# Math rules!

**sin(x)**

$\sin'(x) = \cos(x)$ $\quad \arcsin'(x) = \frac{1}{\sqrt{1-x^2}}$

$\sinh'(x) = \cosh(x)$ $\quad \int \sin(x)\,dx = -\cos(x) + C$

$\sin(x) - \sin(y) = 2\cos\left(\frac{x+y}{2}\right) \cdot \sin\left(\frac{x-y}{2}\right)$

$\sin(x) + \sin(y) = 2\cos\left(\frac{x-y}{2}\right) \cdot \sin\left(\frac{x+y}{2}\right)$

**cos(x)**

$\cos'(x) = -\sin(x)$ $\quad \arccos'(x) = \frac{-1}{\sqrt{1-x^2}}$

$\cosh'(x) = \sinh(x)$ $\quad \int \cos(x)\,dx = \sin(x) + C$

$\cos(x) + \cos(y) = 2 \cdot \cos\left(\frac{x+y}{2}\right) \cdot \cos\left(\frac{x-y}{2}\right)$

$\cos(x) - \cos(y) = -2\sin\left(\frac{x+y}{2}\right) \cdot \sin\left(\frac{x-y}{2}\right)$

**tan(x)**

$\tan'(x) = \frac{1}{\cos^2(x)} = \tan^2(x) + 1$

$\arctan'(x) = \frac{1}{x^2+1}$ $\quad \tanh'(x) = \frac{1}{\cosh^2(x)}$

$\int \tan(x)\,dx = -\log(\cos(x)) + C$

$\tan(x) = \frac{\sin(x)}{\cos(x)}$ $\quad \cot\tan(x) = \frac{\cos(x)}{\sin(x)}$

**$e^{i\varphi}$**

$e^{i\varphi} = \mathrm{cis}(\varphi)$ $\quad e^{-i\varphi} = \cos\varphi - i\sin\varphi$

$\cos(\varphi) = \frac{1}{2}(e^{i\varphi} + e^{-i\varphi})$

$\sin(\varphi) = \frac{1}{2i}(e^{i\varphi} - e^{-i\varphi})$

$2\cos(\varphi) \cdot \cos(\psi) = \cos(\varphi - \psi) + \cos(\varphi + \psi)$

$2\sin(\varphi) \cdot \sin(\psi) = \cos(\varphi - \psi) - \cos(\varphi + \psi)$

$2\sin(\varphi) \cdot \cos(\psi) = \sin(\varphi + \psi) + \sin(\varphi - \psi)$

**Log/ln x**

$e^{a \ln b} = b^a$ $\quad \ln'(x) = 1/x$

$\ln(x^a) = a\ln(x)$ $\quad \int \ln(x)\,dx = x(\ln(x) - 1) + C$

$\ln(x \cdot y) = \ln x \pm \ln y$

# Eigen Tips & Tricks

**FFT Unsupported:**

```
fft.fwd(x)
fft.inv(x)
FFT<double> fft;
..[fft2r(c)]
```

**Dense Matrix:**

```
MatrixXd A(n,m);
MatrixXd A = MatrixXd::Zero(m,n)
MatrixXd A = MatrixXd::Identity(m,n)
MatrixXd A = MatrixXd::Ones(m,n)
MatrixXd A = MatrixXd::Random(m,n)
A.norm();   A.inverse();   A.transpose();
A.size();   A.block(x,y,dx,dy)
A.rowwise();   A.colwise();
A.cols();   A.col(i);   A.leftCols(i);
A.rows();   A.row(i);   A.cwise"XX"
```

```
VectorXd v(n);   v.head(i);   v.tail(i);
v.norm();   v.normalize();
v.segment(x,dx);   v.asDiagonal();
```

**Sparse Matrix:**

```
std::vector<Triplet<double>> triplets;
triplets.emplace_back(i,j,val);
SparseMatrix<double> s;
s.setFromTriplets(triplets.begin(), triplets.end());
s.makeCompressed();
s.outerIndexPtr();  ≙ col_ptr
s.innerIndexPtr();  ≙ row_index
s.valuePtr();       ≙ value_index
s.coeff(i,j)  → value at pos (i,j)
s.coeffRef(i,j) → set value in ass. =
```

**Limit:** `std::numeric_limits<double> epsilon();`

**Array Xd:**

```
Array Xd :: LinSpaced(length, start, end);
Array Xd :: Constant(length, filling val);
```

```
std::vector<type> v;   v.reserve(length);
v.push_back(i);   v.pop_back();
```

```
using namespace std;
using namespace Eigen;
```

**Eigen doc search:** Linear algebra and decomposition
Solving linear least squares systems
Solving sparse linear systems

**Derivative:**

```
double dpx = (s-1) · c[0]
for (int i = 1; i < s-1; i++) {
    dpx = x · dpx + (s-i-1) · c[i];
}
```

**Solving with LU-Decomposition**

```
X = A.fullPivLu().solve(b);
solver = A.fullPivLu();
U = solver.matrixLU().triangularView<Upper>();
L = Identity(n,n);
L.block(0,0,n,n-1).triangularView<StrictlyLower>()
    = solver.matrixLU();
```

**Solving with SVD-Decomposition**

```
JacobiSVD<MatrixXd> svd(A, ComputeThinU | ComputeThinV);
```

[or]

```
auto svd = A.jacobiSvd(ComputeThinU | ComputeThinV);
svd.singularValues();
svd.matrixU();
svd.matrixV();        X = svd.solve(b);
```

**Solving with QR-Decomposition**

```
auto qr = A.householderQr();
Q = qr.householderQ();
R = Identity(n,n);
R.triangularView<Upper>() = qr.matrixQR().triangularView<Upper>();
X = qr.solve(b);
```

**Solving with Sparse Matrix**

```
SparseLU<SparseMatrix<double>> spLU(A);
x = spLU.solve(b);
SparseQR<SparseMatrix<double>> spQR(A);
x = spLU.solve(b);
```

**Another option:**

```
SparseLU<...> spLU;   spLU.analyzePattern(A);
spLU.factorize(A);   spLU.solve(b);
```

C++ : Column-Major
Eigen : CRS is standard

**Complex conjugate:** $\overline{a \pm bi} = a \mp bi$

# Index Tablet notes

Title ; Week nr. ; page nr.


| | | |
|---|---|---|
| 2D Fourier Transformation | 7 | 7 |
| Affine Pullback | 10 | 4 |
| Affine Pullback (Quadrature) | 11 | 10 |
| Aitken-Neville scheme (with c++ code) | 8 | 6,7 |
| Algebraic Convergence | 10 | 8 |
| Algorithm: Gram-Schmidt | 4 | 6 |
| Approximation Error | 10 | 2 |
| Approx. Error of Lagrange (Piecewise) | 11 | 6 to 8 |
| Approx. Err. of Lagrange Interpolation scheme | 10 | 9 to 11 |
| Approximation of functions in 1D | 9 | 6 |
| Approximation Scheme Quadrature | 11 | 11 |
| Approximation with Low-Rank | 5 | 2 |
| Arrow Matrix LSE | 2 | 17 |
| Asymptotic Complexity | 1 | 11 |
| Barycentric Interpolation Approach | 8 | 2 |
| Bernstein Approximation / Polynomial | 9 | 8 |
| Best Low-Rank Approximation | 5 | 2 |
| Block Elimination | 2 | 16 |
| Cancellation | 2 | 4 |
| CSS Format for Sparse Matrices | 3 | 5 |
| Chebychev Implementation (c++ code) | 10,11 | 23,3 |
| Chebychev Implementation (Theory) | 10,11 | 22,1 to 3 |
| Chebychev Interpolation Error | 10 | 16 |
| Chebychev Nodes | 10 | 17 |
| Chebychev Polynomials | 10 | 13 |
| Circulant Matrix | 6 | 2 |
| Circulant Matrix (Diagonalization) | 6 | 7 |
| Complexity (Computation) | 1 | 11 |
| Complexity (Interpolation) | 8 | 1 to 2 |
| Complexity (Normal Equation) | 3 | 20 |
| Complexity (QR Decomposition) | 4 | 17 |
| Complexity Reduction | 1 | 13 |
| Composite Quadrature | 12 | 6 to 9 |
| Compressed Column Storage Format | 3 | 5 |
| Compressed Row Storage Format | 3 | 5 |
| Computational Effort | 1 | 10 |
| Computing with Matrices & Vectors | 1 | 1 |
| Condition Number | 2 | 11 |
| Constrained Least Squares | 5 | 10 |
| Convergence (Algebraic & Exponential) | 10 | 8 |
| Convolution (c++ code) | 6 | 14,15 |
| Convolution (Discrete Periodic 2D) | 7 | 10 |
| Convolution (Discrete, Definition) | 5 | 19 |
| Convolution (Discrete, Formula) | 5 | 16 |
| Convolution (Discrete, Properties) | 5 | 14 |
| Convolution (Periodic) | 6 | 1 |
| COO Format for Sparse Matrices (Triplets) | 3 | 4 |
| CRS Format for Sparse Matrices | 3 | 5 |
| Damped Newton Iteration | 13 | 21,22 |
| Data Classification | 5 | 4 |
| Decomposition : LU | 2 | 14 |
| Decomposition : QR | 4 | 6,10 |
| Decomposition : QR (c++ code) | 4 | 17 |
| Decomposition : QR (complexity) | 4 | 17 |
| Decomposition : SVD | 4 | 18 |
| Decomposition : SVD (c++ code) | 4 | 21 |
| Decomposition : SVD (Pseudo inverse) | 4 | 23 |
| Dense Matrix Storage Format | 1 | 9 |
| Dimension Reduction | 5 | 4 |
| Discrete Convolution (Definition) | 5 | 19 |
| Discrete Convolution (Formula) | 5 | 16 |
| Discrete Convolution (Properties) | 5 | 14 |
| Divided Differences scheme (with c++ code) | 8 | 8,9 |
| Elimination (Block) | 2 | 16 |
| Elimination (Gauss) | 2 | 13 |
| Error Analysis (Lagrange Piecewise) | 11 | 6 to 8 |
| Error Analysis (Approximation) | 10 | 2 |
| Error Analysis (Chebychev Interpolation) | 10 | 16 |
| * Error Analysis (Numbers) | 1 | 4 |
| Error Analysis (Quadrature w. pos. weights) | 12 | 2,3 |
| Error Analysis (Quadrature) | 11 | 12 |
| Error Analysis (Quadrature, Reduce by Factor) | 12 | 5 |
| Example: Chebychev Interpolation | 10 | 20,21 |
| Example: Convolution | 6 | 9 |
| Example: Gram-Schmidt | 4 | 12 |
| Example: QR Decomposition | 4 | 12 |
| Example: Storage Format for Sparse Matrices | 3 | 6 |
| Exponential Convergence | 10 | 8 |
| Filtering Algorithm | 5 | 13 |
| Fixed Point Iteration | 13 | 5 |
| Fixed Point Iteration in higer Dimensions | 13 | 14 to 17 |
| Fixed Point Representation | 1 | 2 |
| Floating Point Representation | 1 | 2 |
| Fourier Matrix | 6 | 6 |
| Fourier Transformation (Fast Implementation) | 7 | 1,2 |
| Fourier Transformation (Frequency Analysis) | 7 | 3 |
| Fourier Transformation (Introduction) | 6 | 5 |
| Fourier Transformation (2D) | 7 | 7 |
| Frequency Analysis (FFT) | 7 | 3 |
| Froberius-Norm | 5 | 2 |
| Function Approximation in 1D | 9 | 6 |
| Function Approximation with Bernstein | 9 | 8 |
| Function Approximation with Taylor | 9 | 7 |
| Gauss Elimination | 2 | 13 |
| Gauss-Legendre Quadrature Formula | 11 | 21 |
| Generalized Solution for least squares Problem | 3 | 19 |
| Gram-Schmidt Algorithm | 4 | 6,10 |
| Hat functions | 7 | 14 |
| Horner scheme | 7 | 17 |
| Householder QR | 4 | 12 to 16 |
| Interpolation (Introduction) | 7 | 11 |
| Interpolation → Horner scheme (with c++ code) | 7 | 17 |
| Interpolation Complexity | 8 | 1 to 3 |
| Interpolation Error (Runge's Phenomenon) | 8 | 5 |
| Interpolation with "hat functions" | 7 | 14 |
| Interpolation with Aitken-Neville scheme (c++) | 8 | 6,7 |
| Interpolation with Barycentric Approach | 8 | 2 |
| Interpolation with Divided Differences (with c++) | 8 | 8,9 |
| Interpolation with Lagrange (Piecewise) | 11 | 4,5 |
| Interpolation with Lagrange Polynomials | 7 | 18,19 |
| Interpolation with Newton Basis | 8 | 3,4 |
| Interpolation with Splines (Algorithm) | 9 | 4,5 |
| Interpolation with Splines (Theory) | 9 | 1 to 3 |
| Interpolation with Vandermonde Matrix | 8 | 1 |
| Iterative Method: Damped Newton Iteration | 13 | 21,22 |
| Iterative Method: Fixed Point Iteration | 13 | 5 |
| Iterative Method: Fixed Point (Higher Dimensions) | 13 | 14 to 17 |
| Iterative Method: Newton Iteration | 13 | 8 |
| Iterative Method: Newton Iteration (Damped) | 13 | 21,22 |
| Iterative Method: Newton Iter. (Higher Dimensions) | 13 | 17 to 19 |
| Iterative Method: Newton Iter. (w/ Derivative/Secant) | 13 | 11 |
| Iterative Method: Secant Iteration | 13 | 11 |
| Iterative Methods for Nonlinear Systems of Eq. | 13 | 1 to 4 |
| Kernel and Range of $A^TA$ | 3 | 16 |
| Kronecker Product | 2 | 3 |
| Lagrange Interpolation (Piecewise) | 11 | 4,5 |
| Lagrange Interpolation (Piecewise, Error Estim.) | 11 | 6 to 8 |
| Lagrange Interpolation scheme (Definition) | 10 | 7 |
| Lagrange Interpolation scheme (Error Estimate) | 10 | 9 to 11 |
| Lagrange Polynomials | 7 | 18,19 |
| Least Squares Problem (Constrained) | 5 | 10 |
| Least Squares Problem (Generalized solution) | 3 | 19 |
| Least Squares Problem (Introduction) | 3 | 11 |
| Least Squares Problem (orthogonal Transform.) | 4 | 3 |
| Least Squares Problem (Solution with SVD) | 4 | 22 |

| | | |
|---|---|---|
| Least Squares Problem (Solution) | 3 | 14 |
| Least Squares Problem (Total LSQ) | 5 | 8 |
| Least Squares Problem (Total LSQ, c++) | 6 | 9 |
| Lebesgue Constant | 10 | 18 |
| Legendre Polynomials (Explicit) | 11 | 20 |
| Legendre Polynomials (Recursive) | 12 | 2 |
| Low-Rank Approximation | 5 | 2 |
| Low-Rank Modification | 3 | 1 |
| LSQ Constrained | 5 | 10 |
| LSQ Generalized Solution | 3 | 19 |
| LSQ Introduction | 3 | 11 |
| LSQ Moore-Penrose Pseudoinverse | 3 | 19 |
| LSQ Orthogonal Transformation | 4 | 3 |
| LSQ Solution | 3 | 14 |
| LSQ Solution with SVD | 4 | 22 |
| LSQ Total | 5 | 8 |
| LSQ Total (c++ code) | 6 | 9 |
| LU Decomposition | 2 | 14 |
| Machine Numbers | 1 | 3 |
| Matrix Fundamentals | 1 | 6 |
| Modification (Low-Rank) | 3 | 1 |
| Moore-Penrose Pseudoinverse | 3 | 19 |
| Newton Basis | 8 | 3,4 |
| Newton Iteration | 13 | 8 |
| Newton Iteration (Damped) | 13 | 21,22 |
| Newton Iteration in higher Dimension | 13 | 17 to 19 |
| Newton Iteration w/ Derivative (Secant) | 13 | 11 |
| Nonlinear Systems of Equations | 13 | 13,14 |
| Normal Equation | 3 | 15 |
| Normal Equation Complexity | 3 | 20 |
| Norm-constrained Extrema | 5 | 1 |
| Numerical Stability | 2 | 9 |
| Order of Quadrature | 11 | 14 |
| Orthogonal Transformation | 4 | 3 |
| Partial Sum (c++ code) | 2 | 2 |
| Periodic Convolution | 6 | 1 |
| Piecewise Lagrange Interpolation | 11 | 4,5 |
| Polynomial Quadrature | 11 | 12 to 14 |
| Principle Component Analysis | 5 | 4 |
| Projection of a Vector | 4 | 6 |
| Pseudoinverse with SVD | 4 | 23 |
| Pullback (Affine) | 10 | 4 |
| Pullback (Affine, Quadrature) | 11 | 10 |
| QR Decomposition | 4 | 6,10 |
| QR Decomposition (c++ code) | 4 | 17 |
| QR Decomposition Complexity | 4 | 17 |
| Quadrature (Affine Pullback) | 11 | 10 |
| Quadrature (Composite) | 12 | 6 to 9 |
| Quadrature (Gauss-Legendre Formula) | 11 | 21 |
| Quadrature (Introduction) | 11 | 9 |
| Quadrature Error | 11 | 12 |
| Quadrature Error (Positive Weights) | 12 | 2,3 |
| Quadrature Error (Reduce by factor) | 12 | 5 |
| Quadrature Order | 11 | 14 |
| Quadrature with Approximation schemes | 11 | 11 |
| Quadrature with Polynomials | 11 | 12 to 14 |
| Range and Kernel of $A^TA$ | 3 | 16 |
| Reduced Complexity | 12 | 13,1 |
| Runge's Phenomenon | 8 | 5 |
| Secant Iteration | 13 | 11 |
| Sherman-Morrison-Woodbury Formula | 3 | 2 |
| Singular Value Decomposition | 4 | 18 |
| Solution to Least Squares Problem | 3 | 14 |
| Sparse Linear System | 3 | 3 |
| Sparse Matrix LSE Solver | 3 | 10 |
| Sparse Matrix Storage Formats | 3 | 4 |
| Splines (Algorithm) | 9 | 4,5 |
| Splines (Theory) | 9 | 1 to 3 |
| Stability (Numerical) | 2 | 9 |
| Sterlings Formula | 10 | 2 |
| Stock Market Trend | 5 | 4 |
| SVD | 4 | 18 |
| SVD (c++ code) | 4 | 21 |
| SVD Pseudoinverse | 4 | 23 |
| Taylor Approximation / Polynomial | 9 | 7 |
| Total least Squares | 5 | 8 |
| Total least Squares (c++ code) | 6 | 9 |
| Triplet Format for Sparse Matrices | 3 | 4 |
| Unconstrained optimization | 14 | 1 to 13 |
| Vandermonde Matrix | 8 | 1 |
| Error Analysis (Lagrange Interpolation) | 10 | 9 to 11 |

1

**Equality checks with eps** $= \max_{x \in \mathbb{R}} \frac{|rd(x) - x|}{|x|}$, $\tilde{f}(x) = f(x)(1+\delta)$, $|\delta| < EPS$

**Error** $Ax_{ex} = b \xrightarrow{\text{machine}} x_{app} \xrightarrow{\text{machine}} b_{app}$

- forward error $x_{ex} - x_{app}$
- backward error $b - b_{app}$ (computable)
- backwards stability $(\tilde{F}(x) = F(\tilde{x})) \Rightarrow$ mixed stability

**Condition number** $C_A = \frac{\sigma_{max}}{\sigma_{min}}$ if small: well-conditioned; if large: ill-conditioned

$C_{A^TA} = C_A^2$

$C_F(x) = \sup_{\Delta x} \left( \frac{\|F(x+\Delta x) - F(x)\|}{\|F(x)\|} \Big/ \frac{\|\Delta x\|}{\|x\|} \right)$

**Kronecker product:** $A^{mn} \otimes B^{lk} = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & A_{22}B & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{bmatrix}^{ml \times nk} = BXA^T$

**Geometric Series:** $\sum_{l=0}^{n-1} a_0 q^l = a_0 \frac{1-q^n}{1-q}$

**Cancellation:** Subtracting two nearly equal numbers (lost/rounded bits in M)

**Vieta's formula:**
- $x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2}$; $x_1 \cdot x_2 = \frac{c}{a}$, $x_1 + x_2 = -\frac{b}{a}$
- $ax^3 + bx^2 + cx + d$; $x_1 + x_2 + x_3 = -\frac{b}{a}$, $x_1 \cdot x_2 \cdot x_3 = -\frac{d}{a}$
  $x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3 = c/a$

**Norm:**

$\|A\|_2^2 = \sigma_{max}(A)^2$

$\|A\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 = \sum \sigma_i^2 = \text{trace}(A^TA)$

$\|v\|_2^2 = \sum |v_i|^2$

$\frac{\partial \|A\|_F^2}{\partial A} = 2A$

$\|f(x)\|_\infty = \max f(x)$ (global maxima)

$\frac{\partial (a^T X b)}{\partial X} = ab^T$

$\|f(x)\|_{L^n(I)}$ – n Norm auf Intervall I

**Cost:**

| | | | |
|---|---|---|---|
| dot product | $O(n)$ | Gauss elimination | $O(n^3)$ |
| tensor product | $O(nm)$ | Solving triangular matrix | $O(n^2)$ |
| matrix product | $O(mnk)$ | LU decomposition | $O(n^3)$ |
| Kronecker product | $O(mlnk)$ or $O(lkn + lnm)$ | Householder QR | $O(mn^2)$ |
| | | Fast Fourier Transform | $O(n \log n)$ |
| | | Hornerscheme | $O(n)$ |

**Matrices** Projection: $A^2 = A$ Symmetric: $A = A^T$ Hermitian: $A = A^H$

orthonormal $\overset{!}{\Rightarrow}$ orthogonal: $Q^{-1} = Q^T$ } $\|Qx\|_2 = \|x\|_2$, $|\det Q| = 1$, $|\lambda| = 1$
unitär: $Q^{-1} = Q^H$
still orthogonal/unitär

**Symmetric positive definit:** $A = A^H$, $\forall x \in \mathbb{R}^n$ $x^H A x > 0 \Leftrightarrow x \neq 0$ ($\geq \rightarrow$ semi)
all $\lambda > 0$

**Determinant:** $\det(A) = \det(U) = \det(I)$

**Regulär:**
- invertible
- $\det(A) \neq 0$
- rank (A) = full
- $Ax = 0 \rightarrow x = 0$
- all $\lambda \neq 0$
- $\mathcal{N}(A) = \{0\}$
- Basis is rows or cols of A
- $A^T$ is regulary
- dim $(\mathcal{R}(A))$ = full rank

**Kernel & range:**
$\mathcal{N}(A) = \mathcal{R}(A^H)^\perp$ dim $(\mathcal{N}(A^{nk})) = k - \dim(R(A))$
$\mathcal{N}(A)^\perp = \mathcal{R}(A^H)$
$\mathcal{N}(A^TA) = \mathcal{N}(A)$
$\mathcal{R}(A^TA) = \mathcal{R}(A^T)$

2

**Methods for LSE:**
- Gauss elimination
- LU decomposition $O(n^3 + Nn^2)$
- Block elimination $O(n)$ in some cases!
- Low rank modification
- Sparse linear system

**Schur complement:** $(A_{22} - A_{21}A_{11}^{-1}A_{12})x_2 = b_2 - A_{21}A_{11}^{-1}b_1$
stable for spd matrices and/or diagonally dominant matrices

**Low rank modification:** $\hat{A} = A + z \cdot e_i e_j^T = A + uv^T \Rightarrow \begin{bmatrix} A & u \\ v^T & -1 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = \begin{bmatrix} \tilde{b} \\ 0 \end{bmatrix}$

**Sherman-Morrison-Woodbury:** $\hat{A} = A + UV^H$ if $I + V^H A^{-1} U$ is regular
$(A + UV^H)^{-1} = A^{-1} - A^{-1}U(I_k + V^H A^{-1} U)^{-1} V^H A^{-1}$

**COO:** Triplets $(i, j, v)$; allows repetition, must be added when creating A!

**CRS:** 3 vectors: val, col-ind, row-ptr

**CCS:** 3 vectors: val, row-ind, col-ptr

last entry = nnz(A) + 1

③

**Lagrange:** $L(M,\lambda) = \|M\|_F^2 + \lambda^T(Mz - g)$, $M = \frac{gz^T}{\|z\|_2^2}$

**Least squares solution:**
- $\dim(\mathcal{R}(A)) = \operatorname{rank}(A) \le n$ (existist inner)
- is unique if $\mathcal{N}(A) = \{0\}$
- if non unique: $x^* = \operatorname{argmin}\{\|x\|_2 : x \in \operatorname{lsq}(A,b)\}$

**Normal equation:** $A^TAx = A^Tb \longrightarrow$ solve for $x$

**Reduced normal equation:** $(AV)^TAVy = (AV)^Tb$ if $y$ unique $\rightarrow x = Vy$

$V^{n,k}$ = basis of $\mathcal{N}(A)^\perp$, $\dim = k$, $\mathcal{N}(AV) = \{0\}$

**Moore-Penrose Pseudo inverse of A:** $V(V^TA^TAV)^{-1}V^TA^T$

**Extended normal equations:** (maintain sparsity, $C_{BA} \approx C_A$ with good $\alpha$)
(sparse + m,n big)

$$A^HAx = A^Hb \Leftrightarrow \begin{bmatrix} -\alpha I & A \\ A^H & 0 \end{bmatrix}\begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

**Orthogonal transformation method:** $A^{m,n}x = b$, $\operatorname{rank}(A) = n$ $m \gg n$

$\operatorname{lsq}(\tilde{A},\tilde{b}) = \operatorname{lsq}(A,b)$, $\tilde{A} = TA$, $\tilde{b} = Tb$
$\llcorner$ unitär

**QR decomposition:** $A = Q_{orth}^{m,m} R_0^{m,n}$ $Rx = Q^Tb$ $C_R = C_A$

A needs full rank $\rightarrow$
- R is unique
- Q is orthonormal & unique
- unstable in subtraction

(full: $\begin{bmatrix} R \\ 0 \end{bmatrix}$, $[Q\ 0]$)

**Total Least Squares:** $\|[A\ b] - [\hat{A}\ \hat{b}]\|_F \rightarrow \min$, $\hat{b} \in \mathcal{R}(\hat{A})$

$$[\hat{A}\ \hat{b}] = \sum_{j=1}^{n} \sigma_j (U)_{:,j} (V)_{:,j}^T$$

$$x = \hat{A}^{-1}\hat{b} = -\frac{1}{(V)_{n+1,n+1}}(V)_{1:n,n+1}$$

**Constrained Least Squares:**
- $\|Ax - b\|_2 \rightarrow \min$ & $Cx_0 = d$

  $x = x_0 + V_2 y$, $x_0 = V_1\Sigma_0^{-1}U^Td$, $\|Ax - b\|_2 \rightarrow \min$
- $\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} x \\ m \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$ (augmented/extended norm. Eq.)

**Singular Value Decomposition:** $A^{m,n} = U^{m,m}\Sigma^{m,n}V^{T\,n,n}$

($\sigma_i = \sqrt{EW}$ of $A^TA$ or $AA^T$) $U$ & $V$ unitär

reduced/thin: $m > n$: $U^{m,m} \rightarrow U^{m,n}$, $\Sigma^{m,n} \rightarrow \Sigma^{n,n}$
$\mathcal{O}(mn^2)$ $m > n$ $m < n$: $\Sigma^{m,n} \rightarrow \Sigma^{m,m}$, $V^{n,n} \rightarrow V^{n,m}$

$\operatorname{rank}(A) = r$, $r = \#\sigma_i \neq 0$
$\mathcal{N}(A) = V$ $AA^T$: U are orthonormal EV's
$\mathcal{R}(A) = U$ $A^TA$: V are orthonormal EV's

Jacobi SVD is numerically stable!

**Generalized:** $A = [U_1\ U_2]\begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V_1^H \\ V_2^H \end{bmatrix}$, $\mathcal{N}(V_1^H) = \mathcal{R}(V_1)^\perp = \mathcal{R}(V_2)$

$x = V_1\Sigma_r^{-1}U_1^Hb$ (pseudo inverse: $A^+ = V_1\Sigma_r^{-1}U_1^H$)

④

**Discrete Convolution:**
- Finite: finite input & finite output
- time-invariant: shift + filter $\hat{=}$ filter + shift output
- linear: $F(\alpha x_j + \beta y_j) = \alpha F(x_j) + \beta F(y_j)$
- causal: output only depends on past + present inputs!

**FIR filter:** finite impulse response (finite nnz)

**LT-FIR:** finite time-invariant linear causal filter

$$y_k = \sum_{j=0}^{m-1} x_j h_{k-j} \qquad k = 0, \dots, m+n-2$$
$$= \sum_{j=0}^{n-1} x_{k-j} h_j$$

input length: m
filter length: n

**Discrete n-periodic convolution:** $y_k = \sum_{j=0}^{n-1} p_{k-j}x_j = \sum_{j=0}^{n-1} p_j x_{k-j}$

$$p_j = \sum_{l \in \mathbb{Z}} h_{j+ln}$$

**Circulant matrix:** $c_{ij} = p_{j-i}$, $1 \le i,j \le n$

$$\begin{bmatrix} p_0 & & p_{n-1} \\ p_{n-1} & \ddots & \\ \vdots & & p_1 \\ p_1 & & p_0 \end{bmatrix}$$

for non periodic:
- $L = m + n - 1$
- Zero pad X to length L $\rightarrow$ L-Periodic!

**Eigenvectors:** $Cv_k = \lambda_k v_k \rightarrow V = F_n$, $\lambda_k = \sum_{j=0}^{n-1} u_j \omega_n^{-jk}$

④

**Discrete Fourier Transform (DFT):** $\overline{F}_n = [\omega_n^{ij}]_{i,j=0}^{n-1}$

$\omega_n = e^{-\frac{2\pi i}{n}}$ $\qquad \overline{F}_n^{-1} = \frac{1}{n} F_n^H = \frac{1}{n}\overline{F}_n$

Vector im Fourierraum ist symmetrisch! $\qquad \frac{1}{\sqrt{n}}\overline{F}_n$ is unitary + symmetric

**Diagonalization of circulant matrices:**

$$C\overline{F}_n = \overline{F}_n \operatorname{diag}(d_1, \dots, d_n)$$

$$d = F_n \cdot [u_0, \dots, u_{n-1}]^T, \quad u_{i-j} = c_{ij} \mathrel{\hat{=}} (C)_{i,0}$$

**Convolution with Fourier:** $x^L *_L p^L \mathrel{\hat{=}} \mathrm{IDFT}(\mathrm{DFT}(x^L) \bullet \mathrm{DFT}(p^L))$

$\mathcal{O}(n^2)$

**Frequency Filtering via DFT:** $c_k = (\overline{F}_n x)_k$ $\qquad$ high: middle, low: edges

deleting → set $\emptyset$

**2D DFT:**
- $C = \overline{F}_m Y^{m,n} \overline{F}_n = \overline{F}_m (\overline{F}_n Y^T)$
- $Y = \overline{F}_m^{-1} C \overline{F}_n^{-1} = \frac{1}{mn} \overline{F}_m C \overline{F}_n$

↳ Convolution: $U^{m,n} *_{m,n} X^{m,n} = \frac{1}{mn} \overline{F}_m \left[ (F_m U F_n)_{i,j} \cdot (F_m X F_n)_{i,j} \right] \overline{F}_n$

$j = 0:m-1$ & $i = 0:n-1$

⑤

**Piecewise linear interpolation:** $p(t) = \sum_{j=0}^{n} y_j \cdot b_j(t)$, $b_j(t) = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$

$\mathcal{O}(n^3 N)$

**Global polynomial interpolation:** $p_k(t) = \sum_{i=0}^{k} \alpha_i t^i$

**Lagrange polynomials:** $L_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - t_j}{t_i - t_j}$

$\mathcal{O}(n^2 N)$

- several data sets on same data point: $p(t) = \sum_{i=0}^{n} y_i L_i(t)$

**Vandermonde matrix:**

$$\begin{bmatrix} 1 & t_0 & t_0^2 & \cdots & t_0^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_n & t_n^2 & \cdots & t_n^n \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}$$

$\mathcal{O}(n^3)$

**Barycentric interpolation:** $p(t) = \sum_{i=0}^{n} y_i \frac{\lambda_i}{t - t_i} \prod_{j=0}^{n} (t - t_j)$

$\lambda_i = 1 / (t_i - t_0)(t_i - t_1) \cdots (t_i - t_n) \qquad n \neq i$

**Newton basis:** $N_i(t) = \prod_{j=0}^{i-1} (t - t_j)$, $N_0(t) = 0$, $p(t) = \sum_{j=0}^{n} a_i N_i(t)$

$\mathcal{O}(n^2 N)$

more stable than Lagrange

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ \vdots & N_1(t_1) & & \vdots \\ \vdots & \vdots & \ddots & \\ 1 & N_1(t_n) & \cdots & N_n(t_n) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}$$

**Partial interpolating polynomials (Aitken-Neville-scheme)** $\qquad \mathcal{O}(n^2)$

$$p_{k,l}(x) = \left( (x - t_k) p_{k+1,l}(x) - (x - t_l) p_{k,l-1}(x) \right) / (t_l - t_k)$$

$$= p_{k+1,l}(x) + \frac{x - t_l}{t_l - t_k} \left( p_{k+1,l}(x) - p_{k,l-1}(x) \right) \qquad 0 \le k \le l \le n$$

$p_{k,k}(x) = y_k$

**Cubic Spline interpolation:** $(t_i, y_i)_{i=0:n}$ $\qquad S_j(t) = a_j + b_j t + c_j t^2 + d_j t^3$

1. Vector $h_j = t_{j+1} - t_j \qquad j = 0:n-1$
2. Vector $r_j = \frac{y_{j+2} - y_{j+1}}{h_{j+1}} - \frac{y_{j+1} - y_j}{h_j} \qquad j = 0:n-2$
3. Tridiagonale Matrix
   ↳ solve

$$\begin{bmatrix} \frac{h_0 + h_1}{3} & \frac{h_1}{6} & \cdots & 0 \\ \frac{h_1}{6} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{h_{n-2}}{6} \\ 0 & \cdots & \frac{h_{n-2}}{6} & \frac{h_{n-2} + h_{n-1}}{3} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_{n-1} \end{bmatrix} = \begin{bmatrix} r_0 \\ \vdots \\ r_{n-2} \end{bmatrix}$$

4. $\sigma_0 = \sigma_n = 0$
5. $a_j = y_j$ ; $\quad b_j = \frac{y_{j+1} - y_j}{h_j} - \frac{h_j (2\sigma_j + \sigma_{j+1})}{6}$

$c_j = \frac{\sigma_j}{2}$ ; $\quad d_j = \frac{\sigma_{j+1} - \sigma_j}{6 h_j}$

6

Taylor polynom: $f(t) = \sum_{i=0}^{k} \frac{f^{(i)}(t_0)}{j!} (t-t_0)^j$

– Error: $f(t) - T_k(t) = \frac{f^{(k+1)}(\xi)}{(k+1)!} (t-t_0)^{k+1}$

Bernstein approximation: $p_n(t) = \sum_{j=0}^{n} f(j/n)\, B_j^n(t)$

Bernstein polynomials: $B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j}$, $\sum_{j=0}^{n} B_j^n(t) \equiv 1$

Chebychev polynomials: $T_n(t) = \cos(n \cdot \arccos(t))$ $\boxed{-1 \le t \le 1}$

$\Theta(n \log n)$

$T_{n+1}(t) = 2t\, T_n(t) - T_{n-1}(t)$, $T_0 = 1$, $T_1(t) = t$

Nodes: $t_k = a + \frac{1}{2}(b-a)\left(\cos\left(\frac{2k+1}{2(n+1)}\pi\right)+1\right)$, $k = 0:n$

Interpolation scheme

$I_J f$ – function to be approx.

set of nodes

$\|f - I_J f\|_{L^\infty([a,b])} \to \min$

$\|f\|_2 = \sqrt{\int |f(x)|^2 dx}$

$p(x) = \sum_{j=0}^{n} \alpha_j T_j(x)$ ; $p(t_k) = f(t_k) = y_k$

7, 8

Quadrature: $\int_a^b f(t)\,dt \approx Q_n(f) = \sum_{j=1}^{n} \omega_j^n f(c_j^n)$

$= \int_a^b p_{n-1}(t)\,dt = \sum_{i=0}^{n-1} f(t_i) \int_a^b L_i(t)\,dt$

$\omega_i = \int_a^b L_{i-1}(t)\,dt$, $c_i = t_{i-1}$

Legendre polynomial: $\int_{-1}^{1} P_n(t)\, q(t)\,dt = 0$, $P_n(1) = 1$

$P_{n+1}(t) = \frac{2n+1}{n+1} t\, P_n(t) - \frac{n}{n+1} P_{n-1}(t)$, $P_0 = 1$, $P_1(t) = t$

Newton iteration: $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$

Fixpoint Iteration: $\Phi(x) = x - \frac{f(x)}{f'(x)}$

$\Phi'(x) = \frac{f''(x) f(x)}{(f'(x))^2}$