\{\|x\|_2, x \in lsq(A, b)\}$$

The generalized solution x^+ of the linear systems of equations $Ax = b$ is given by:

$$x^+ = V(V^T A^T AV)^{-1}(V^T A^T b)$$

Where the matrix $V(V^T A^T AV)^{-1}V^T$ is called the *Moore-Penrose pseudoinverse* of A.

Code

We want to solve $A^T Ax = A^T b$. To do this, we use the following procedure:

1. Compute the regular matrix $C = A^T A$.
2. Compute the right hand side vector $c = A^T b$.
3. Solve the symmetric positive definite linear system of equations $Cx = c$.

```
MatrixXd C = A.transpose() * A;
VectorXd c = A.transpose() * b;
VectorXd x = C.llt().solve(c);
```

The asymptotic complexity is $O(n^2m + n^3)$.