Contents

General Relative condition nuumber for $\mathbf{f} : \mathbb{R}^n -> \mathbb{R}^m$ if the peturbed input $\tilde{\mathbf{x}}$ approaches the non-perturbed

$$\frac{||\mathbf{f}(\mathbf{x}) - \mathbf{f}(\tilde{\mathbf{x}})||}{||\mathbf{f}(\mathbf{x})||} \leq \kappa_{rel}(\mathbf{f})(\mathbf{x}) \cdot \frac{||\mathbf{x} - \tilde{\mathbf{x}}||}{||\mathbf{x}||} \tag{1}$$

Additionally, one has to outfit both multidimensional quantitities with a corresponding norm (idealls, it is the same type of norm for both). If the function is continously differentiable we get the first order approximation to

$$\kappa_{rel}(\mathbf{f})(\mathbf{x}) \doteq \frac{||\mathbf{x}||}{||\mathbf{f}(\mathbf{x})||} ||\mathbf{f}'(\mathbf{x})|| \tag{2}$$

Where the derivative of the function is its Jacobian. Therefore, one has to define an appropriate matrix norm.

Name	Symbol	Condition number
Addition / Subtraction	$x + a$	$\left\vert \frac{x}{x+a} \right\vert$
Scalar Multiplication	ax	1
Division	$1/x$	1
Polynomial	x^n	$\vert n \vert$
Exponential function	e^x	$\vert x \vert$
Natural logarithm function	$\ln(x)$	$\left\vert \frac{1}{\ln(x)} \right\vert$
Sine function	$\sin(x)$	$\vert x \cot(x) \vert$
Cosine function	$\cos(x)$	$\vert x \tan(x) \vert$
Tangent function	$\tan(x)$	$\vert x(\tan(x) + \cot(x)) \vert$
Inverse sine function	$\arcsin(x)$	$\frac{x}{\sqrt{1-x^2}\arcsin(x)}$
Inverse cosine function	$\arccos(x)$	$\frac{\vert x \vert}{\sqrt{1-x^2}\arccos(x)}$
Inverse tangent function	$\arctan(x)$	$\frac{x}{(1+x^2)\arctan(x)}$

Relative condition number of a multi-variant problem with one-dimensional output

$$\kappa_{rel}(f, \vec{x}) = \frac{\sum_{i=1}^{n} |f_{x_i}(\vec{x})| \cdot |x_i|}{|f(\vec{x})|} \tag{3}$$

Condition number of a matrix (e.g. choose the infinity matrix norm - row sum norm)

$$\kappa_{rel}(A) = ||A|| \cdot ||A^{-1}|| \tag{4}$$

Rule of thumb for the loss of digits of accuracy

$$\text{loss} = log_{10}(\kappa_{rel}) \tag{5}$$

One-dimensional Taylor-Series Expansion for a n+1 continously differentiable function on the the interval a to b with a $\xi \in [x, x+h]$

$$f(x+h) = \sum_{k}^{n} \frac{1}{k!} f^{(k)}(x) h^k + \frac{1}{(n+1)!} f^{(n+1)}(\xi) h^{n+1} \tag{6}$$

1 Matrix Factorization

1.1 QR Decompostion with Householder transformations

Householder transformations are unitary and hermitian.

$$\mathbf{T}^{-1} = \mathbf{T}^H = \mathbf{T} = I_n - 2\frac{\vec{u}\vec{u}^H}{\vec{u}^H\vec{u}} \in \mathbb{C}^{nxn} \tag{7}$$

1.2 Cost of Decompositions

- **LU** $\propto 2\frac{n^3}{3}$
- Householder **QR** $\propto 4\frac{n^3}{3}$
- Givens **QR** $\propto 6\frac{n^3}{3}$

2 Polynomial Interpolation

2.1 Aitken-Neville Algorithm

Polynomial interpolation with Aitken-Neville of a set of $n+1$ data points to interpolate a polynomial of degree n in the form $P(x) = p_{0,n}(x)$. Start by setting the initial values in the recursive scheme to $p_{i,i}(x) = y_i$. Then calculate the other values by:

$$p_{i,j}(x) = \frac{(x - x_j)p_{i,j-1}(x) - (x - x_i)p_{i+1,j}(x)}{x_i - x_j} \tag{8}$$

More efficient evaluation of the Aitken-Polynomial at one certain point

$$seematlab \tag{9}$$

Aitken's theorem states that if two polynomial functions $M(t)$ and $N(t)$ interpolate $n-2$ common points, and $M(t)$ is additionally interpolating a point before all the others and $N(t)$ is addtionally interpolating a point at the end of the interval, then there is a unique polynomial $P(t)$ that interpolates all value pairs as follows

$$P(t) = N(t) + \frac{t - t_k}{t_k - t_i}(N(t) - M(t)) \tag{10}$$

2.2 Newton Algorithm

Newton's Method provides a nice evalutation of a polynomial interpolation at many points since it easily gives you the functional form of the interpolating polynomial. Define the divided differences

$$f[t_i, t_{i+1}, ..., t_k] = \frac{f[t_{i+1}, ..., t_k] - f[t_i, ..., t_{k-1}]}{t_k - t_i}, \quad f[t_i] := f(t_i) = y_i \tag{11}$$

Then the polynomial is defined as

$$P(t) = \sum_{k=0}^{n} f[t_0, ..., t_k] \prod_{i=0}^{k-1}(t - t_i) \tag{12}$$

Using Horner's scheme, the polynomial can be evaluated more efficiently

```
P_n := f[t_0, t_1, ..., t_n]
k=n-1:-1:0:
    P_k := P_{k+1} * (t - t_k) + f[t_0, t_1, ..., t_k]
```

The Chebychev polynomials are directly defined by

$$T_n(x) = \cos(n \arccos(x)) \tag{13}$$

Or by a three term recursion (it shares the first two polynomials with the common monomial basis)

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad T_0(x) = 1, \quad T_1(x) = x \tag{14}$$

Chebychev nodes on an interval from (a, b). n nodes with index k. They are the roots of the Chebychev polynomial of degree n.

$$x_k = \frac{1}{2}(a + b) + \frac{1}{2}(b - a)\cos\left(\frac{2k - 1}{2n}\pi\right) \tag{15}$$

Hermite Polynomials

For efficient evaluation of cubic splines, use a different basis instead of the standard monomial one (where we would have to solve for all 4 coefficients on each sub-interval) - This basis has no special name

$$\Pi_3 = \text{span}\left\{1-\xi, \xi, -\frac{\xi}{3}+\frac{\xi^2}{2}-\frac{\xi^3}{6}, -\frac{\xi}{6}+\frac{\xi^3}{6}\right\}, \quad \xi = \frac{t-t_i}{t_{i+1}-t_i} \tag{16}$$

Then the interpolating polynomial is given by

$$s_3(t) = y_i(1-\xi)+y_{i+1}\xi+h_i^2 y_i^{''}\left(-\frac{\xi}{3}+\frac{\xi^2}{2}-\frac{\xi^3}{6}\right)+h_i^2 y_{i+1}^{''}\left(-\frac{\xi}{6}+\frac{\xi^3}{6}\right) \tag{17}$$

This is beneficial since y_i is already available due to the inerpolating condition. The second derivatives at each node can be efficiently computed. Introduce the following abbreviations ($\mu_i + \lambda_i = 1$)

$$\mu_i := \frac{h_{i-1}}{h_{i-1}+h_i} = \frac{t_i - t_{i-i}}{t_{i+1}-t_{i-1}} > 0 \tag{18}$$

$$\lambda_i := \frac{h_i}{h_{i-1}+h_i} = \frac{t_{i+1} - t_i}{t_{i+1}-t_{i-1}} > 0 \tag{19}$$

Then the following triagonal (non-square since we have 2 DoF) system has to be solved

$$\mu_i y_{i-1}^{''} + 2y_i^{''} + \lambda_i y_{i+1}^{''} = 6y_{i-1,i,i+1} \tag{20}$$

This creates $(n-2)xn$ system for the $n-2$ interior nodes. It uses the second devided difference (refer to Newton's interpolation scheme) defined by

$$y_{i-1,i,i+1} = \frac{y_{i,i+1} - y_{i-1,i}}{h_{i-1}+h_i} \tag{21}$$

With the regular divided differenc given by

$$y_{i,i+1} = \frac{y_{i+1}-y_i}{h_i} \tag{22}$$

This system is underconstrained. We have to prescribe boundary conditions. These can be of following type

- Hermite splines $s_3^{'}(t_0) = y_0^{'}$ and $s_3^{'}(t_n) = y_n^{'}$ are prescribed
- $s_3^{''}(t_0) = y_0^{''}$ and $s_3^{''}(t_n) = y_n^{''}$ are prescribed. "Natural Spline" if both are set to zero.

- Periodic Splines $s_3^{'}(t_0) = s_3^{'}(t_n)$ and $s_3^{''}(t_0) = s_3^{''}(t_n)$ are prescribed

By the help of the Gershgorin Disks we can proof that under a given bounary condition the cubic spline is unique.

The erros for the different derivatives are given

...

By this we can conclude that that the splines converge against the function they are interpolating for increasingly finer subintervals. For Polynomials this is not given (only on Chebychev nodes).

Gershgorin disks give an estimate to where eigenvalues of a matrix lie in the complex plain. Assume $A \in \mathbb{C}^{nxn}$

$$R_i := \left\{ z \in \mathbb{C} || z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\} \tag{23}$$

The the set of z (=R) describe the Gershgorin disks in which the eigenvalue can be found

---

The Legendre Polynomials form a basis of the space of polynomials and are directly defined by

$$P_n(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2 - 1)^n] \tag{24}$$

The recursive definition is given by

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \quad P_0 = 1, \quad P_1 = x \tag{25}$$

Eigenvalues should numerically never be determined by the roots of the characteristic polynomial. Instead use other algorithms like (inverse) power iteration or the QR algorithm. Then the condition number of finding an eigenvalue is given as

$$\kappa(\lambda) = ||\mathbf{V}|| \cdot ||\mathbf{V}^{-1}|| \tag{26}$$

With the eigenvector matrix $\mathbf{V}$. If the matrix $\mathbf{A}$ is normal, i.e. $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$, then $\mathbf{V}$ is unitary and the problem is perfectly conditioned.

3 Quadrature - Numerical Integration

3.1 Newton-Cotes-Methods

Newton-Cotes type quadrature interpolates the integrand by the help of polynomials on an equidistant grid. Composite rules define their sub-interval length to $\frac{b-a}{n}$ with n sub-intervals.

name	definition	remainder
Trapezoid	$\frac{b-a}{2}(f(a)+f(b))$	$-\frac{f''(\eta)}{12}(b-a)^3$
Kepler	$\frac{b-a}{6}\left(f(a)+4f(\frac{a+b}{2})+f(b)\right)$	$-\frac{f^{(4)}(\eta)}{90}\left(\frac{b-a}{2}\right)^5$
Comp. Trapez.	$\frac{h}{2}\left(f(a)+2\sum_{j=1}^{n-1}f(x_j)+f(b)\right)$	$-\frac{b-a}{12}h^2f''(\eta)$
Simpson	$\frac{h}{3}\sum_{j=1}^{n/2}[f(x_{2j-2})+4f(x_{2j-1})+f(x_{2j})]$	$-\frac{h^4}{180}(b-a)f^{(4)}(\eta)$

3.2 Adaptive Composite Rules

- Start with course sub-intervals h_1
- Half the sub-interval-lengths everywhere $h_2 = h_1/2$
- For each interval calculate the error $err_j = |S_{\text{coarse}} - S_{\text{fine}}|$
- Weight each error with the sub-interval length $err_j \frac{h_j}{b-a}$
- Refine the sub-intervals further (recursively) if error is above a set tolerance