1 VIM Commands

Format Code: `GG=gg`

Show Line Numbers: `set nu`

Find and Replace: `:%s/{t1}/{t2}/g`

2 Programming Tricks

2.1 Lambda Functions

```
double a = ...; MatrixXd Y = ...;
auto g = [a,&X] (VectorXd y) {
  return a*X*y;
};
```

2.2 Plots with MathGL and Figure Wrapper

```
#include <figure/figure.hpp>

int main() {
    // Create vectors to keep track of (N,err)
    vector<double> points;
    vector<double> errors;
    // Compute Integral for various numbers of
      gauss points
    for(unsigned N = 1; N < max_N; ++N) {
        // Compute approximated integral
        double I_approx = doquadrule(N);
        // Compute error
        double err = std::abs(I_ex - I_approx);
        // Kepp track of results
        points.push_back(N);
        errors.push_back(err);
    }

    // Create plot with results
    mgl::Figure fig;
    fig.title("Quadrature error");
    // linear in log-log: algebraic:   C*n^h
    // linear in lin-log: exponential: C*q^n
    //          (x    , y)
    fig.setlog(true, true);
    fig.plot(points, errors, "
      +r").label("Error");
    // add a reference line (makes mostly sence
      for algebraic)
    fig.fplot("x^(-4)", "k--").label("O(n^{-4})");
    fig.xlabel("No. of quadrature nodes");
    fig.ylabel("|Error|");
    fig.legend();
    fig.save("QuadrErr"); // saves as QuadrErr.eps

    return 0;
}
```

3 Basic Math

3.1 Solutions to Quadratic Equation

$ax^2 + bx + c = 0 \Longrightarrow x_{1/2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

3.2 Complex Numbers

$e^{i\varphi} = \cos(\varphi) + i\sin(\varphi)$

$z = x + iy \Longleftrightarrow x = \operatorname{Re} z,\ y = \operatorname{Im} z$

$$\begin{aligned} z &= x + iy \\ &= re^{i\varphi} \end{aligned} \Longleftrightarrow \begin{cases} x = r\cos\varphi \\ y = r\sin\varphi \end{cases} \Longleftrightarrow \begin{cases} r &= |z| \\ \varphi &= \arccos(x/r) \\ &= \arcsin(y/r). \end{cases}$$

$\overline{z} = x - iy \qquad |z| = \sqrt{z\overline{z}} = r$

$\frac{a+bi}{c+di} = \frac{v}{w} = \frac{v}{w}\frac{\overline{w}}{\overline{w}} = \frac{v\overline{w}}{|w|^2} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$

3.3 Common Integrals

$f(x)$	$F(x)$
$x^\alpha,\ (\alpha \neq 0)$	$\frac{x^{\alpha+1}}{\alpha+1} + C$
$\frac{1}{x}$	$\ln(\lvert x\rvert) + C$
e^x	$e^x + C$
α^x	$\frac{\alpha^x}{\ln(\alpha)} + C$
$\sin(x)$	$-\cos(x) + C$
$\cos(x)$	$\sin(x) + C$
$\sinh(x)$	$\cosh(x) + C$
$\cosh(x)$	$\sinh(x) + C$
$\frac{1}{\sqrt{1-x^2}}$	$\arcsin(x) + C$
$\frac{-1}{\sqrt{1-x^2}}$	$\arccos(x) + C$
$\frac{1}{1+x^2}$	$\arctan(x) + C$
$\frac{1}{\sqrt{1+x^2}}$	$\operatorname{arcsinh}(x) + C$
$\frac{1}{\sqrt{x^2-1}}$	$\operatorname{arccosh}(x) + C$
$\frac{1}{1-x^2}$	$\operatorname{arctanh}(x) + C$
$\tan(x)$	$-\log(\lvert\cos(x)\rvert) + C$
$\log(x)$	$x(\log(x) - 1) + C$

3.4 Trig. Functions as Euler Functions

$\sin(t) = \frac{e^{it}-e^{-it}}{2i} \qquad \cos(t) = \frac{e^{it}-e^{-it}}{2i}$

$\sinh(z) = \frac{e^z-e^{-z}}{2} \qquad \cosh(z) = \frac{e^z+e^{-z}}{2}$

$\tan(z) = \frac{\sin(z)}{\cos(z)} = -i\frac{e^{iz}-e^{-iz}}{e^{iz}+e^{-iz}} \qquad \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z-e^{-z}}{e^z+e^{-z}}$

3.5 Trigonometric Identities

$\sin^2(x) + \cos^2(x) = 1 \qquad \sinh^2(x) - \cosh^2(x) = 1$

$\sin^2(x) = \frac{1}{2} - \frac{1}{2}\cos(2x) = 1 - \cos^2(x) \qquad \cot(x) = \frac{1}{\tan(x)}$

$\cos^2(x) = \frac{1}{2} + \frac{1}{2}\cos(2x) = 1 - \sin^2(x)$

$\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\alpha)$

$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$

$\sin(2\alpha) = 2\sin(\alpha)\cos(\alpha) \qquad \cos(2\alpha) = \cos^2(\alpha) - \sin^2(\alpha)$

3.6 Series

Geometric Series

$S_n = a_0 \sum_{k=0}^n q^k = a_0\frac{1-q^{n+1}}{1-q} = a_0\frac{q^{n+1}-1}{q-1}$

$S = a_0 \sum_{k=0}^\infty = \frac{1}{1-q} \quad \text{if } |q| < 1$

Arithmetic Series

$S_n = \sum_{k=0}^n (k \cdot d + a_0) = (a_0 + a_n) \cdot \frac{(n+1)}{2}$

where $a_i = i \cdot d + a_0$, or $a_i = i\underbrace{(a_{n+1} - a_n)}_{=d} + a_0$.

3.7 Taylor Expansions

$e^z = \sum_{k=0}^\infty \frac{z^k}{k!} = 1 + z + \frac{z^2}{2} + \frac{z^3}{3!} + \frac{z^4}{4!} + \cdots$

$\sin(\varphi) = \sum_{k=0}^\infty (-1)^k \frac{\varphi^{2k}}{(2k)!} = \varphi - \frac{\varphi^3}{3!} + \frac{\varphi^5}{5!} + \cdots$

$\sinh(z) = \sum_{k=0}^\infty \frac{z^{2k+1}}{(2k+1)!} = z + \frac{z^3}{3!} + \frac{z^5}{5!} + \cdots$

$\cos(\varphi) = \sum_{k=0}^\infty (-1)^k \frac{\varphi^{2k+1}}{(2k+1)!} = 1 - \frac{\varphi^2}{2!} + \frac{\varphi^4}{4!} + \cdots$

$\cosh(z) = \sum_{k=0}^\infty \frac{z^{2k}}{(2k)!} = 1 + \frac{z^2}{2!} + \frac{z^4}{4!} + \cdots$

$\tan(\varphi) = \ldots\text{complicated}\ldots = 1 + \frac{\varphi^3}{3} + \frac{2\varphi^5}{15} + \cdots$

$\tanh(z) = \ldots\text{complicated}\ldots = 1 - \frac{z^3}{3} + \frac{2z^5}{15} - \cdots$

$\ln(1+z) = \sum_{k=1}^\infty \frac{(-1)^{k+1}}{k} z^k = z - \frac{z^2}{2} + \frac{z^3}{3} + \cdots$

$(1+z)^\alpha = \sum_{k=0}^\infty \binom{\alpha}{k} z^k = 1 + \alpha z + \frac{\alpha(\alpha-1)}{2!} z^2 + \cdots$

3.8 Even and Odd Functions

Even $\forall x\colon f(x) = f(-x)$
Odd $\forall x\colon f(-x) = -f(x)$

3.9 Basis Transformation Matrix

Let $\mathbf{A}$ and $\mathbf{B}$ be matrices with basis vectors as columns for some n-dimensional space.

$$\mathbf{A} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ | & | & & | \end{bmatrix}$$

The transformation $T\colon \mathbb{R}^n \to \mathbb{R}^n$ with transformation matrix $\mathbf{T} \in \mathbb{R}^{n\times n}$ of a vector in basis representation w.r.t basis $\mathbf{A}$, into the basis representation w.r.t. $\mathbf{B}$ is:

$$\mathbf{T} = \begin{bmatrix} | & | & & | \\ T(\mathbf{a}_1) & T(\mathbf{a}_2) & \cdots & T(\mathbf{a}_n) \\ | & | & & | \end{bmatrix}$$

4 Matrices and Vectors

D. (Tensor Product) of two *vectors* $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ is the matrix $\mathbf{W}$ which is defined as

$$\mathbf{W} = \mathbf{v}\mathbf{u}^\mathsf{T} = \begin{pmatrix} | & | & & | \\ u_1\mathbf{v} & u_2\mathbf{v} & \cdots & u_m\mathbf{v} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n\times m}.$$

Hence $(\mathbf{W})_{ij} = (\mathbf{v}\mathbf{u}^\mathsf{T})_{ij} = v_i u_j$.

D. (Tensor Product) of two *matrices* $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{p\times n}$ is the matrix $\mathbf{W}$ which is defined as

$$\mathbf{W} = \mathbf{A}\mathbf{B}^\mathsf{T} = \sum_{\ell=1}^{n} \mathbf{a}_\ell \mathbf{b}_\ell^\mathsf{T} \in \mathbb{R}^{m\times p}.$$

Hence, the tensor product of two matrices is just the sum of the tensor products of the column vectors.

D. (Kronecker Product) of two *matrices* $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{B} \in \mathbb{R}^{p\times q}$ ist the matrix $\mathbf{K} \in \mathbb{R}^{np\times mq}$, where

$$\mathbf{K} = \mathbf{A}\otimes\mathbf{B} = \begin{pmatrix} \boxed{a_{11}\cdot\mathbf{B}} & \boxed{a_{12}\cdot\mathbf{B}} & \cdots & \boxed{a_{1n}\mathbf{B}} \\ \boxed{a_{21}\cdot\mathbf{B}} & \boxed{a_{22}\cdot\mathbf{B}} & \cdots & \boxed{a_{2n}\mathbf{B}} \\ \vdots & \vdots & \ddots & \vdots \\ \boxed{a_{m1}\cdot\mathbf{B}} & \boxed{a_{m2}\cdot\mathbf{B}} & \cdots & \boxed{a_{mn}\mathbf{B}} \end{pmatrix}.$$

4.1 Complexity of Algebraic Operations

$\alpha \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{B} \in \mathbb{R}^{n\times k}$

- Scaling $\alpha\mathbf{x} \in \mathcal{O}(n)$
- Dot Product $\mathbf{x}^\mathsf{H}\mathbf{x} \in \mathcal{O}(n)$
- Tensor Product $\mathbf{x}\mathbf{y}^\mathsf{H} \in \mathcal{O}(nm)$
- Matrix-Vector Mult $\mathbf{A}\mathbf{x} \in \mathcal{O}(mn)$
- Matrix-Matrix Product $\mathbf{A}\mathbf{B} \in \mathcal{O}(mnk)$
- Solving $[\ \mathbf{A} \mid \mathbf{u}\ ] \in \mathcal{O}(mnn)$
- Kroneker Product times Vec $(\mathbf{A}\otimes\mathbf{B})\mathbf{x} \in \mathcal{O}(n^3)$

4.2 Tricks to Reduce Complexity

- Exploit Associativity of Operations
- Exploit Hidden summations (Tensor Product, SVD)
- Find hidden Cumulative sums
- Use fast Kronecker Products

5 Numerical Stability

D. (Cancellation) When two numbers of about the same size are subtracted then we may have a large relative error (depending how the relative error was before).

5.1 Tricks to Avoid Cancellation

- Identities: Trigonometric,...
- Case-Distinctions
- Taylor Approximations
- Theorems: Vieta
- Computing Diff. Quot through Approx.
- Don't subtract (almost) equal and collinear vectors
- Avoid alternating signs in series

6 Linear Systems of Equations

- Gauss solve $\mathcal{O}(n^3)$
- LU: decomp $\mathcal{O}(n^3)$, solve $\mathcal{O}(n^2)$
- Inverse: compute $\mathcal{O}(n^3)$, solve $\mathcal{O}(n^2)$

7 Singular Value Decomposition

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{H} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\mathsf{H}, \qquad \mathbf{A} \in \mathbb{K}^{m\times n},$$

$p := \min\{m,n\}$, $r := \operatorname{rank}(\mathbf{A})$, $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_p)$
$\sigma_1 > \sigma_2 > \ldots > \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$

Full:

- $\mathbf{U} \in \mathbb{K}^{m\times m}$ $[\mathcal{R}(\mathbf{A}) | \mathcal{N}(\mathbf{A})]$ (unitary)
- $\mathbf{\Sigma} \in \mathbb{K}^{m\times n}$ (generalized diagonal)
- $\mathbf{V} \in \mathbb{K}^{n\times n}$ $[\mathcal{R}(\mathbf{A}^\mathsf{H}) | \mathcal{N}(\mathbf{A}^\mathsf{H})]$ (unitary)

Economical:

- $\mathbf{U} \in \mathbb{K}^{m\times p}$ $[\mathcal{R}(\mathbf{A})]$ (orthogonal columns)
- $\mathbf{\Sigma} \in \mathbb{K}^{p\times p}$ (diagonal)
- $\mathbf{V} \in \mathbb{K}^{n\times p}$ $[\mathcal{R}(\mathbf{A}^\mathsf{H})]$ (orthogonal columns)

Numerical Rank $r := \max_{j\in\{1,\ldots,p\}} \left(\frac{\sigma_j}{\sigma_1} \geq TOL\right)$

Cost of Eco SVD $\mathcal{O}(\min\{m,n\}^2 \max\{m,n\})$
$\to$ Linear in big dimension if other is small.

8 Least Squares

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{x} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m$$

There is no solution if $\mathbf{b} \notin \mathcal{R}(\mathbf{A})$.

D. (A Least Squares Solution)

$$\mathbf{x}^* \in \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 = \operatorname*{arg\,min}_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ij}x_j - b_j \right)^2$$
$$= \operatorname{lsq}(\mathbf{A}, \mathbf{b}) = \left\{ \mathbf{x} \;\middle|\; \mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} = \mathbf{A}^\mathsf{T}\mathbf{b} \right\}.$$

- **unique** iff $\operatorname{rank}(\mathbf{A}) = n$, $\ker(\mathbf{A}) = \{\mathbf{o}\}$
- **not unique** iff $\operatorname{rank}(\mathbf{A}) < n$, then $\ker(\mathbf{A}) \supset \{\mathbf{o}\}$.

Geometric Interpretation Projection of $\mathbf{b}$ onto $\mathcal{R}(\mathbf{A})$. So $\mathbf{b} - \mathbf{A}\mathbf{x}$ will be orthogonal to any $\mathbf{z} = \mathbf{A}\mathbf{y} \in \mathcal{R}(\mathbf{A})$, so writing

$$\langle \mathbf{Ay}, \mathbf{b}-\mathbf{Ax}\rangle = 0 \iff \mathbf{A}^\mathsf{T}\mathbf{Ax} = \mathbf{A}^\mathsf{T}\mathbf{b}$$

leads to the normal equations that are satisfied iff $\mathbf{x}$ is a lsq solution.

Advantage $n \times n$ system is possibly smaller than the orig.

D. (Generalized Solution)

$$\mathbf{x}^\dagger = \min\{\|\mathbf{x}\|_2\}\ \mathbf{x} \in \mathrm{lsq}(\mathbf{A},\mathbf{b})$$

8.1 Four Fundamental Subspaces Theorem

For $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{A}\colon \mathbb{R}^n \to \mathbb{R}^m$ we have

$$\mathcal{N}(\mathbf{A}) \perp \mathcal{R}(\mathbf{A}^\mathsf{T}) \qquad \mathcal{N}(\mathbf{A}^\mathsf{T}) \perp \mathcal{R}(\mathbf{A})$$

$$\mathcal{N}(\mathbf{A}) \oplus \mathcal{R}(\mathbf{A}^\mathsf{T}) = \mathbb{R}^n \qquad \mathcal{N}(\mathbf{A}^\mathsf{T}) \oplus \mathcal{R}(\mathbf{A}) = \mathbb{R}^m$$

8.2 Solution Spaces of Lsq Solutions

T. For $\mathbf{A} \in \mathbb{R}^{m\times n}$, $(m \geq n)$ it holds that

- $\mathcal{N}(\mathbf{A}^\mathsf{T}\mathbf{A}) = \mathcal{N}(\mathbf{A}) \subset \mathbb{R}^n$
- $\mathcal{R}(\mathbf{A}^\mathsf{T}\mathbf{A}) = \mathcal{R}(\mathbf{A}) \subset \mathbb{R}^n$

8.3 Normal Equation Methods

8.3.1 Through Normal Equation

1. Compute $\mathbf{C} := \mathbf{A}^\mathsf{T}\mathbf{A}$, $\mathcal{O}(n^2m)$
2. Compute rhs vec $\mathbf{c} := \mathbf{A}^\mathsf{T}\mathbf{b}$, $\mathcal{O}(nm)$
3. Solve LSE $\mathbf{Cx} = \mathbf{c}$, $\mathcal{O}(n^3)$

Total complexity: $\mathcal{O}(n^3 + n^2m)$.

If $\mathbf{A}$ has full rank, then the LSE is s.p.d. $\to$ no worries about stability, 3-loop elimination is good, no pivoting.

- $\mathbf{A}^\mathsf{T}\mathbf{A}$ is symmetric, and
- $\forall \mathbf{x} \neq \mathbf{o}\colon \mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{Ax} = \|\mathbf{Ax}\|_2^2 > 0$, since $(\ker(\mathbf{A}) = \{\mathbf{o}\}$.

8.3.2 Orthogonal Transformation Methods

Idea: Transform $\mathbf{Ax} = \mathbf{b}$ into $\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}}$ such that $\mathrm{lsq}(\mathbf{A},\mathbf{b}) = \mathrm{lsq}(\widetilde{\mathbf{A}},\widetilde{\mathbf{b}})$. Now the nice thing is that for orthogonal transformations $\mathbf{T}$ it hods that

$\arg\min_\mathbf{x} \|\mathbf{Ax}-\mathbf{b}\|_2 \arg\min_\mathbf{x} \|\mathbf{TAx}-\mathbf{Tb}\|_2$

and orthogonal transformations are numerically stable.

Orth. Transf.: Rotations, Permutations, Reflections, ...

Approach: Transform $\mathbf{A}$ into upper triangular $\mathbf{R}$.

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\|\mathbf{Ax}-\mathbf{b}\| = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\left\|\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\begin{bmatrix}x_1\\ \vdots\\ x_n\end{bmatrix} - \begin{bmatrix}\widetilde{b}_1\\ \vdots\\ \widetilde{b}_n\\ \vdots\\ \widetilde{b}_m\end{bmatrix}\right\| = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\left\|[\mathbf{R}]\begin{bmatrix}x_1\\ \vdots\\ x_n\end{bmatrix} - \begin{bmatrix}\widetilde{b}_1\\ \vdots\\ \widetilde{b}_n\end{bmatrix}\right\|$$

Solving Least Squares via QR-Transformation

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\|\mathbf{Ax}-\mathbf{b}\|_2 = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\|\mathbf{QRx}-\mathbf{b}\|_2$$

$$= \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\left\|\mathbf{Q}^\mathsf{T}\mathbf{QRx}-\mathbf{Q}^\mathsf{T}\mathbf{b}\right\|_2 = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\left\|\mathbf{Rx}-\mathbf{Q}^\mathsf{T}\mathbf{b}\right\|_2$$

Then we remove the last n rows of $\mathbf{R}$ and $\mathbf{Q}^\mathsf{T}\mathbf{b}$ and if $\mathrm{rank}(\mathbf{A}) = n$ we can invert $\mathbf{R}$ and get the solution $\mathbf{x} = \widetilde{\mathbf{R}}^{-1}\widetilde{\mathbf{b}}$.

QR-Decomposition $\mathbf{q}_1 = \frac{1}{\|\mathbf{a}_1\|}\mathbf{a}_1$

$$\underbrace{\begin{pmatrix}\mathbf{a}_1 & \cdots & \mathbf{a}_n\end{pmatrix}}_{\mathbf{A}} = \underbrace{\begin{pmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_n\end{pmatrix}}_{\mathbf{Q}}\underbrace{\begin{pmatrix}\mathbf{q}_1^\mathsf{T}\mathbf{a}_1 & \mathbf{q}_1^\mathsf{T}\mathbf{a}_2 & \cdots & \mathbf{q}_1^\mathsf{T}\mathbf{a}_n\\ 0 & \mathbf{q}_2^\mathsf{T}\mathbf{a}_2 & \cdots & \mathbf{q}_2^\mathsf{T}\mathbf{a}_n\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \cdots & 0 & \mathbf{q}_n^\mathsf{T}\mathbf{a}_n\end{pmatrix}}_{\mathbf{R}}$$

Complexity $\mathcal{O}()$

Sidenote: Adding a column to $\mathbf{A}$ in QR decomp:
$\mathbf{q}_{n+1} = \frac{1}{\|\mathbf{a}_{n+1}-\mathbf{QQ}^\mathsf{T}\mathbf{a}_{n+1}\|_2}(\mathbf{a}_{n+1} - \mathbf{QQ}^\mathsf{T}\mathbf{a}_{n+1})$
Complexity $\mathcal{O}(mn)$

Adding a row to $\mathbf{A}$ in QR decomp: see script.

Other orthogonal transformation methods are: Attacks with Givens rotations, or Householder reflections.

8.4 Total Least Squares

Given $\mathbf{A} \in \mathbb{K}^{m\times n}$, $m > n$, $\mathrm{rank}(\mathbf{A}) = n$, $\mathbf{b} \in \mathbb{K}^n$. Both with measurement errors.

Goal Find *nearest* solvable linear system $\left[\ \widetilde{\mathbf{A}}\ \middle|\ \widetilde{\mathbf{b}}\ \right]$:

$$\underset{[\ \widetilde{\mathbf{A}}\ |\ \widetilde{\mathbf{b}}\ ]}{\arg\min}\left\|[\ \mathbf{A}\ |\ \mathbf{b}\ ] - \left[\ \widetilde{\mathbf{A}}\ \middle|\ \widetilde{\mathbf{b}}\ \right]\right\|_F \quad \text{s.t. } \widetilde{\mathbf{b}} \in \mathcal{R}(\widetilde{\mathbf{A}})$$

Solution is the best rank-n-approximation of $[\ \mathbf{A}\ |\ \mathbf{b}\ ]$.
Let $[\ \mathbf{A}\ |\ \mathbf{b}\ ] = \mathbf{U\Sigma V}^\mathsf{H}$, then

$$\left[\ \widetilde{\mathbf{A}}\ \middle|\ \widetilde{\mathbf{b}}\ \right] = (\mathbf{U})_{:,1:n}(\mathbf{\Sigma})_{1:n,1:n}(\mathbf{V}_{:,1:n})^\mathsf{H}$$

and the solution of the equation is

$$\left[\ \widetilde{\mathbf{A}}\ \middle|\ \widetilde{\mathbf{b}}\ \right](\mathbf{V})_{:,n+1} = \mathbf{o}$$

$$\widetilde{\mathbf{A}}(\mathbf{V})_{1:n,n+1} + \widetilde{\mathbf{b}}(\mathbf{V})_{n+1,n+1} = \mathbf{o}$$

$$\widetilde{\mathbf{A}}\underbrace{\frac{1}{(\mathbf{V})_{n+1,n+1}}(\mathbf{V})_{1:n,n+1}}_{=\widetilde{\mathbf{x}}} = \widetilde{\mathbf{b}}$$

8.5 Constrained Least Squares

Given: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $m \geq n$, $\mathrm{rank}(\mathbf{A}) = n$, $\mathbf{b} \in \mathbb{R}^m$
$\mathbf{C} \in \mathbb{R}^{p\times n}$, $p < n$, $\mathrm{rank}(\mathbf{C}) = p$, $\mathbf{d} \in \mathbb{R}^p$

Goal: $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n}\|\mathbf{Ax}-\mathbf{b}\|_2 \quad \text{s.t. } \mathbf{Cx} = \mathbf{d}$

8.5.1 Solution via Lagrangian Multipliers

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\,\overbrace{\max_{\boldsymbol{\lambda}}\underbrace{\frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2 - \boldsymbol{\lambda}^\mathsf{T}(\mathbf{Cx}-\mathbf{d})}_{=:L(\mathbf{x},\boldsymbol{\lambda})}}^{=\infty \text{ if constraint } \mathbf{Cx}=\mathbf{d} \text{ is volated}}$$

Now the clue is that L must be flat at the solution point, computing the partial derivatives gives us the *augmented normal equations*.

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{x}} &= \mathbf{A}^\mathsf{T}(\mathbf{Ax}-\mathbf{b}) + \mathbf{C}^\mathsf{T}\boldsymbol{\lambda} = \mathbf{o}\\ \frac{\partial L}{\partial \boldsymbol{\lambda}} &= \mathbf{Cx}-\mathbf{d} = \mathbf{o}\end{aligned} \iff \begin{bmatrix}\mathbf{A}^\mathsf{T}\mathbf{A} & \mathbf{C}^\mathsf{T}\\ \mathbf{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\mathbf{b}\\ \mathbf{d}\end{bmatrix}$$

Solving them gives us $\mathbf{x}$ as part of the sol.

8.5.2 Solution via SVD

9 Filtering Algorithms

9.1 Signal Sequences

D. (Bi-Infinite Sequence) $(x_j)_{j\in\mathbb{Z}} \in \ell^\infty(\mathbb{Z})$.

Com. ℓ^∞ means that it's bounded.

Com. If x_j is samped at aequidistant points in time (time interval Δt), then $x_j \sim X(j \cdot \Delta t)$.

Com. if the signal is finite

$(x_j)_{j\in\mathbb{Z}} = (\dots, 0, x_0, x1, \dots, x_n, 0, \dots)$

then we can identify it with a vector $\mathbf{x} \in \mathbb{R}^n$.

9.2 LT-FIR Channels

D. (Filter/Channel) $F\colon \ell^\infty(\mathbb{Z}) \to \ell^\infty(\mathbb{Z})$

D. (Impulse) at t_0 is the sequence $(\delta_{0,j})_{j\in\mathbb{N}}$

D. (Impulse Response) $(h_j)_{j\in\mathbb{Z}} = F((\delta_{ij}))_{j\in\mathbb{Z}}$

D. (Finite Channel) for every finite input it produces a finite output.

D. (Causal Channel) if the output does not start before the input.

D. (Shift Operator) $S_m((x_j)_{j\in\mathbb{Z}}) = (x_{j+m})_{j\in\mathbb{Z}}$.

D. (Time-Invariant) for all inputs, shifting the input leads to the same output shifted by the same amout; it *commutes* with the shift operator:

$$\forall (x_j)_{j\in\mathbb{Z}}\, \forall m \quad F(S_m((x_j)_{j\in\mathbb{Z}})) = S_m(F((x_j)_{j\in\mathbb{Z}}))$$

D. (Linear Channel)

$F(\alpha(x_j)_{j\in\mathbb{Z}} + \beta(y_j)_{j\in\mathbb{Z}}) = \alpha F((x_j)_{j\in\mathbb{Z}}) + \beta F((y_j)_{j\in\mathbb{Z}})$

D. (LT-FIR Channel) is a channel that is *linear*, *time-invariant*, *causal*, and *finite*.

9.3 Discrete Convolutions

LT-FIR Formula

The output $(y_j)_{j\in\mathbb{Z}}$ for the input $(x_j)_{j\in\mathbb{Z}}$ of a LT-FIR channel F with impulse response $(h_j)_{j\in\mathbb{Z}}$ can be written as a weighted sum of time-shifted impulse responses:

$$F\left((x_j)_{j\in\mathbb{Z}}\right) = F\left(\sum_{k\in\mathbb{Z}} x_k(\delta_{k,j})_{j\in\mathbb{Z}}\right) \overset{\text{lin.}}{=} \sum_{k\in\mathbb{Z}} F\left(x_k(\delta_{k,j})_{j\in\mathbb{Z}}\right)$$
$$\overset{\text{lin.}}{=} \sum_{k\in\mathbb{Z}} x_k F\left((\delta_{k,j})_{j\in\mathbb{Z}}\right) \overset{\text{tim. inv.}}{=} \sum_{k\in\mathbb{Z}} x_k (h_{j-k})_{j\in\mathbb{Z}}.$$

If the signal is finite then the output will be too, and we can write it as the following matrix equation:

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m+n-2} \end{pmatrix}}_{\text{Output Signal Vector } \mathbf{y}} = \underbrace{\begin{pmatrix} h_0 & 0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ h_{m-2} & \cdots & h_2 & h_1 & h_0 & 0 \\ h_{m-1} & h_{m-2} & \cdots & h_2 & h_1 & h_0 \\ 0 & h_{m-1} & h_{m-2} & \cdots & h_2 & h_1 \\ 0 & 0 & h_{m-1} & h_{m-2} & \cdots & h_2 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & h_{m-1} & h_{m-2} \\ 0 & \cdots & \cdots & 0 & 0 & h_{m-1} \end{pmatrix}}_{\text{Filter Mapping Matrix } \mathbf{F}} \underbrace{\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix}}_{\text{Input Signal Vector } \mathbf{x}}$$

This is called a *discrete convolution*.

D. (Discrete Convolution) Given

$\mathbf{x} = (x_0, \dots, x_{n-1})^\mathsf{T} \in \mathbb{K}^n$, $\mathbf{h} = (h_0, \dots, h_{m-1})^\mathsf{T} \in \mathbb{K}^m$

their *discrete convolution* is the vector $\mathbf{y} \in \mathbb{K}^{m+n-1}$ (0-indexing) with components

$$y_k = \sum_{j=0}^{\min\{n-1,m-1\}} h_{k-j}x_j, \quad k = 0, \dots, m+n-2 \quad (h_j := 0 \text{ for } j < 0).$$

Another shorter notation for the convolution is:

$$\mathbf{y} = \mathbf{h} \star \mathbf{x} = \mathbf{x} \star \mathbf{h}.$$

Com. $\star$ is commutative, since

$\mathbf{y} = \sum_{k\in\mathbb{Z}} x_k h_{j-k} = \mathbf{x} \star \mathbf{h} \overset{\ell := j-k}{=} \sum_{\ell\in\mathbb{Z}} x_{j-\ell}h_\ell = \mathbf{h} \star \mathbf{x}$.

D. (n-Periodic Signal) $\forall j \in \mathbb{Z}\colon x_j = x_{j+n}$.

Com. So we need n numbers to describe it: $x_0, \dots, x_{n-1}$.

D. (n-Periodic Impulse) $\sum_{k\in\mathbb{Z}} (\delta_{nk,j})_{j\in\mathbb{Z}}$

Since an n-periodic signal has been going on since forever, we know that the output of an LT-FIR filter F also to be n-periodic. So F can be described by a linear mapping $\mathbb{R}^n \to \mathbb{R}^n$.

$$\mathbf{y} = \mathbf{F}\mathbf{x} = \overbrace{\begin{bmatrix} p_0 & p_{n-1} & p_{n-2} & \cdots & & \cdots & p_1 \\ p_1 & p_0 & p_{n-1} & \cdots & & \cdots & p_2 \\ p_2 & p_1 & p_0 & \ddots & & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & & \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots \\ p_{n-2} & p_{n-3} & \cdots & & \ddots & p_0 & p_{n-1} \\ p_{n-1} & p_{n-2} & \cdots & & \cdots & p_1 & p_0 \end{bmatrix}}^{\mathbf{F} = \text{circul}(\mathbf{p}), \quad \mathbf{p} = (p_0, \dots, p_{n-1})} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \\ \vdots \\ x_{n-1} \end{bmatrix}$$

So $(\mathbf{F})_{ij} = p_{i-j}$, $1 \le i, j \le n$, and $p_j = p_{j+n}$ for $1 - n \le j < 0$. So

$$y_k = \sum_{j=0}^{n-1} p_{k-j}x_j$$

Note that the coefficients $p_0, \dots, p_{n-1}$ represent the *periodic impulse response*, but do not (necessarily) agree with the *impulse response*. They satisfy the following relationship

$$p_j = \sum_{k=0}^{\lfloor \frac{m-j}{n} \rfloor} h_{j+nk}, \quad j \in \{0, \dots, n-1\},$$

if $(\dots, 0, h_0, \dots, h_m, 0, \dots)$ is the *impulse response* of the filter F. This process is called the n-periodic convolution.

D. (n-Periodic Discrete Convolution) Given two n-periodic sequences $(p_k)_{k\in\mathbb{Z}}$ and $(x_k)_{k\in\mathbb{Z}}$ the n-periodic convolution yields the n-periodic sequence:

$$(y_k)_{k\in\mathbb{Z}} = (p_k)_{k\in\mathbb{Z}} \star_n (x_k)_{k\in\mathbb{Z}}$$
$$y_k := \sum_{j=0}^{n-1} p_{k-j}x_j = \sum_{j=0}^{n-1} x_{k-j}p_j, \quad k \in \mathbb{Z}$$

Or in matrix-vector notation we have

$$\mathbf{y} = \mathbf{p} \star_n \mathbf{x} = \text{circul}(\mathbf{p})\mathbf{x} = \text{circul}(\mathbf{x})\mathbf{p} = \mathbf{x} \star_n \mathbf{p}.$$

Note the commutativity of $\star_n$.

Periodic Convolution $\overset{\sim}{=}$ mult. w. a circulant matrix

---

D. (Circulant Matrix) A matrix $\mathbf{C} = [c_{ij}]_{i,j=1}^n \in \mathbb{K}^{n\times n}$ is *circulant* iff

$$\exists (p_j)_{j\in\mathbb{Z}}\colon \begin{array}{l} (p_j)_{j\in\mathbb{Z}} \text{ is an } n\text{-periodic sequence} \\ \wedge \quad \forall i, j, 1 \le i, j \le n\colon c_{ij} = p_{j-i}. \end{array}$$

9.4 Disc. Conv. via Periodic Disc. Conv.

We want to compute the discrete convolution

$\mathbf{y} = \mathbf{h} \star \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{h} \in \mathbb{R}^m$,

through a function that computes the periodic convolution. In order to get the right result we have to choose a sufficently large period, such that the convolutions do not interfere:

$$p = 2\max\{m,n\} - 1$$

Then we can compute the discrete convolution through the periodic convolution by making use of 0-padding:

$$\widetilde{\mathbf{h}} = \begin{bmatrix}\mathbf{h}\\ \mathbf{o}\end{bmatrix} \in \mathbb{R}^p, \quad \widetilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x}\\ \mathbf{o}\end{bmatrix} \in \mathbb{R}^p,$$

Then we have:

$$\mathbf{y} = \mathbf{h} \star \mathbf{x} = (\widetilde{\mathbf{h}} \star_p \widetilde{\mathbf{x}})_{1:(m+n-1)}$$

9.5 Discrete Fourier Transforms

Observation: All circulant matrices in $\mathbb{R}^{n\times n}$ have the same eigenvectors (unit length) but different eigenvalues.

D. (n-th Root of Unity)

$\omega_n := e^{-i\frac{2\pi}{n}} = \cos\left(\frac{2\pi}{n}\right) - i\sin\left(\frac{2\pi}{n}\right)$

Properties: $\sum_{k=0}^{n-1}\omega_n^{kj} = \begin{cases} n, & \text{if } j \equiv_n 0,\\ 0, & \text{otherwise.}\end{cases}$

$\omega_n^n = 1 \quad \omega_n^{-j} = \overline{\omega_n^j} \quad \omega_j^{\frac{1}{2}} = -1 \quad \forall k \in \mathbb{Z}\colon \omega_n = \omega_n^{k+n}$

D. (Fourier Matrix) The fourier matrix

$$\mathbf{F}_n := \left[\omega_n^{\ell j}\right]_{\ell,j=0}^{n-1} \in \mathbb{C}^{n\times n}$$

contains the eigenvectors $\{\mathbf{v}_0,\dots,\mathbf{v}_n\}$ of any circulant matrix in $\mathbb{C}^{n\times n}$.

$$\mathbf{F}_n = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\ 1 & \omega_n^4 & \omega_n^8 & \omega_n^{12} & \cdots & \omega_n^{4(n-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_n^{n-2} & \omega_n^{2(n-2)} & \omega_n^{3(n-2)} & \cdots & \omega_n^{(n-1)(n-2)} \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{matrix}\mathbf{v}_0^\top\\ \mathbf{v}_1^\top\\ \mathbf{v}_2^\top\\ \mathbf{v}_3^\top\\ \mathbf{v}_4^\top\\ \\ \mathbf{v}_{n-2}^\top\\ \mathbf{v}_{n-1}^\top\end{matrix}$$

$$\mathbf{v}_0 \quad \mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3 \quad \cdots \quad \mathbf{v}_{n-1}$$

Com. Note that the eigenvectors do not have unit length!

Com. The column vectors are called the *trigonometric basis*.

The matrix $\mathbf{F}_n$ has the following properties:

$\mathbf{F}_n = \mathbf{F}_n^\top$(symmetric) $\qquad \mathbf{F}_n \neq \mathbf{F}_n^\mathsf{H}$(not hermitian)

$\frac{1}{\sqrt{n}}\mathbf{F}_n$ is *unitary* $\qquad (\frac{1}{\sqrt{n}}\mathbf{F}_n)^\mathsf{H}(\frac{1}{\sqrt{n}}\mathbf{F}_n) = \mathbf{I}$

$(\frac{1}{\sqrt{n}}\mathbf{F}_n)^{-1} = \frac{1}{\sqrt{n}}\mathbf{F}_n^\mathsf{H} = \frac{1}{\sqrt{n}}\overline{\mathbf{F}}_n^\top = \frac{1}{\sqrt{n}}\overline{\mathbf{F}}_n$

$\mathbf{F}_n^\mathsf{H}\mathbf{F}_n = n\cdot\mathbf{I}$ and $(\mathbf{F}_n)^{-1} = \frac{1}{n}\overline{\mathbf{F}}_n$, because

$$(\mathbf{F}_n)^{-1} = \left(\frac{\sqrt{n}}{\sqrt{n}}\mathbf{F}_n\right)^{-1} = \frac{1}{\sqrt{n}}\left(\frac{1}{\sqrt{n}}\mathbf{F}_n\right)^{-1} = \frac{1}{\sqrt{n}}\left(\frac{1}{\sqrt{n}}\mathbf{F}_n\right)^\mathsf{H} = \frac{1}{n}\mathbf{F}_n^\mathsf{H} = \frac{1}{n}\overline{\mathbf{F}}_n.$$

D. (Discrete Fourier Transform (DFT))

A DFT (*forward* transform) is the linear map

$$\mathcal{F}_n\colon \mathbb{C}^n \to \mathbb{C}^n, \quad \mathbf{y} \mapsto \mathbf{F}_n\mathbf{y} \in \mathbb{C}^n.$$

So we get

$$\mathbf{c} = \mathbf{F}_n\mathbf{y} \qquad c_k := \sum_{j=0}^{n-1} y_j\omega_n^{kj}, \quad k = 0,\dots,n-1.$$

And using the inverse of $\mathbf{F}_n$, we get the *inverse* DFT

$$\mathbf{y} = \frac{1}{n}\overline{\mathbf{F}}_n\mathbf{c} \qquad y_k = \frac{1}{n}\sum_{j=0}^{n-1} c_j\omega_n^{-kj}, \quad k = 0,\dots,n-1.$$

Com. $\mathbf{F}_n^{-1} = \frac{1}{n}\overline{\mathbf{F}}_n$.

L. (Diagonalization of Circulant Matrices)

Any circulant matrix $\mathbf{C} := \mathrm{circul}(\mathbf{u}) \in \mathbb{K}^{n\times n}$ can be diagonalized as follows:

$$\mathbf{C} = \frac{1}{n}\overline{\mathbf{F}}_n\,\mathrm{diag}(\mathbf{F}_n\mathbf{u})\mathbf{F}_n$$

C. (Multiplication with Circulant Matrices)

The multiplication of $\mathbf{x} \in \mathbb{R}^n$ with a circulant matrix $\mathbf{C} := \mathrm{circul}(\mathbf{u}) \in \mathbb{K}^{n\times n}$ can be expressed as follows:

$$\begin{aligned}\mathbf{u} \star_n \mathbf{x} = \mathbf{C}\mathbf{x} &= \frac{1}{n}\overline{\mathbf{F}}_n\,\mathrm{diag}(\mathbf{F}_n\mathbf{u})\mathbf{F}_n\mathbf{x}\\ &= \mathrm{invdft}(\mathrm{dft}(\mathbf{u}) \odot \mathrm{dft}(\mathbf{x})).\end{aligned}$$

9.6 Fast Fourier Transform

$\mathcal{O}(n\log(n))$ for inverse and forward.

9.7 Toeplitz Matrix Techniques

See book on how to estimate the parameters of a filter.

10 Interpolation

D. (Interpolation Problem)

Given $(t_i, y_i)_{i=0}^n \in I \times \mathbb{R}$

Seeked Interpolant f, $f \in C^0(I)$ that satisfies the interpolation conditions: $\forall i \in \{0,\dots,n\} : f(t_i) = y_i$.

10.1 Interpolation in General

D. (Cardinal Basis) A *cardinal basis* $\{b_0,\dots,b_n\}$ (set of functions) for an interpolation problem satisfies the following:

$$\forall i,j \in \{0,\dots,n\} : \quad b_i(t_j) = \delta_{ij} := \begin{cases}1, & i = j,\\ 0, & i \neq j.\end{cases}$$

If we have $n+1$ basis functions $\{b_0,\dots,b_n\}$ and $n+1$ points $(t_i,y_i)_{i=0}^n$, then the interpolation problem has a unique solution $\boldsymbol{\alpha}$ (assuming the nodes are pairwise different):

$$\mathbf{A}\boldsymbol{\alpha} = \mathbf{y} \iff \begin{bmatrix} b_0(t_0) & \cdots & b_n(t_0)\\ b_0(t_1) & \cdots & b_n(t_1)\\ \vdots & \ddots & \vdots\\ b_0(t_n) & \cdots & b_n(t_n)\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\\ \vdots\\ \alpha_n\end{bmatrix} = \begin{bmatrix}y_0\\ y_1\\ \vdots\\ y_n\end{bmatrix}$$

So the solution is $\boldsymbol{\alpha} = \mathbf{A}^{-1}\mathbf{y}$. As we can see it's obtained through a linear map $\mathbf{A}^{-1}$. The interpolant is thus determined through the linear mapping (which we call an interpolation scheme):

$$I_\mathcal{T}\colon \mathbb{R}^n \to C^0(I)$$

$$\mathbf{y} \mapsto f(t) = \sum_{j=0}^n (\mathbf{A}^{-1}\mathbf{y})_j b_j(t)$$

given a fixed set of nodes $\mathcal{T} = \{t_0,\dots,t_n\}$.

Now if $\{b_0,\dots,b_n\}$ is a *cardinal basis* for the interpolation problem, then $\mathbf{A} = \mathbf{I}$, and thus we have

$$f(t) = \sum_{j=0}^n y_j b_j(t).$$

– 10.2 – (Global) Polynomial Interpolation

D. (Vector Space $\mathcal{P}_n$)

$$\mathcal{P}_n := \left\{ t \mapsto \sum_{j=0}^{n} \alpha_j t^j \;\middle|\; \alpha_0, \ldots, \alpha_n \in \mathbb{R} \right\}$$

C. $\dim(\mathcal{P}_n) = n + 1$.

D. (Monomial Basis for $\mathcal{P}_n$) $\{t \mapsto t^k\}_{k=0}^{n}$

D. (Eval. of Polynomials with Horner Scheme)

$p(t) = t\cdot(\cdots(t\cdot(t\cdot(\alpha_k t+\alpha_{k-1})+\alpha_{k-2}\cdots)+\alpha_2)+\alpha_1)+\alpha_0$

Com. $\mathcal{O}(n)$

– 10.2.1 – Lagrange Interpolation

The cardinal basis for an interpolation problem (with distinct increasing nodes, and as above) is given trough the lagrange polynomials $\{L_i\}_{i=0}^{n}$.

$$L_i(t) = \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{(t-t_j)}{(t_i-t_j)} \in \mathcal{P}_n.$$

Com. It's easy to see that $\forall i,j \in \{0,\ldots,n\} : L_i(t_j) = \delta_{ij}$.

Then the interpolant is given by

$$f(t) = \sum_{i=0}^{n} y_i L_i(t) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{(t-t_j)}{(t_i-t_j)}.$$

– 10.3 – Algorithms for Poly. Interpolation

See book: Aitken Neville, Newton Scheme, . . .

11 Approx. of Functions in 1D

D. (Approx. Scheme) = Sampling + Interpolation

$f\colon I \subset \mathbb{R} \to \mathbb{R} \overset{\text{sampling}}{\to} (t_i, y_i := f(t_i))_{i=0}^{n} \overset{\text{interpolation}}{\to} \hat{f} := I_{\mathcal{T}}\mathbf{y}\ (\hat{f}(t_i) = y_i)$

Com. Now we the freedom to choose the points.

T. (L^∞ Polynomial Best Approximation Estimate)

If $f \in C^r([-1,1])$ (r times continuously differentiable), $r \in \mathbb{N}$, then, for any polynomial of degree $n \geq r$,

$$\inf_{p\in\mathcal{P}_n} \|f-p\|_{L^\infty([-1,1])} \leq (1+\pi^2/2)^r \frac{(n-r)!}{n!} \left\|f^{(r)}\right\|_{L^\infty([-1,1])}$$
$$\leq C(r) n^{-r} \left\|f^{(r)}\right\|_{L^\infty([-1,1])}.$$

Com. So we have algebraic convergence $\mathcal{O}(n^{-r})$ if we can somehow bound the norm of the derivative!

So we'll study families of approximation schemes $\{A_n\}$ and see how $\|f - A_n f\|$ behaves as a function of $n \to \infty$.

– 11.1 – Affine Transf. of Approx. Schemes

Let's say we have an affine linear map

$$\Phi\colon [a,b] \to [c,d]$$

(that maps intervals as with numerical quadrature), and the pullback

$$\Phi^*\colon C^0([c,d]) \to C^0([a,b])$$

then we can use an approximation scheme A on $[a,b]$ to create an approximation scheme $\hat{A}$ on $[c,d]$ as follows:

$$A\colon C^0([a,b]) \to \mathcal{P}_n([a,b]), \quad f \mapsto A(f)$$
$$\hat{A}\colon C^0([c,d]) \to \mathcal{P}_n([c,d]), \quad f \mapsto ((\Phi^*)^{-1} \circ A \circ \Phi)(f)$$

– 11.1.1 – Norms under Affine Pullbacks

$\|f\|_{L^\infty([c,d])} = \|\Phi^* f\|_{L^\infty([a,b])}$

$\|f - Af\|_{L^\infty([c,d])} = \left\|\Phi^* f - \hat{A}(\Phi^* f)\right\|_{L^\infty([a,b])}$

Since for the derivative of the pullback it holds that $(\Phi^* f)(t)^{(k)} = f^{(k)}(\Phi(t)) \cdot (\Phi'(t))^k = (\Phi^* f^{(k)})(t) \cdot (\Phi'(t))^k$. we have

$$\left\|(\Phi^* f)^{(r)}\right\|_{L^\infty([a,b])} \overset{\text{deriv.}}{=} \left\|\left(\Phi^* f^{(r)}\right) \cdot \Phi'^r\right\|_{L^\infty([a,b])}$$
$$\overset{\text{norm}}{=} \left\|\left(\Phi^* f^{(r)}\right)\right\|_{L^\infty([a,b])} \cdot \|\Phi'\|^r_{L^\infty([a,b])}$$
$$\overset{\text{pullb.}}{=} \left\|f^{(r)}\right\|_{L^\infty([c,d])} \cdot \|\Phi'\|^r_{L^\infty([a,b])}$$
$$= \ldots$$

Note that Φ' on $[a,b]$ is a constant.

– 11.1.2 – L^∞ Poly. Best. App. Est. on Arb. Int.

T. (L^∞ Poly. best app. est. on arb. interval)

If $f \in C^r([a,b])$ (r times continuously differentiable), $r \in \mathbb{N}$, then, for any polynomial of degree $n \geq r$,

$$\inf_{p\in\mathcal{P}_n} \|f-p\|_{L^\infty([a,b])} = \inf_{p\in\mathcal{P}_n} \|\Phi^*(f-p)\|_{L^\infty([-1,1])}$$
$$= \inf_{p\in\mathcal{P}_n} \|\Phi^* f - \Phi^* p)\|_{L^\infty([-1,1])} = \inf_{p\in\mathcal{P}_n} \|(\Phi^* f) - p\|_{L^\infty([-1,1])}$$
$$\leq (1+\pi^2/2)^r \frac{(n-r)!}{n!} \left\|\Phi^* f^{(r)}\right\|_{L^\infty([-1,1])}$$
$$= C(r) \left(\frac{b-a}{n}\right)^r \left\|f^{(r)}\right\|_{L^\infty([a,b])}.$$

– 11.2 – Lagrangian Approximation Schemes

D. (Lagrangian Approximation) Is just an approximation scheme denoted by $L_{\mathcal{T}} f$ for a function f that (picks some nodes in some way) and then uses Lagrange interpolation.

Com. For instance, one could use equidistant nodes.

– 11.2.1 – Convergence of Approximation Schemes

D. (Algebraic Convergence)

$\|f - A_n f\| = \mathcal{O}(n^{-p})$, with rate $p > 0$.

D. (Exponential convergence)

$\|f - A_n f\| = \mathcal{O}(q^n)$, with $0 < q < 1$.

How to Detect the type of Convergence

- **Algebraic Convergence** $\epsilon_i \approx C n_i^{-p}$
  Affine linear relationship in a *log-log* scale:
  $$\log(\epsilon_i) \approx \log(C) - p\log(n_i)$$
  We then just apply linear regression for the data points $(\log n_i, \log \epsilon_i)$ to get a lsq estimate for the rate p.
- **Exponential Convergence** $\epsilon_i \approx C e^{-\beta n_i}$
  Affine linear relationship in a *lin-log* scale:
  $$\log(\epsilon_i) \approx \log(C) - \beta n_i$$
  We then just apply linear regression for the data points $(n_i, \log \epsilon_i)$ to get a lsq estimate for the rate $q := e^{-\beta}$.

– 11.2.2 – Representation of Interpolation Error

See book.

12 Numerical Quadrature

D. (n-point Quadrature Formula)

$$I := \int_a^b f(t)\,dt \approx \sum_{j=0}^{n} w_j^{(n)} f(c_j^{(n)}) =: Q_n(f).$$

Com. $\text{Cost}(Q_n) = n \cdot \text{Cost}(f_{\text{eval}})$.

12.1 Pullback to Reference Interval

Usually $[a,b]=[-1,1]$ or $[a,b]=[0,1]$.

$\Phi\colon [a,b] \to [c,d] \quad t \mapsto c + \frac{d-c}{b-a}\cdot(t-a)$

$\Phi'\colon [a,b] \to [c,d] \quad t \mapsto \frac{d-c}{b-a}$

$\Phi^{-1}\colon [c,d] \to [a,b] \quad x \mapsto a + \frac{b-a}{d-c}\cdot(x-c)$

Now the pullback transforms any functions as follows: $\Phi^*\colon C^0([c,d]) \to C^0([a,b])$. So for $f(t) \in C^0([c,d])$, $(\Phi^* f)(t) = (f\circ\Phi)(t) = f(\Phi(t)) \in C^0([a,b])$.

$(\Phi^*)^{-1}\colon C^0([a,b]) \to C^1([c,d]) \quad (\Phi^*)^{-1} f = f\circ(\Phi^*)^{-1}$

Now this is the integral that we would like to compute on the interval $[c,d]$ for a specific integrand $f \in C^0([c,d])$

$$I = \int_c^d f(x)\,dx.$$

Now we don't know any quadrtature weights and nodes for the interval $[c,d]$, so we pull back the integral to the interval $[a,b]$, because for any function g on the interval $[a,b]$ we know the following quadrature formula (weights and nodes) that approximates the integral of any integrand $g \in C^0([a,b])$ on $[a,b]$ the best. So for any $g \in C^0([a,b])$ we know the optimal weights and nodes for an n-point quadrature formula.

$$\int_a^b g(t)\,dt \approx \sum_{i=1}^{n} w_i^{(n)} f\left(c_i^{(n)}\right) = Q_n(g)$$

So we pull back the integral to the interval $[a,b]$ and scale the result to obtain the original I. To pull the integral from $[c,d]$ to the reference interval $[a,b]$ we use the following substitution:

$$x = \Phi(t) \iff t = \Phi^{-1}(x)$$
$$dx = \Phi'(t)$$

So this gives us

$$I = \int_c^d f(x)\,dx = \int_{a=\Phi^{-1}(c)}^{b=\Phi^{-1}(d)} f(\Phi(t))\cdot\Phi'(t)\,dt$$

$$= \overbrace{\frac{d-c}{b-a}}^{=\Phi'(t)} \cdot \int_a^b \overbrace{f(\Phi(t))}^{=g(t)=(\Phi^* f)(t)}\,dt$$

Now we have a function $g(t) = (\Phi^* f)(t) = f(\Phi(t))$ that we integrate over the reference interval $[a,b]$, so we can use the quadrature weights and nodes to determine the integral.

$$\approx \frac{d-c}{b-a}\sum_{i=1}^{n} w_i^{(n)} g\left(c_i^{(n)}\right) = \frac{d-c}{b-a} Q_n(g)$$

Or we can write the quadrature formula in terms of f with other weights and nodes

$$= \frac{d-c}{b-a}\sum_{i=1}^{n} w_i^{(n)} f\left(\Phi\left(c_i^{(n)}\right)\right)$$

$$= \sum_{i=1}^{n} \widehat{w}_i^{(n)} f\left(\widehat{c}_i^{(n)}\right) = \widehat{Q}_n(f). \qquad \widehat{w}_i^{(n)} = \Phi'\cdot w_i^{(n)} = \tfrac{d-c}{b-a} w_i^{(n)}, \quad \widehat{c}_i^{(n)} = \Phi\left(c_i^{(n)}\right)$$

where $\widehat{Q}_n$ is a quadrature formula on the interval $[c,d]$ for any function (here we use f).

---

13 Iterative Methods

D. (Newton's Method)

$F\colon \mathbb{R}^n \to \mathbb{R}^n$

$\widetilde{F}\colon \mathbb{R}^n \to \mathbb{R}^n$ (affine approx of F at $\mathbf{x}^{(k)}$)

$\mathbf{x} \mapsto F(\mathbf{x}) + DF(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)})$

The objective is to pick the next $\mathbf{x}^{(k+1)}$ as the zero of $\widetilde{F}$.

$$\mathbf{x}^{(k+1)} := \Phi(\mathbf{x}^{(k)} = \mathbf{x}^{(k)} \overbrace{-DF(\mathbf{x}^{(k)})^{-1}F(\mathbf{x}^{(k)})}^{\text{Newton Correction}}.$$

where Φ is the SSM function and DF is usually a jacobian matrix evaluated at $\mathbf{x}^{(k)}$.

14 Numerical Integration

D. (First-Order ODE) $\dot{\mathbf{y}}(t) = \mathbf{f}(t,\mathbf{y}(t))$

D. (Autonomous ODE) $\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t))$

D. (IVP) $\dot{\mathbf{y}}(t) = \mathbf{f}(t,\mathbf{y}(t))$, $\mathbf{y}(t_0) = \mathbf{y}_0$

D. (Autonomous IVP) $\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t))$, $\mathbf{y}(0) = \mathbf{y}_0$

T. (Time-Invariance of Autonomous ODEs)

If $t \mapsto \mathbf{y}(t)$ is a solution of an anutonomous ODE, then for any $\tau \in \mathbb{R}$, the shifted function $t \mapsto \mathbf{y}(t-\tau)$ is also a solution. Thus we can always make the canonical choice $t_0 = 0$.

14.1 Conversion Techniques

14.1.1 Autonomization

We convert $\dot{\mathbf{y}}(t) = \mathbf{f}(t,\mathbf{y}(t)) \in \mathbb{R}^d$ into an autonomous ODE of the form $\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t))$ by defining

$$\mathbf{z}(t) := \begin{bmatrix} | \\ \mathbf{y}(t) \\ | \\ t \end{bmatrix} = \begin{bmatrix} | \\ \widetilde{\mathbf{z}}(t) \\ | \\ z_{d+1} \end{bmatrix} \in \mathbb{R}^{d+1}$$

So, since $\frac{d}{dt}t = 1$, we get the autonomous ODE

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} | \\ \dot{\mathbf{y}}(t) \\ | \\ 1 \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{f}(z_{d+1}(=t), \widetilde{\mathbf{z}}(t)(=\mathbf{y}(t))) \\ | \\ 1 \end{bmatrix}$$

And now the first d coefficients of the solution $\mathbf{z}(t)$ will give us $\mathbf{y}(t)$.

14.1.2 Higher Order to First Order

Convert the ODE $\mathbf{y}^{(n)} = f(t, \mathbf{y}(t), \dot{\mathbf{y}}(t), \dots, \mathbf{y}^{(n-1)(t)}) \in \mathbb{R}^d$ as follows:

$$\mathbf{z}(t) := \begin{bmatrix} t \\ \mathbf{y}(t) \\ \mathbf{y}^{(1)}(t) \\ \vdots \\ \mathbf{y}^{(n-1)}(t) \end{bmatrix} = \begin{bmatrix} z_0 \\ \mathbf{z}_1(t) \\ \mathbf{z}_2(t) \\ \vdots \\ \mathbf{z}_n(t) \end{bmatrix} \in \mathbb{R}^{nd}$$

Then the derivative of $\mathbf{z}(t)$ is

$$\dot{\mathbf{z}}(t) = \mathbf{g}(\mathbf{z}(t)) = \begin{bmatrix} 1 \\ \mathbf{z}_2(t) \\ \vdots \\ \mathbf{z}_n(t) \\ \mathbf{f}(z_0, \mathbf{z}_1(t), \dots, \mathbf{z}_n(t)) \end{bmatrix}.$$

And the solution is given by the d rows after the first row of $\mathbf{z}$. Note that for an IVP the initial values for $\mathbf{y}(t_0), \dot{\mathbf{y}}(t), \dots, \mathbf{y}^{(n-1)}(t)$ have to be specified.

14.2 Evolution Operators

D. (Evolution Operator)

The evolution operator for an autonomous ODE $\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t))$ is a mapping of points in state space $D \subset \mathbb{R}^d$:

$$\mathbf{\Phi}^t : D \to D$$
$$\mathbf{y}_0 \mapsto \mathbf{y}(t)$$

where $t \mapsto \mathbf{y}(t)$ is the solution of the IVP. We may also let t vary, which spawns a *family* of mappings $\{\mathbf{\Phi}^t\}$ of the state space onto itself. However, it can also be viewed as a mapping with two arguments, a duration t and an initial state value $\mathbf{y}_0$.

$$\mathbf{\Phi} : \mathbb{R} \times D \to D$$
$$(t, \mathbf{y}_0) \mapsto \mathbf{\Phi}^t \mathbf{y}_0 := \mathbf{y}(t)$$

where $t \mapsto \mathbf{y}(t) \in C^1(\mathbb{R}, \mathbb{R}^d)$ is the unique (global) solution of the IVP $\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t))$, $\mathbf{y}(0) = \mathbf{y}_0$.

Com. Note that $t \mapsto \mathbf{\Phi}^t \mathbf{y}_0$ describes the solution $\mathbf{y}(t)$ of $\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t))$ for $\mathbf{y}(0) = \mathbf{y}_0$ (a trajectory). Therefore, by virtue of definition, we have

$$\frac{\partial \mathbf{\Phi}(t, \mathbf{y})}{\partial t} = \frac{d\mathbf{y}(t)}{dt} = \dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t)) = \mathbf{f}(\mathbf{\Phi}^t \mathbf{y}).$$

14.3 Polygonal Approximation of ODEs

14.3.1 Objectives

- Given $(t_0, \mathbf{y}_0)$ approximate $\mathbf{y}(T)$ at final time T.
- Approximate the trajectory $t \mapsto \mathbf{y}(t)$ for an IVP.

14.3.2 Temporal Meshes

D. (Temporal Mesh)

$\mathcal{M} := \{t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N := T\} \subset [t_0, T]$

Com. In this lecture we treat examples where we assume that the interval of interest is contained in the solution of the IVP.

14.3.3 Explicit Euler Method

For $t \in [t_k, t_{k+1}]$ we assume (fwd. diff. quot.)

$$\dot{\mathbf{y}}(t) \approx \frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{t_{k+1} - t_k} = \mathbf{f}(t_k, \mathbf{y}_k) \approx \mathbf{f}(t_k, \mathbf{y}(t_k))$$

Which gives the recursion

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_k, \mathbf{y}_k), \qquad k = 0, \dots, N-1$$

14.3.4 Implicit Euler Method

For $t \in [t_k, t_{k+1}]$ we assume (bw. diff. quot.)

$$\dot{\mathbf{y}}(t) \approx \frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{t_{k+1} - t_k} = \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) \approx \mathbf{f}(t_{k+1}, \mathbf{y}(t_{k+1}))$$

Which gives the equation

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}), \qquad k = 0, \dots, N-1$$

Com. May involve solving a LSE for $\mathbf{y}_{k+1}$.

14.3.5 Implicit Midpoint Method

Using the symmetric difference quotient:

$\dot{\mathbf{y}}(t) \approx \frac{\mathbf{y}(t+h) - \mathbf{y}(t-h)}{2h}$

and the approx. lin. of $\mathbf{y}$ around t we get for $t \in [t_k, t_{k+1}]$

$\dot{\mathbf{y}}(t) \approx \frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{h_k} = \mathbf{f}\left(\frac{1}{2}(t_k + t_{k+1}), \frac{1}{2}(\mathbf{y}_k + \mathbf{y}_{k+1})\right) \approx \mathbf{f}\left(\frac{1}{2}(t_k + t_{k+1}), \mathbf{y}\left(\frac{1}{2}(t_k + t_{k+1})\right)\right)$

Which gives the equation for $k = 0, \dots, N-1$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \mathbf{f}\left(\tfrac{1}{2}(t_k + t_{k+1}), \tfrac{1}{2}(\mathbf{y}_k + \mathbf{y}_{k+1})\right)$$

14.3.6 Euler Polygon from Approximations

Given the Approximation $\mathbf{y}_0, \dots, \mathbf{y}_N$ we can create the approximating Euler Polygon for Φ^t as follows:

$\mathbf{y}_h : [t_0, t_N] \to \mathbb{R}^d$

$$t \mapsto \mathbf{y}_k \frac{t_{k+1} - t}{t_{k+1} - t_k} + \mathbf{y}_{k+1} \frac{t - t_k}{t_{k+1} - t_k} \quad \text{for } t \in [t_k, t_{k+1}].$$

14.4 General Single-Step Methods

D. (Discrete Evolution) The methods above describe how to obtain $\mathbf{y}_{k+1}$ from $\mathbf{y}_k$ - so in some sense, for a timestep h, they describe a mapping $\mathbf{\Psi}$ that approximates the Evolution operator $\mathbf{\Phi}$ discretely, so $\mathbf{\Psi}(h, \mathbf{y}) \approx \Phi^h \mathbf{y}$. That's why we call it discrete evolution.

14.5 Convergence of SSMs

D. (Discretization Error) $\epsilon_N := \|\mathbf{y}(T) - \mathbf{y}_N\|$.

We study the *asymptotic* error for mostly equidistant meshes $\mathcal{M}_N := \left\{t_k := \frac{k}{N} T \mid k = 0, \dots, N\right\}$ in terms of $h \to 0$. Usually the error converges algebraically in terms of the stepsize, so $\epsilon_N \in \mathcal{O}(h^p)$, where p is called the *order* of the method. If we know the exact value, and we want to estimate the rate, we can do this as follows in each approximation step:

$$p \approx \log_2 \left(\frac{\epsilon_{\text{old}}}{\epsilon_{\text{new}}}\right)$$

14.6 Explicit Range Kutta Methods

Basic Idea:

Let's say we have an IVP $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t))$, $\mathbf{y}(0) = \mathbf{y}_0$. Then we know by the fundamental theorem of calculus:

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + \overbrace{\int_{t_k}^{t_{k+1}} \mathbf{f}(\tau, \mathbf{y}(\tau))\, d\tau}^{\text{Fund. Thm.: } \mathbf{y}(t_{k+1}) - \mathbf{y}(t_k)}.$$

D. (Explicit Runge-Kutta Method)

For $b_i, a_{ij} \in \mathbb{R}$, $c_i := \sum_{j=1}^{i-1} a_{ij}$, $i, j = 1, \dots, s$, $s \in \mathbb{N}$, an *s-stage explicit Runge-Kutta single step method* (RK-SSM) for the ODE $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t))$, $\mathbf{f} : \Omega \to \mathbb{R}^d$, is defined by $(\mathbf{y}_0 \in D)$

$$\mathbf{k}_i := \mathbf{f}\left(t_0 + h c_i, \mathbf{y}_0 + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j\right), \quad i = 1, \dots, s,$$

$$\mathbf{y}_{k+1} := \mathbf{y}_k + h\sum_{i=1}^{s} b_i\mathbf{k}_i.$$

The vectors $\mathbf{k}_i \in \mathbb{R}^d$, $i = 1,\dots,s$, are called *increments*, $h > 0$ is the size of the timestep.

```
for every time interval from ℓ = 1 to N
  compute the interval width h (may be uniform)
  initialize K to contain the s k_i s
  for i = 1 to s
    compute the k_i
  then compute the next evolution step through
  the quadrature rule:
  y_ℓ := y_ℓ + h Σ_{i=1}^{s} b_i k_i
```

C. (Consistent RK-SSMs)

A s-step RK-SSM is *consistent* with the ODE $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t))$, if and only if, $\sum_{i=1}^{s} b_i = 1$.

Create Higher Order SSMs through Bootstr.

Goal: Convergence of $\mathcal{O}(h^{p+1})$

Given: Method with convergence of $\mathcal{O}(h^{p})$

In short: Since in the quadrature we multiply by h, if we use the other method to approximate the evaluations of the quadrature, we'll get a method of $\mathcal{O}(h^{p+1})$.

Butcher Scheme Notation for Explicit RK-SSM

$$\left[\begin{array}{c|c} \mathbf{c} & \mathbf{U} \\ \hline & \mathbf{b}^\top \end{array}\right] = \left[\begin{array}{c|cccc} c_1 & 0 & \cdots & \cdots & 0 \\ c_2 & a_{21} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array}\right] \in \mathbb{R}^{(s+1)\times(s+1)}$$

14.7 Why High Order Met. is Desirable

Let's assume that we know the order of one method

$$err(h_{\text{old}}) \approx Ch^p$$

for a meshwidth h_{old}. Now we want to reduce the meshwidth, such that we get an asymptotic error reduction of

$$\frac{err(h_{\text{new}})}{err(h_{\text{old}})} \overset{!}{=} \frac{1}{\rho} \quad \text{for reduction factor } \rho > 1.$$

Then we have

$$\frac{err(h_{\text{new}})}{err(h_{\text{old}})} = \frac{C\cdot h_{\text{new}}^p}{C\cdot h_{\text{old}}^p} = \left(\frac{h_{\text{new}}}{h_{\text{old}}}\right)^p \overset{!}{=} \frac{1}{\rho}$$
$$\iff \quad h_{\text{new}} := \rho^{-\frac{1}{p}} h_{\text{old}} = \frac{1}{\sqrt[p]{\rho}} h_{\text{old}}$$

Now this tells us that if we want to decrease the error by a factor of ρ, we have to decrease h_{new} as above. Now, the larger the order p, the less we have to reduce h_{new} to get a prescribed (relative) reduction of the error!

15 SSMs for Stiff IVPs

15.1 Stability of $\dot{y} = \lambda y$ for Expl. RK

T. (Stability Function of Explicit RK-Methods)

For a Butcher scheme

$$\left[\begin{array}{c|c} \mathbf{c} & \mathbf{U} \\ \hline & \mathbf{b}^\top \end{array}\right]$$

the recursions for k_i and y_{k+1} gives us the following system

$$\begin{bmatrix} \mathbf{I} - z\mathbf{U} & \mathbf{o} \\ -z\mathbf{b}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ y_{k+1} \end{bmatrix} = y_k \begin{bmatrix} \mathbf{1} \\ 1 \end{bmatrix} \qquad \begin{aligned} \mathbf{k} &= (k_1,\dots,k_s)^\top/\lambda \\ z &= \lambda h \\ \mathbf{1} &= (1,\dots,1)^\top \end{aligned}$$

Which gives us

$$y_{k+1} = \underbrace{\left(1 + z\mathbf{b}^\top(\mathbf{I} - z\mathbf{U})^{-1}\mathbf{1}\right)}_{=S(z)=S(\lambda h)} y_0.$$

The discrete evolution $\mathbf{\Psi}^h$ of an explicit s-stage RK SSM with the upper Butcher scheme for the ODE $\dot{y}(t) = \lambda y$ amounts to a multiplication with the number

$$y_{k+1} = \mathbf{\Psi}_\lambda^h = S(\lambda h) y_k$$

where S is the *stability function*

$$S(z) := \underbrace{1 + z\mathbf{b}^\top(\mathbf{I} - z\mathbf{U})^{-1}\mathbf{1}}_{\text{solving LSE w. block elim.}} = \underbrace{\det\left(\mathbf{I} - z\mathbf{U} + z\mathbf{1}\mathbf{b}^\top\right)}_{\text{solving LSE w. Cram. Rule}}$$

with $z = \lambda h$. So we have $y_k = S(z)^k y_0$.

- $|S(\lambda h)| > 1 \implies$ blow-up
- $|S(\lambda h)| \leq 1 \implies$ stable approx.

for general RK-methods.

The trick is to pick h sufficiently small if λ is big! So, we're

C. A stability function $S(z)$ for a consistent s-step explicit RK-method is a non-constant polynomial in z of degree $\leq s$, $S(z) \in \mathcal{P}_s$.

Stability Function for Specific RK-Methods

- Explicit Euler: $S(z) = 1 + z$
- Explicit Trapezoidal Method: $S(z) = 1 + z + \frac{1}{2}z^2$
- RK4 Method: $S(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4$

Sidenote on Blow-Ups

We know that for a linear ODE $\dot{y} = \lambda y$ the solution is $y(t) = c\cdot e^{\lambda t}$. Now if we say that $\lambda \in \mathbb{C}$, $\lambda = a + bi$, then we have the following:

$$\left\|ce^{\lambda t}\right\| = \|c\|\,\|e\lambda t\| = \|c\|\left\|e^{(a+bi)t}\right\| = \underbrace{\|c\|\left\|e^{at}\right\|}_{=r}\underbrace{\left\|e^{i\overbrace{bt}^{=\varphi}}\right\|}_{=1} = \|$$

Thus, for

- $\mathrm{Re}(\lambda) = a < 0$, we have an exponential decay in our fuction $y(t)$ (decay equation), so $\lambda_k \to 0$ for $k \to \infty$ (we have a exponential decrease). So we have take care that the numerical solution does not blow up, because the exact solution doesn't. That's when we have to make sure we use a small timestep h. Note that the blow-up happens due to the nature of the discrete evolution obtained through the S function - we're exponentiating it.
- $\mathrm{Re}(\lambda) = a > 0$, we have an exponential blow-up in the function $y(t)$ (growth equation), so the exact solution $\lambda_k \to \infty$ for $k \to \infty$ (has a blow-up). So we don't need to worry about a blow-up of the numerical solution (because the exact does too) - this is actually desirable.

15.2 Systems of Linear ODEs: $\dot{\mathbf{y}} = \mathbf{M}\mathbf{y}$

Let's say we have the following ODE (or IVP)

$$\dot{\mathbf{y}} = \mathbf{M}\mathbf{y}, \qquad \mathbf{M} \in \mathbb{R}^{d\times d}, \qquad \mathbf{y}(0) = \mathbf{y}_0$$

Then we can diagonalize $\mathbf{M}$ as follows:

$$\mathbf{M} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1},$$
$$\mathbf{V}, \mathbf{D} \in \mathbb{C}^{d\times d}, \quad \mathbf{V} \text{ regular}, \quad \mathbf{D} = \operatorname{diag}(\lambda_1,\dots,\lambda_d).$$

Then write the ODE as d decoupled scalar linear ODEs

$$\dot{\mathbf{y}} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}\mathbf{y} \iff \underbrace{\mathbf{V}^{-1}\dot{\mathbf{y}}}_{\dot{\mathbf{z}}} = \mathbf{D}\underbrace{\mathbf{V}^{-1}\dot{\mathbf{y}}}_{\mathbf{z}} \iff \dot{\mathbf{z}} = \mathbf{D}\mathbf{z} \iff \begin{matrix} \dot{z}_1 = \lambda_1 z_1 \\ \vdots \\ \dot{z}_d = \lambda_d z_d \end{matrix}$$

And the solution $\mathbf{z}$ is

$$\mathbf{z}(t) = \begin{bmatrix} c_1 e^{\lambda_1 t} \\ \vdots \\ c_d e^{\lambda_d t} \end{bmatrix} = \operatorname{diag}(e^{\lambda_1 t}, \dots, e^{\lambda_d t})\mathbf{c} \quad \begin{matrix}\text{for some const.} \\ \mathbf{c} = (c_1, \dots, c_d)^\top\end{matrix}$$

And we know that $\mathbf{y}(t) = \mathbf{V}\operatorname{diag}(e^{\lambda_1 t}, \dots, e^{\lambda_d t})\mathbf{c}$. So according to the IVP the equation has to satisfy

$$\mathbf{y}(0) = \mathbf{y}_0 = \mathbf{V}\operatorname{diag}(1, \dots, 1)\mathbf{c} = \mathbf{V}\mathbf{I}\mathbf{c} \implies \mathbf{c} = \mathbf{V}^{-1}\mathbf{y}_0$$

Final solution for IVP $\mathbf{y}(t) = \mathbf{V}\operatorname{diag}(e^{\lambda_1 t}, \dots, e^{\lambda_d t})\mathbf{V}^{-1}\mathbf{y}_0$.

Solve it with General RK-Method

So, we have the ODE: $\dot{\mathbf{y}} = \mathbf{M}\mathbf{y} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}\mathbf{y}$.

Now the idea is to transform it as follows with

$$\mathbf{z}_k = \mathbf{V}^{-1}\mathbf{y}_k, \qquad \widehat{\mathbf{k}}_i = \mathbf{V}^{-1}\mathbf{k}_i.$$

So we get the following recursion equations

$$\widehat{\mathbf{k}}_i = \mathbf{D}\left(\mathbf{z}_0 + h\sum_{j=1}^{i-1} a_{ij}\widehat{\mathbf{k}}_j\right), \qquad \mathbf{z}_{i+1} = \mathbf{z}_i + h\sum_{i=1}^{s} b_i\widehat{\mathbf{k}}_i$$

So now again with the diagonalization we end up with decoupled scalar ODEs

$$\dot{\mathbf{z}}_\ell = \lambda_\ell \mathbf{z}_\ell, \qquad \ell = 1, \dots, d.$$

Now using the RK-method we get the discrete evolution which is diagonalized too:

$$\mathbf{y}_{k+1} = \boldsymbol{\Psi}^h\mathbf{y}_k, \qquad (\mathbf{z}_{k+1})_\ell = \boldsymbol{\Psi}^h_\ell(\mathbf{z}_k)_\ell$$

Now in order to avoid the blow-up of the $\mathbf{y}_k$s we can also look at the sequences produced in the $(\mathbf{z}_k)_\ell$ scalar problems. So we have to look at the specific $S(z)$, $z = \lambda h$ for the scalar equation.

Solving it with Explicit Euler Method

Now if we were using the explicit Euler method, the update step would be:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{M}\mathbf{y}_k$$

Now, since we have the diagonalization of $\mathbf{M}$, the update step is:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{V}\mathbf{D}\mathbf{V}^{-1}\mathbf{y}_k$$

And, if we again use $\mathbf{z}_{k+1} = \mathbf{V}^{-1}\mathbf{y}_{k+1}$, then we can do the update step much simpler:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + h\mathbf{D}\mathbf{z}_k = (\mathbf{I} + h\mathbf{D})\mathbf{z}_k$$

$$\iff \begin{matrix} (\mathbf{z}_{k+1})_1 = (1 + h\lambda_1)(\mathbf{z}_k)_1 \\ \vdots \\ (\mathbf{z}_{k+1})_d = (1 + h\lambda_d)(\mathbf{z}_k)_d \end{matrix}$$

So we have an explicit euler recursion step as with linear ODEs. Now the big advantage is that these ODEs $(\dot{\mathbf{z}})_i = \lambda(\mathbf{z})_i$ are decoupled so we know that there is a blow-up if:

$$\begin{aligned} \text{blow-up of } (\mathbf{y}_k) &\iff \exists i \in \{1, \dots, d\} : S(h\lambda_i) > 1 \\ &\iff \exists i \in \{1, \dots, d\} : |1 + h\lambda_i| > 1 \end{aligned}$$

So we have the following time-step constraint for h:

$$\forall i \in \{1, \dots, d\} : \quad h < \frac{2}{|\lambda_i|}.$$

So we have to pick:

$$h < \frac{2}{\max_{i\in\{1,\dots,d\}} |\lambda_i|}.$$

Now if there is one eigenvalue with positive real part, then the exact solution

---

T. ((Abs.) Stab. of Exp. RK-. for LS of ODEs)

The sequence $(\mathbf{y}_k)_k$ of approximation generated by an explicit RK-SSM with stability function S applied to the linear autonomous ODE $\dot{\mathbf{y}} = \mathbf{M}\mathbf{y}$, $M \in \mathbb{C}^{d\times d}$ with uniform timestep $h > 0$ *decays exponentially* for every initial state $\mathbf{y}_0 \in \mathbb{C}^d$, if and only if $|S(\lambda_i h)| < 1$ for all eigenvalues λ_i of $\mathbf{M}$.

Now recall, even if $\mathbf{M} \in \mathbb{R}^{d\times d}$, the eigenvalues $\lambda_i \in \mathbb{C}$ of the diagonalization can be complex. Recall that $z_k = S(\lambda)^k y_0$ so

$$y_k \to 0 \text{ for } k \to \infty \quad \iff \quad |S(\lambda h)| < 1.$$

Hence the modulus $|S(\lambda h)|$ tells us for which combinations of λ and stepsize h we achieve exponential decay $y_k \to 0$ for $k \to \infty$, which is the desirable behavior of the approximations for $\operatorname{Re}\lambda < 0$.

D. (Region of (Absolute) Stability)

Let the discrete evolution Psi for a SSM applied to the scalar linear ODE $\dot{y} = \lambda y$, $y \in \mathbb{C}$, be of the form

$$\Psi^h y = S(z)y, \quad y \in \mathbb{C}, h > 0 \text{ with } z := h\lambda$$

and a function $S\colon \mathbb{C} \to \mathbb{C}$. Then the *region of (absolute) stability of the single step method is given by*

$$S_\Psi := \{z \in \mathbb{C} \mid |S(z)| < 1\} \subset \mathbb{C}.$$

Com. So, an explicit RK-SSM will generate exponentially decaying solution sfor the linear ODE $\dot{\mathbf{y}} = \mathbf{M}\mathbf{y}$, $\mathbf{M} \in \mathbb{C}^{d\times d}$, for every initial state $\mathbf{y} \in \mathbb{C}^d$, if an donly if $\lambda_i h \in S_\Psi$ for all eigenvalues λ_i of $\mathbf{M}$.

Com. So the region of stability is always a bounded region in the complex plane.

15.3 Stiff IVPs

Now let's consider the case where we have non-linear ODEs. So, let

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$$

be a non-linear ODE with initial value $\mathbf{y}_0$.

Now let $0 < t \ll 1$, then $\mathbf{y}(t) \approx \mathbf{y}(0) = \mathbf{y}_0$ approximately. So we can linearize $\mathbf{y}$ around $\mathbf{y}_0$.

$$\dot{\mathbf{y}} \approx \mathbf{f}(\mathbf{y}_0) + D\mathbf{f}(\mathbf{y}_0)(\mathbf{y} - \mathbf{y}_0). \qquad \text{(Linearization)}$$

Now if we replace $\approx$ by $=$ then we get an *(affine)* linear ODE. So we replace the Jacobian by a matrix $\mathbf{M} := D\mathbf{f}(\mathbf{y}_0)$, and so we get a linear ODE with some constant term

$$\dot{\mathbf{y}} = \mathbf{M}\mathbf{y} \overbrace{-\mathbf{M}\mathbf{y}_0 + \mathbf{f}(\mathbf{y}_0)}^{+\mathbf{b}\text{ (const)}} = \mathbf{M}\mathbf{y} + \mathbf{b}$$

So for small times $t \mapsto \mathbf{y}(t)$ behaves like the solution of an affine linear ODE.

Now it turns out that this linearization can also be done for RK-methods.