Computational Astrophysics I: Introduction and basic concepts

Helge Todt

Astrophysics
Institute of Physics and Astronomy
University of Potsdam

SoSe 2024, 17.6.2024

Root finding - Iterative techniques

Transcendent equations I

Problem: Finding roots for equations that cannot be solved analytically, i.e. finding x_0 for $f(x_0) = 0$

Transcendent equation: quantum states in a square well

The 1d potential $V(x)$ for the Schrödinger equation

$$V(x) = \begin{cases} -V_0 & , |x| \leq a \\ 0 & , |x| \geq a \end{cases} \tag{1}$$

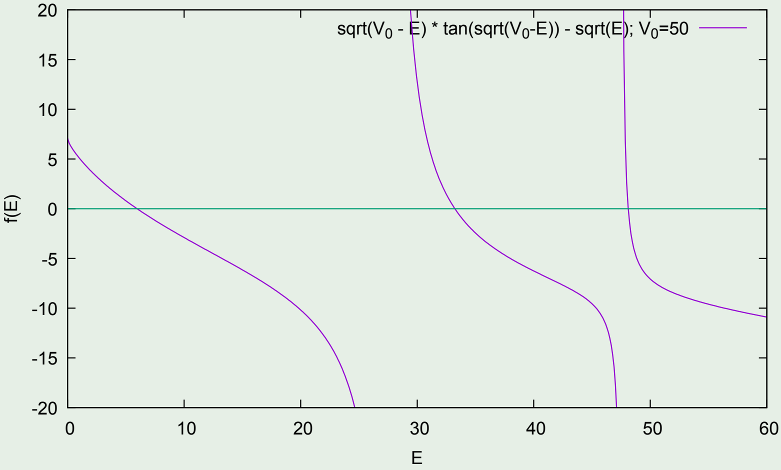
has bound states with energies $E = -E_\mathrm{B} < 0$

$$\sqrt{2m(V_0 - E_\mathrm{B})} \tan\left[a\sqrt{2m(V_0 - E_\mathrm{B})}\right] = \sqrt{2mE_\mathrm{B}} \tag{2}$$

$\rightarrow$ e.g., for $2m = 1$, $a = 1$ we want to find the roots E_B of

$$f(E_\mathrm{B}) = \sqrt{V_0 - E_\mathrm{B}} \tan\left(\sqrt{V_0 - E_\mathrm{B}}\right) - \sqrt{E_\mathrm{B}} \stackrel{!}{=} 0 \tag{3}$$

Hints: Transcendent equation: quantum states in a square well



Note: the function $f(E_\mathrm{B})$

- is not continuous
- has multiple roots

Transcendent equations III

Roots of numerically derived functions

Some functions cannot even be written analytically, e.g.

- $x(t)$ for the Kepler problem
- solutions of the Lane-Emden equation $\theta_n(\xi)$ for $n \neq \{0, 1, 5\}$

$\rightarrow$ roots can be found numerically by trial-and-error algorithms, i.e. iteratively until some specified level of precision is reached

Bisection I

$\rightarrow$ very stable (root is always found if conditions fulfilled), but also very slow iterative procedure
$\rightarrow$ needs *two* start values $[x_1, x_2]$ for estimating x_0
If $f(x)$ continuous on $[a, b]$ and $f(a) \cdot f(b) < 0$, then the intermediate value theorem guarantees the existence of an $x_0 \in [a, b]$ with $f(x_0) = 0$.

Bisection algorithm

1. start with interval $[x_1, x_2]$ on which $f(x)$ *changes sign* (so $f(x_1) \cdot f(x_2) < 0$) $\rightarrow$ contains root
2. choose new x_3 as the midpoint of the interval $x_3 = \frac{x_1+x_2}{2}$
3. calculate $f(x_3)$: either $f(x_3)$ is sufficiently close to 0 $\rightarrow$ root is x_3
   or x_3 is a new interval endpoint:
   if $f(x_3) \cdot f(x_1) > 0 \rightarrow$ new interval is $[x_3, x_2]$
   or if $f(x_3) \cdot f(x_1) \leq 0 \rightarrow$ new interval is $[x_1, x_3]$
4. goto step 2

$\rightarrow$ nested intervals enclosing the root
$\rightarrow$ as interval is halved every step, gain ≈ 1 digit each 3 steps (2^3)

$\rightarrow$ similar to Newton's method (see below), actually approximation with finite differences

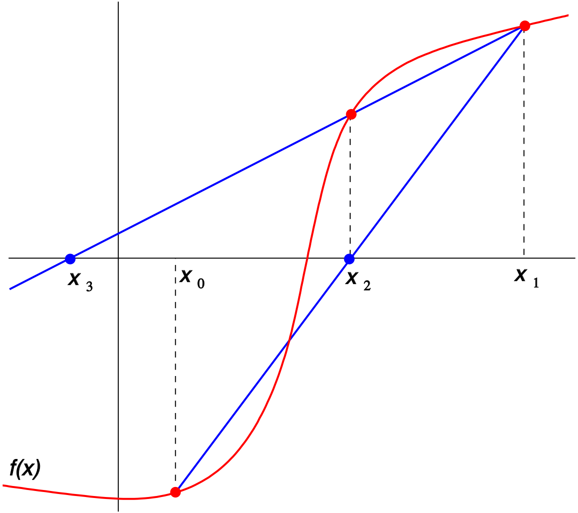
Requirement: $f(x)$ continuous and $\exists x_0 \in [a, b]$ with $f(x_0) = 0$.
Then: line trough $(x_0, f(x_0))$ and $(x_1, f(x_1))$, so that

$$y = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1) + f(x_1)$$

with root

$$x = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

$\rightarrow$ new point $(x_2, f(x_2))$ repeat with x_1, x_2 instead of x_0, x_1



The secant method II

Secant method

1. start with interval $x_1 \neq x_2$ close to the root
2. iterate

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{4}$$

$\rightarrow$ superlinear convergence, per iteration about 1.6 more correct digits
$\rightarrow$ convergence not assured (especially as $f(x_n) \cdot f(x_{n-1})$ is not necessary 0)
$\rightarrow$ numerically limited by subtractive cancelation, as fraction $\rightarrow$ 0/0

Regula falsi method I

$\rightarrow$ refinement of bisection by combining it with the secant method

Regula falsi (False position method)

1. as for bisection: start with interval $[x_1, x_2]$ with $f(x_1) \cdot f(x_2) < 0$
2. calculate the zero of the secant

$$x_3 = x_1 - \frac{x_2 - x_1}{f(x_2) - f(x_1)} f(x_1) = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} \tag{5}$$

3. if $f(x_3) = 0 \rightarrow$ stop, else
4. if $f(x_1) \cdot f(x_3) > 0 \rightarrow$ replace $x_1 = x_3$
   if $f(x_2) \cdot f(x_3) > 0 \rightarrow$ replace $x_2 = x_3$
5. goto 2

$\rightarrow$ superlinear convergence (more than one significant digit per iteration) as for secant method

$\rightarrow$ advantage: numerically stable, no evaluation of derivatives required, computation of function values is reused

$\rightarrow$ preferred method for 1d problems or *Newton-Raphson method* (Newton 1669, Raphson 1690) to solve numerically non-linear equations or systems of equations

$\rightarrow$ quicker than bisection, but sometimes problematic

Idea: start with approximation x_0, draw tangent at $(x_0, f(x_0))$, determine intersection with x-axis $\rightarrow$ new approximation for root

Derivation: evaluate function $f(x)$ around x_0 (Taylor expansion)

$$f(x_0 + \Delta x) \simeq f(x_0) + f'(x_0) \cdot \Delta x \tag{6}$$

$$\text{(linear approximation = tangent } t \text{ on } x_0 \text{ shall vanish)} \tag{7}$$

$$\rightarrow\; t(x) = f(x_0) + f'(x_0) \cdot \Delta x \stackrel{!}{=} 0 \tag{8}$$

$$\rightarrow\; \Delta x = -\frac{f(x_0)}{f'(x_0)} \tag{9}$$

the correction Δx added on x_0 gives improved guess for root

Newton's method II

Newton's method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{10}$$

Convergence:

If $f : [a, b] \to \mathbb{R}$ is a $\mathcal{C}^2$ function with

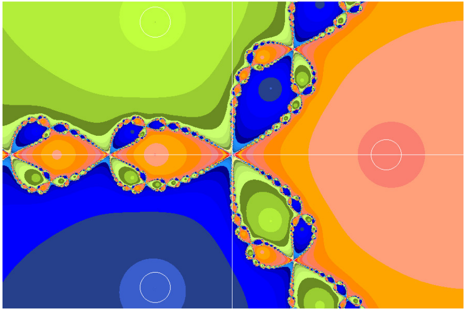
1. f has a root ξ in $[a, b]$
2. $f'(x) \neq 0$ for $x \in [a, b]$
3. f is *either* convex $(f'' \geq 0)$ *or* concave $(f'' \leq 0)$ in $[a, b]$
4. the iterated x_1 for $x_0 = a$ and $x_0 = b$ are in $[a, b]$

Then: For any $x_0 \in [a, b]$ the values $x_1, x_2, \ldots$ from Eq. (10) are in $[a, b]$ and the sequence converges monotonically to ξ.

Remarks:

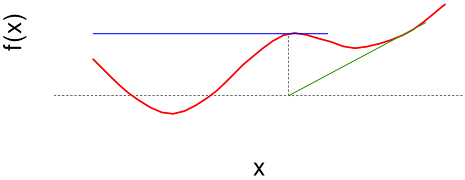
- only locally convergent,

  i.e. result depends on start approximation for x_0
  $\rightarrow$ Newton fractal for $z^3 - 1 = 0$



- in some situations Newton's method may fail (see requirements):
  - if x_n is at local extremum

  with $f(x_n) \neq 0 \rightarrow$ tangent with slope 0,
  i.e. $f'(x_n) = 0 \rightarrow$ infinite correction
  $\rightarrow$ solution: start over with different x_0

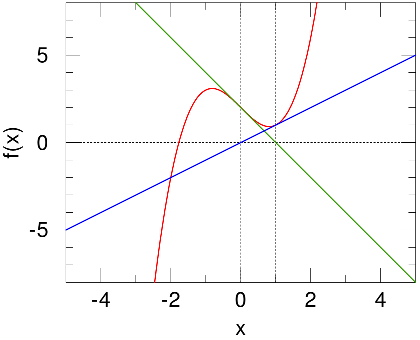


Newton's method IV

- infinite loop,

e.g., $f(x) = x^3 - 2x + 2$
with $x_0 = 0 \to f(0) = 2,\ f'(0) = -2$
$\to x_1 = 0 - \frac{2}{-2} = 1$ and
for $x_0 = 1 \to f(1) = 1,\ f'(1) = 1$
$\to x_1 = 1 - \frac{1}{1} = 0$

$\to$ happens if x_0 in region where $f(x)$ not "linear enough" (vizualization may help to find better initial guess)



- convergence is quadratic, i.e. with every step two more significant digits
- instead of analytic $f'(x)$ numeric approximation $f'(x_n) \simeq \frac{f(x_n+h)-f(x_n)}{h}$ sufficient
  $\to$ even rough (or constant!) approximation may be sufficient
- if convergent, method is stable

Backtracing

→ solution to some problems (i.e. infinite loop) with large corrections

So: if for new guess $x_0 + \Delta x$

$$|f(x_0 + \Delta x)|^2 > |f(x_0)|^2 \tag{11}$$

→ backtrack, try smaller guess, e.g., $x_0 + \Delta x/2$, if still condition (11), try $x_0 + \Delta x/4$ and so on

→ because tangent line will lead to *decrease* in $f(x)$, even small step Δx sufficient

Extension to multidimensional case

for multidimensional function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + J(\mathbf{x}) \cdot \mathbf{h} + \mathcal{O}(||\mathbf{h}||^2) \tag{12}$$

where $\mathbf{J}(\mathbf{x}) = \mathbf{f}'(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x})$ the Jacobi matrix, the matrix of the partial derivatives w.r.t. x:

$$J(\mathbf{x}) := \left(\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right)_{ij} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \dots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \tag{13}$$

Therefore

$$\mathbf{x}_{n+1} = \mathbf{x}_n - J^{-1}(\mathbf{x})\, f(\mathbf{x}_\mathrm{n}) \rightarrow\ \Delta \mathbf{x}_n = -J^{-1}(\mathbf{x}_n) f(\mathbf{x}_n) \tag{14}$$

As inversion of J is expensive, usually solve system of linear equations:

$$J(\mathbf{x}_n)\Delta\mathbf{x}_n = -f(\mathbf{x}_n) \tag{15}$$

to get $\Delta\mathbf{x}_n$ and then $\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_n$

$\rightarrow$ Newton-Raphson method in n dimensions (i.e. system of equations) is expensive, therefore often used: *quasi Newton methods*

Example: statistical equilibrium

In the non-LTE case population numbers of ions n from statistical equations with transition rates P_{ij}, stationary: $\sum_{i\neq j} n_i \, P_{ij} = \sum_{j\neq i} n_j \, P_{ji}$ with $P_{ij} := -\sum_i P_{ji} \rightarrow \mathrm{n} \cdot \mathrm{P}(\mathrm{n}, J, T_\mathrm{e}) = 0$, matrix has block structure (but coupling extra line from charge conservation / electron density):

$$\mathrm{P} = \left[\begin{array}{c|c|c} \mathrm{H} & & \\ \hline & \mathrm{He} & \\ \hline & & \mathrm{N} \end{array}\right] \tag{16}$$

together with $J = \Lambda S(\mathrm{n})$. When using net-radiative brackets or accelerated Λ iteration:
$\rightarrow$ non-linear system of N equations $\rightarrow N^3$ derivatives (N derivatives for $N \times N$ rates)

Instead of calculating n^3 derivatives use modified *secant* equation

$$x_{k+1} = x_k - f(x_k) B_k^{-1} \tag{17}$$

$$\text{with "slope"} \quad B_{k+1} = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} = \frac{\Delta y_k}{\Delta x_k} \qquad \rightarrow \Delta y_k = B_{k+1} \Delta x_k \tag{18}$$

But: Eq. (18) defines B only as $n-1$ dimensional subspace $\rightarrow$ need further constraints.
Broyden (1965): use updating algorithm

$$B_{k+1} = B_k + \frac{\Delta x_k^T \otimes (\Delta y_k - \Delta x_k B_k)}{|\Delta x_k|^2} \tag{19}$$

with dyadic product of two vectors (columns $\times$ rows) yielding matrix elements:
$(u^T \otimes v)_{ij} = u_i v_j$
Advantage: Broyden's formula (19) can be inverted analytically by help of

Sherman-Morrison-Woodbury lemma

$$(A + u^{\mathsf{T}} \otimes v)^{-1} = A^{-1} - \frac{A^{-1} u^{\mathsf{T}} \otimes v A^{-1}}{1 + v\, A^{-1} u^{\mathsf{T}}} \tag{20}$$

with row-vectors u, v and an invertible matrix A the required B_{k+1}^{-1} can be directly obtained from previous B_k^{-1}:

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(B_k^{-1} \Delta x_k^{\mathsf{T}}) \otimes (\Delta x_k - \Delta y_k B_k^{-1})}{(\Delta y_k B_k^{-1}) \cdot \Delta x_k^{\mathsf{T}}} \tag{21}$$

$\rightarrow$ no operations between full matrices involved $\rightarrow$ only $\sim N^2$ multiplications

Broyden method

1. select starting point x_0 (e.g., initial guess on n from LTE population numbers) and starting matrix $B_0^{-1} = (f')^{-1}$ (Newton step)
2. $x_{k+1} = x_k - f(x_k) B_k^{-1}$
3. stop if $|\Delta x| < \epsilon$
4. else update Broyden matrix Eq. (21)

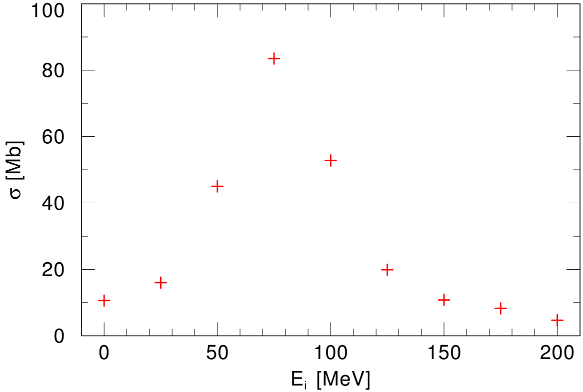
$$B_{k+1}^{-1} = B_k^{-1} + \frac{(B_k^{-1} \Delta x_k^{\mathrm{T}}) \otimes (\Delta x_k - \Delta y_k B_k^{-1})}{(\Delta y_k B_k^{-1}) \cdot \Delta x_k^{\mathrm{T}}}$$

5. $k = k + 1$ goto 2

Interpolation

Consider following measurement of a cross section

E_i [MeV]	0	25	50	75	100	125	150	175	200
$\sigma(E_i)$ [Mb]	10.6	16.0	45.0	83.5	52.8	19.9	10.8	8.25	4.7



The cross section can be described by Breit-Wigner formula

$$f(E) = \frac{f_r}{(E - E_r)^2 + \Gamma^2/4} \qquad (22)$$

Interpolating data II

Interpolation problem

Task: Determine $\sigma(E)$ for values of E which lie between measured values of E

By, e.g.,

- numerical interpolation (assumption of data representation by polynomial in E):
  - piecewise constant $\rightarrow$ step function (easy to implement, error goes as $\sim y_i'(x_{i+1} - x_i)$)
  - piecewise linear (special case of polynomial)
  - polynomial (Lagrange)
  - piecewise Lagrange, cubic spline

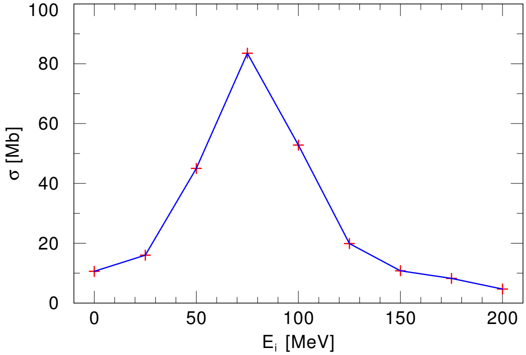
  $\rightarrow$ ignores errors in measurement (noise)
- fitting parameters of an underlying model, e.g., Breit-Wigner with f_r, E_r, Γ, (taking errors into account), i.e., minimizing χ^2
- Fourier analysis

Linear interpolation

tabulated function $y_i = y(x_i), i = 1 \dots N$, e.g., for interval x_i, x_{i+1}, linear interpolation in this interval is by

$$y = A(x)y_i + B(x)y_{i+1} \tag{23}$$

$$A \equiv \frac{x_{i+1} - x}{x_{i+1} - x_i} \qquad B \equiv 1 - A = \frac{x - x_i}{x_{i+1} - x_i} \tag{24}$$



or:

$$y = y_i + (y_{i+1} - y_i)\frac{x - x_i}{x_{i+1} - x_i}$$

disadvantages:

- not differentiable at nodes x_i
- error $\sim y_i''(x_{i+1} - x_i)^2$

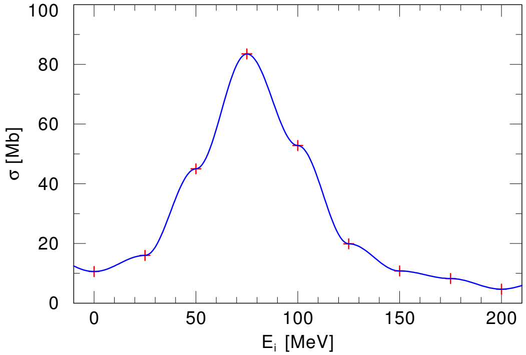
Cosine interpolation

a smoother transition between intervals can be achieved by piecewise cosine functions:

$$t = \frac{x - x_i}{x_{i+1} - x_i} \quad \text{(mapping on unit interval } [0, 1]) \tag{25}$$

$$B = (y_{i+1} + y_i)/2 \ ; \qquad A = y_i - B \tag{26}$$

$$y = A \cos(\pi\, t) + B \tag{27}$$



note, that at the nodes x_i because of $\cos'(0) = 0 = \cos'(\pi) \to y_i' = 0$

Lagrange interpolation (global)

- fit $(n-1)$th degree polynomial through n data points

$$p(x) = y_1\lambda_1(x) + y_2\lambda_2(x) + \ldots + y_n\lambda_n(x) \tag{28}$$

$$\lambda_i(x) = \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_1}{x_i - x_1}\frac{x - x_2}{x_i - x_2}\cdots\frac{x - x_n}{x_i - x_n} \tag{29}$$

  where $\sum_{i=1}^{n} \lambda_i(x) = 1$
- practical: the λ_i are independent from the values of the function values $f_i \rightarrow$ for same nodes $x_i \rightarrow$ same λ_is (e.g., when measuring different y_is for same x_is)
- so, for $n = 9 \rightarrow (n-1) = 8$th degree polynomial
- note that $\lambda_i(x_j) = \delta_{ij}$

Example: Lagrange interpolation polynomial $n = 3$

$n = 3$ data points $\to n - 1 = 2$ degree polynomial, e.g., for points $P_1 = (-1; 4), P_2 = (0; 1), P_3 = (2; 5)$ $(x_1 = -1;\ x_2 = 0;\ x_3 = 2)$

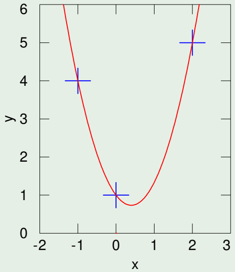
$$\lambda_1 = \frac{x - x_2}{x_1 - x_2} \cdot \frac{x - x_3}{x_1 - x_3} = \frac{(x-0)}{(-1-0)} \cdot \frac{(x-2)}{(-1-2)} = \frac{x^2 - 2x}{3} \tag{30}$$

$$\lambda_2 = \frac{x - x_1}{x_2 - x_1} \cdot \frac{x - x_3}{x_2 - x_3} = \frac{(x-(-1))}{(0-(-1))} \cdot \frac{(x-2)}{(0-2)} = \frac{x^2 - 2 - x}{-2} \tag{31}$$

$$\lambda_3 = \frac{x - x_1}{x_3 - x_1} \cdot \frac{x - x_2}{x_3 - x_2} = \frac{(x-(-1))}{(-2-(-1))} \cdot \frac{(x-0)}{(2-0)} = \frac{x^2 + x}{6} \tag{32}$$

$$p(x) = y_1 \cdot \lambda_1 + y_2 \cdot \lambda_2 + y_3 \cdot \lambda_3 = 4 \cdot \frac{x^2 - 2x}{3} + 1 \cdot \frac{x^2 - 2 - x}{-2} + 5 \cdot \frac{x^2 + x}{6} \tag{33}$$

$$= \frac{5}{3}x^2 - \frac{4}{3}x + 1 \tag{34}$$



Check: $\lambda_1 + \lambda_2 + \lambda_3 = \frac{x^2 - 2x}{3} + \frac{x^2 - 2 - x}{-2} + \frac{x^2 + x}{6} = 1$

Application: Newton-Cotes formulae for integration

Idea: interpolate $f(x)$ in $\int_a^b f(x)dx$ with polynomial of degree n and integrate this polynomial exactely (note: now n =degree, start with $j = 0$):

$$\int_a^b f(x)dx \approx \int_a^b p_n(x)dx = \int_a^b \sum_{i=0}^{n} f(x_i) \cdot \lambda_i(x) \tag{35}$$

$$\lambda_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \quad \xrightarrow{x=a+ht} \quad \phi_i(t) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - j}{i - j} \tag{36}$$

Note that the transformation $x = a + ht$ means that $x_0 = a + h \cdot 0$, $x_1 = a + h \cdot 1, \ldots$
(equidistant subintervals h on x-axis)
Therefore the integration of $p_n(x)$ yields

$$\int_a^b \sum_{i=0}^{n} f(x_i) \cdot \lambda_i(x) = h \sum_{i=0}^{n} f_i \int_0^n \phi_i(t)dt = h \sum_{i=0}^{n} f_i w_i \tag{37}$$

Example: Newton-Cotes formula $n = 1$

$$w_0 = \int_0^n \phi_0(t)dt = \int_0^1 \frac{t-1}{0-1}dt = \int_0^1 (1-t)dt = \frac{1}{2} \tag{38}$$

$$w_1 = \int_0^n \phi_1(t)dt = \int_0^1 \frac{t-0}{1-0}dt = \int_0^1 t\,dt = \frac{1}{2} \tag{39}$$

$$\int_a^b p_1(x)dx = h\sum_{i=0}^{1} f_i w_i = h\,(f_0\frac{1}{2} + f_1\frac{1}{2}) = \frac{h}{2}(f_0 + f_1) \tag{40}$$

$\rightarrow$ trapezoid rule

Analogously for $n = 2$, e.g.,

$$w_0 = \int_0^2 \frac{t-1}{0-1} \cdot \frac{t-2}{0-2}dt = \frac{1}{2}\int_0^2 (t^2 - 3t + 2)dt = \frac{1}{3} \tag{41}$$

and $w_1 = \frac{4}{3}$, $w_2 = \frac{1}{3} \rightarrow \int_a^b p_2(x)dx = \frac{h}{3}(f_0 + 4f_1 + f_2) \rightarrow$ Simpson's rule

Interpolating data IX

$\rightarrow$ closed Newton-Cotes formulae with nodes t_i on $[0,1]$: $t_0 = 0, t_i = \frac{i}{n}, t_n = 1$, use mapping $x_i = a + t_i(b-a)$, so

$$\int_a^b f(x)dx = \int_a^b p_n(x)dx + E_f = (b-a)\sum_{i=0}^{n} w_i f(x_i) + E_f \tag{42}$$

n	name	nodes t_i	weights w_i	E_f
1	trapezoid rule	$0 \; 1$	$\frac{1}{2} \; \frac{1}{2}$	$-\frac{(b-a)^3}{12} f''$
2	Simpson's rule	$0 \; \frac{1}{2} \; 1$	$\frac{1}{6} \; \frac{4}{6} \; \frac{1}{6}$	$-\frac{(b-a)^5}{2880} f^{(4)}$
3	3/8 rule	$0 \; \frac{1}{3} \; \frac{2}{3} \; 1$	$\frac{1}{8} \; \frac{3}{8} \; \frac{3}{8} \; \frac{1}{8}$	$-\frac{(b-a)^5}{6480} f^{(4)}$
4	Milne rule	$0 \; \frac{1}{4} \; \frac{2}{4} \; \frac{3}{4} \; 1$	$\frac{7}{90} \; \frac{32}{90} \; \frac{12}{90} \; \frac{32}{90} \; \frac{7}{90}$	$-\frac{(b-a)^7}{1935360} f^{(6)}$
	...			

for $n \geq 8$ some weights w_i are also negative $\rightarrow$ subtractive cancellation $\rightarrow$ useless

Note, again: $\sum w_i = 1$. The error: $E_f = h^{p+1} \cdot K \cdot f^{(p)}(\xi)$, $\xi \in (a,b)$

Trick: Neville's algorithm (sometimes confused with Aitken's method)

Instead of computing the whole Lagrange polynomial: nested linear interpolations

Where the $f_{i\ldots j}$ are recursively computed, e.g.,

$$\begin{array}{cccc}
x_1 & f_1 & & \\
 & & f_{12} & \\
x_2 & f_2 & & f_{123} \\
 & & f_{23} & \\
x_3 & f_3 & &
\end{array}$$

$$f_{i\ldots j} = \frac{x - x_j}{x_i - x_j} f_{i\ldots j-1} + \frac{x - x_i}{x_j - x_i} f_{i+1\ldots j} \tag{43}$$

$$f_{123} = \frac{x - x_3}{x_1 - x_3} f_{12} + \frac{x - x_1}{x_3 - x_1} f_{23} \tag{44}$$

→ sequence of ... linear interpolations = interpolation with polynomial of $n - 1$ degree

→ error can be estimated from $\frac{|f_{i\ldots j} - f_{i..j-1}| + |f_{i\ldots j} - f_{i+1..j}|}{2}$,

e.g, $\frac{|f_{12345} - f_{1234}| + |f_{12345} - f_{2345}|}{2}$

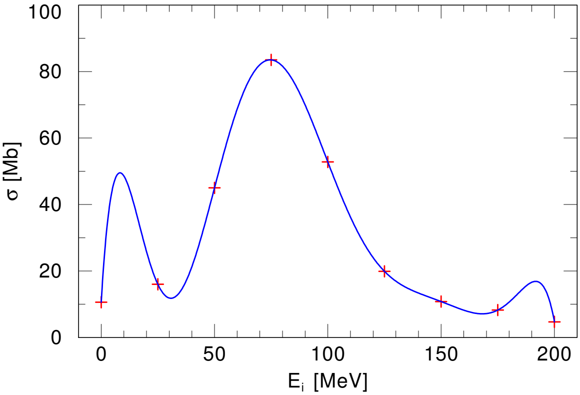
Neville's algorithm: code

```
// input : given points xi[], fi[], value of x for interpolation
// output: f at position x, error estimate df

for (i = 1 ; i <= n ; ++i) ft[i] = fi[i] ;

for (i = 1 ; i <= n-1 ; ++i) {
  for (j = 1 ; j <= n-i ; ++j) {
    x1 = xi[j] ; x2 = xi[j+1] ;
    f1 = ft[j] ; f2 = ft[j+1] ;
    ft[j] = (x - x1)/(x2 - x1) * f2 + (x - x2)/(x1 - x2) * f1
  }
}
f = ft[1] ;
df = (fabs(f - f1) + fabs(f - f2))/2. ;
```

Interpolating data XII

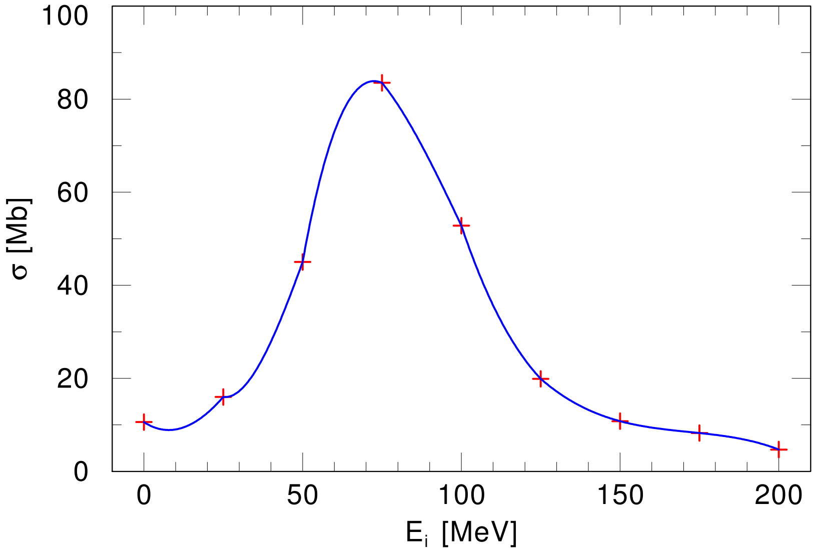
Runge's phenomenon:
polynomials $\rightarrow \pm\infty$ for $x \rightarrow \pm\infty$.

If function has different behavior (e.g., asymptotically constant) $\Rightarrow$ oscillations at intervall limits (e.g., for Runge's function $\frac{1}{1+x^2}$)



Interpolating data XIII

one possible solution for the problem of Runge's phenomenon: piecewise polynomials
here: 2nd degree polynomials (parabola, requires 3 points)



Problem:
not differentiable at x_i

Interpolating data XIV

better: Cubic Hermite spline

- remember: piecewise linear interpolation with functions

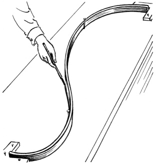
$$A(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} \qquad B(x) = 1 - A = \frac{x - x_i}{x_{i+1} - x_i} \tag{45}$$

$$\rightarrow \; y(x) = A(x)\, y_i + B(x)\, y_{i+1} \tag{46}$$

$\rightarrow$ 2nd derivative=0 in interval and undefined/infinite at interval points

- idea: get interpolation with smooth 1st derivative and continuous in 2nd derivative

A flat spline (lath) with fixed points (ducks) has minimum energy of bending $\rightarrow$ e.g., used for construction of hulls



Burmester stencils are splines of 3rd degree

- if (assume!): not only y_i given, but also y_i'' $\rightarrow$ add cubic polynomial with 2nd derivative varying linearly between y_i'' to y_{i+1}'' *and* zero values for x_i and x_{i+1} (so y_i, y_{i+1} unchanged):

$$y(x) = A(x)\, y_i + B(x)\, y_{i+1} + C(x)\, y_i'' + D(x)\, y_{i+1}'' \tag{47}$$

$$C(x) \equiv \frac{1}{6}(A^3(x) - A(x))(x_{i+1} - x_i)^2 \qquad D(x) \equiv \frac{1}{6}(B^3(x) - B(x))(x_{i+1} - x_i)^2 \tag{48}$$

$\rightarrow x$ dependence only through $A(x)$, $B(x)$ $\rightarrow$ cubic x-dependence in $C(x)$, $D(x)$

- check: now y_i'' is 2nd derivative of interpolating polynomial (calculating $dA/dx, \ldots$):

$$\frac{dy}{dx} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{3A^2 - 1}{6}(x_{i+1} - x_i)y_i'' + \frac{3B^2 - 1}{6}(x_{i+1} - x_i)y_{i+1}'' \tag{49}$$

$$\frac{d^2y}{dx^2} = Ay_i'' + By_{i+1}'' \tag{50}$$

note that $A = 1$ and $B = 0$ at x_i; and $A = 0$ and $B = 1$ at x_{i+1}, so y'' is ok ($\checkmark$)

- however: in most cases y_i'' not known
  idea $\rightarrow$ 1st derivative shall be continuous across interval boundaries $\rightarrow$ gives equation for 2nd derivatives
- so: Eq. (49) shall be same for x_i on $[x_{i-1}, x_i]$ and on $[x_i, x_{i+1}]$ (for $i = 2, \ldots, N-1$) yielding $N-2$ equations

$$\frac{x_i - x_{i-1}}{6} y''_{i-1} + \frac{x_{i+1} - x_{i-1}}{3} y''_i + \frac{x_{i+1} - x_i}{6} y''_{i+1} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{51}$$

  with N unknown y_i'' $\rightarrow$ need further constraint
- often: y_1'' and y_N'' set to 0 $\rightarrow$ natural cubic spline
- advantage of cubic splines: linear set of equations and also tridiagonal, each y_i'' couples only to nearest neighbors

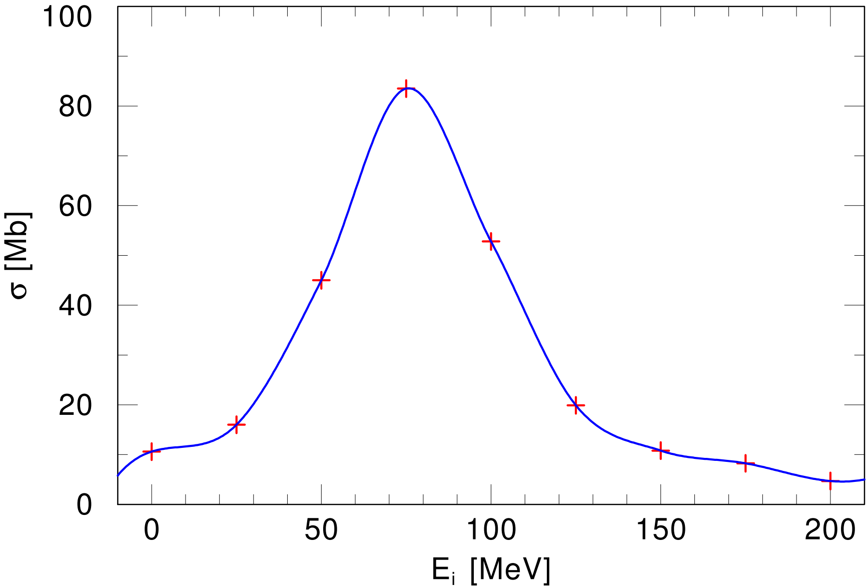
- hence with mapping $t = (x - x_i)/(x_{i+1} - x_i)$ on unit interval $[0, 1]$

$$p(t) = T\, M_\mathrm{h}\, C = (t^3 \;\, t^2 \;\, t) \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_i \\ y_{i+1} \\ m_i \\ m_{i+1} \end{pmatrix} \tag{52}$$

$$\begin{aligned} p(t) = {} & (2t^3 - 3t^2 + 1) y_i + (-2t^3 + 3t^2) y_{i+1} \\ & + (t^3 - 2t^2 + t) m_i + (t^3 - t^2) m_{i+1} \end{aligned} \tag{53}$$

with the numericial 1st derivatives $m_i = \frac{1}{2}\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}} + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$ and $m_{i+1} = \frac{1}{2}\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{y_{i+2} - y_{i+1}}{x_{i+2} - x_{i+1}}\right)$ and $m_1 = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$ and $m_n = 0$

Interpolating data XVIII

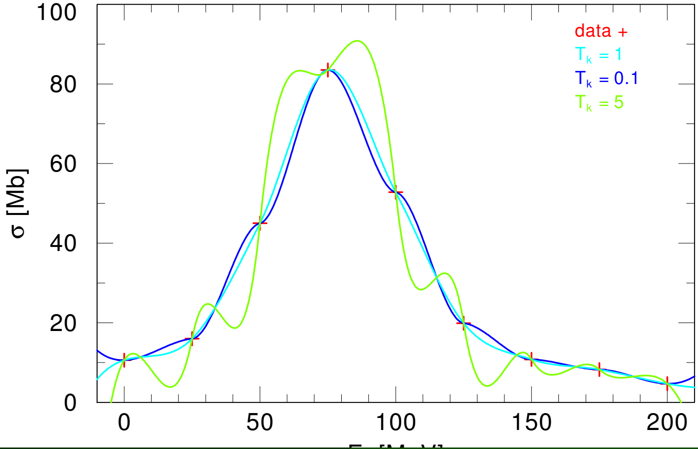


Cubic spline interpolation

Catmull-Rom splines

The “width” of the curve segment can be controlled by a parameter T_k according to (for $k = 2, \ldots, n-2$):

$$m_k = T_k \frac{y_{k+1} - y_{k-1}}{x_{k+1} - x_{k-1}} \tag{54}$$



Simplest method on a rectilinear 2D grid: bilinear interpolation, i.e, linear interpolation in one direction, then again in another direction
$\rightarrow$ as for Neville's algorithm $2\times$ linear = quadratic order
If four f values are given as follows: $f_1 : Q_{11} = (x_1, y_1)$, $f_2 : Q_{12} = (x_1, y_2)$, $f_3 : Q_{21} = (x_2, y_1)$, $f_4 : Q_{22} = (x_2, y_2)$ then
1. linear interpolation in x-direction:

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \tag{55}$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \tag{56}$$

2. linear interpolation in y-direction:

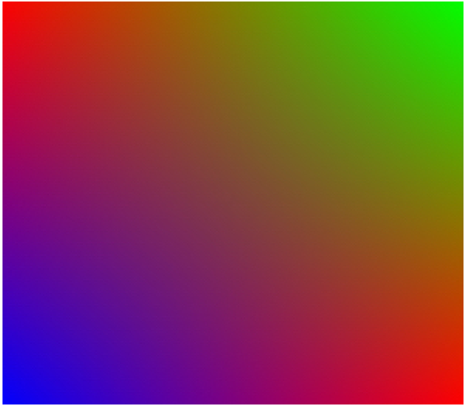
$$\begin{aligned}
f(x,y) &\approx \frac{y_2-y}{y_2-y_1} f(x,y_1) + \frac{y-y_1}{y_2-y_1} f(x,y_2) \\
&= \frac{y_2-y}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1} f(Q_{11}) + \frac{x-x_1}{x_2-x_1} f(Q_{21})\right) \\
&+ \frac{y-y_1}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1} f(Q_{12}) + \frac{x-x_1}{x_2-x_1} f(Q_{22})\right) \\
&= \frac{1}{(x_2-x_1)(y_2-y_1)} \left(f(Q_{11})(x_2-x)(y_2-y)\right. \\
&+ f(Q_{21})(x-x_1)(y_2-y) + f(Q_{12})(x_2-x)(y-y_1) \\
&\left.+ f(Q_{22})(x-x_1)(y-y_1)\right)
\end{aligned} \tag{57}$$

$\rightarrow$ same result as for 1. y-direction + 2. x direction

Interpolating in 2D III

So:

$$f(x,y) = \frac{1}{(x_2-x_1)(y_2-y_1)} \cdot \left(f_1(x_2-x)(y_2-y) + f_3(x-x_1)(y_2-y) + f_2(x_2-x)(y-y_1) + f_4(x-x_1)(y-y_1)\right) \quad (58)$$



Example, here: rgb colors on corner points
$f_{11} = b$, $f_{12} = f_{21} = r$, $f_{22} = g$

Interpolating in 2D IV

As the interpolation can also be written as:

$$f(x,y) = \sum_{i=0}^{1}\sum_{j=0}^{1} a_{ij}x^i y^j = a_{00} + a_{10}x + a_{01}y + a_{11}xy \tag{59}$$

$$a_{00} = f(0,0), \tag{60}$$

$$a_{10} = f(1,0) - f(0,0), \tag{61}$$

$$a_{01} = f(0,1) - f(0,0), \tag{62}$$

$$a_{11} = f(1,1) + f(0,0) - \big(f(1,0) + f(0,1)\big). \tag{63}$$

$\rightarrow$ interpolation only linear along lines of const. x or const. y, any other direction:
quadratic in position (but linear in f)

$\rightarrow$ other method: bicubic interpolation $f(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$ with 16 coefficients

$\rightarrow$ extension to 3D: trilinear interpolation, tricubic interpolation (64 coefficients)