Taylor's Theorem:

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)(x-x_0)^2}{2!}$$
$$+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)^{(n+1)}$$

where $x_0 \le \xi(x) \le x$

$P_n(x)=f(x)-R_n(x) \leftarrow$ truncation error/remainder

Single floating point operation: relative error |1-p*/p| is very small:$<b^{1-k}$(chopping), $<0.5b^{1-k}$(rounding) (b=base) but increases w/more fl() operations.

Problem Conditioning: Ill conditioned if exact solution changes greatly with small changes in data. Only well-conditioned if for ALL small $\varepsilon_i, \{\hat{r}_i\} \approx \{r_i\}$

$$\text{data } \{d_i\} \xrightarrow{\text{Exact Arithmetic}} \text{computed solution } \{r_i\}$$

$$\text{perturbed data } \{\hat{d}_i\}=\{d_i+\varepsilon_i\} \xrightarrow{\text{Exact Arithmetic}} \text{exact solution } \{\hat{r}_i\}$$

With $|\varepsilon_i/d_i|$ small.

Algorithm Stability: an algorithm is stable if it determines a computed solution (using floating-point arithmetic) that is close to the exact solution of some small perturbation of the given problem.

Stable if ANY data $\{\hat{d}_i\} \approx \{d_i\}, \{\hat{r}_i\} \approx \{r_i\}$

Only **unstable** if NO data $\{\hat{d}_i\} \approx \{d_i\}, \{\hat{r}_i\} \approx \{r_i\}$

$$\text{data } \{d_i\} \xrightarrow{\text{Floating Point}} \text{computed solution } \{r_i\}$$

$$\text{perturbed data } \{\hat{d}_i\}=\{d_i+\varepsilon_i\} \xrightarrow{\text{Exact Arithmetic}} \text{exact solution } \{\hat{r}_i\}$$

With $|\varepsilon_i/d_i|$ small.

Bisection Method: used to compute a zero of any function f(x) that is continuous on any interval [a,b] for which $f(a)\times f(b)<0$ has linear convergence $(\alpha=1)$

- *a and b are two initial approximations*
- *new approximation is the midpoint of [a,b],*
- $c=(a+b)/2$
- *if* $f(c)=0$, *stop*
  otherwise, a new interval that is half the length
- *of the previous interval is determined:*
  - o *if* $f(a)\times f(c)<0$ *set* $b \leftarrow c$
  - o *if* $f(b)\times f(c)<0$ *set* $a \leftarrow c$
- *repeat until [a,b] is sufficiently small – c will*
- *be close to a zero of f(x)*
- *if tolerance is specified, stop when* $(b-a)/2<\varepsilon$

Max absolute error of successive approx:

$$|p-p_n| \le \frac{b-a}{2^n},\ p_n=mid(a,b)$$

\# of iterations N to achieve absolute error < ε:

$$N>\frac{\ln(b-a)-\ln(\varepsilon)}{\ln(2)}$$

Newton's Method: if *p* is a zero of *f(x)*, and *p₀* is an approximation of *p*

$$p_n=p_{n-1}-\frac{f(p_{n-1})}{f'(p_{n-1})}$$

- always converges if the initial approximation p_0 is sufficiently close to the root *p*
- converges quadractically $(\alpha=2)$ if *p* is a simple zero (multiplicity, m = 1) otherwise linear

$f\in C^3[a,b]$ means $f(x), f'(x), f''(x)$ all exist and are continuous on [a,b], $p\in[a,b]$ is a root of $f(x)=0, \underline{f'(p)\ne 0}$

Multiplicity

- If a zero exists at *p* for *f(x)*, keep taking derivatives until $f^m(p)\ne 0$
- m = multiplicity
- If m= 1, then *p* is a simple zero of *f(x)*

Secant Method:

$$p_{n+1}=p_n-f(p_n)\frac{p_n-p_{n-1}}{f(p_n)-f(p_{n-1})}$$

Error in k^{th} approx.: $e_k=p_k-p$, $e_{n+1}\approx|e_n\|e_{n-1}|$

Order of convergence, $\alpha=\dfrac{(1+\sqrt{5})}{2}\approx 1.618$ for a simple zero

Horner's Algorithm: Given a polynomial

$$P(x)=\sum_{i=0}^{n}a_ix^i$$

and a value x_0, it evaluates *P(x₀)* and *P'(x₀)*

Given $a_0, a_1, \ldots a_n$ and x_0, compute:

$b_n = a_n$	$c_n = b_n$
$b_{n-1} = a_{n-1}+b_nx_0$	$c_{n-1} = b_{n-1}+c_nx_0$
$b_1 = a_1+b_2x_0$	$c_1 = b_1+c_2x_0 = P'(x_0)$
$b_0 = a_0+b_1x_0 = P(x_0)$	

$P(x) = (x-x_0)Q(x)+b_0$

original polynomial:

$P(x) = a_nx^n+ a_{n-1}x^{n-1}+\ldots a_1x+a_0$

deflated polynomial:

$Q(x) = b_1+b_2x+b_3x^2+b_nx^{n-1}$

Newton's Method w/Horner's Algorithm and Polynomial Deflation:

- to approx. a zero of P(x)
- choose an initial approx. p_0

i = N: use Horner's Algorithm to evaluate $\{b_n \ldots b_0\}$ and $\{c_n \ldots c_0\}$ so $b_0 = P(p_{N-1})$ and $c_1 = P'(p_{N-1})$

compute $p_N \leftarrow p_{N-1}-b_0/c_1$

p_N is the approx. to a zero of P(x)

deflated polynomial is $Q(x) = b_1+b_2x+b_3x^2+\ldots b_Nx^{N-1}$

Muller's Method:

$$P(x)=a(x-x_2)^2+b(x-x_2)+c$$

*P(x) and f(x) are equal at x_0, x_1, x_2

$$c=f(x_2)$$
$$b=\frac{(x_0-x_2)^2[f(x_1)-f(x_2)]-(x_1-x_2)^2[f(x_0)-f(x_2)]}{(x_0-x_2)(x_1-x_2)(x_0-x_1)}$$
$$a=\frac{(x_1-x_2)[f(x_0)-f(x_2)]-(x_0-x_2)[f(x_1)-f(x_2)]}{(x_0-x_2)(x_1-x_2)(x_0-x_1)}$$

to determine the next approx. x_3.

$$x_3=x_2-\frac{2c}{b+sign(b)\sqrt{b^2-4ac}}$$

Make the denominator as small as possible to make x_3 as close to x_2 as possible.

then reinitialize x_0, x_1, x_2 to be x_1, x_2, x_3 and repeat

Order of convergence = 1.84 for simple zero

Polynomial Interpolation:

- let y=f(x) and y_1=f(x_1) for given values x_1
- then P(x_1)=y_1 interpolates f(x)
- general form (n=2):

$$P(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1+\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2$$

- or linear interpolation (n=1)

$$P(x)=\frac{(x-x_1)}{(x_0-x_1)}y_0+\frac{(x-x_0)}{(x_1-x_0)}y_1$$

Error Term of Polynomial Interpolation:

$$f(x)=P(x)+\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)(x-x_n)$$

where P(x) is the interpolation polynomial and x_0, x_1, x_n are distinct points, therefore:

$$|f(x)-P(x)|\le\frac{1}{(n+1)!}\max|f^{(n+1)}(\xi(x))|\max|(x-x_0)(x-x_1)\ldots(x-x_n)|$$

- to determine max, either use inspection or take derivative and set it equal to 0
- assume f(x)∈$C^{(n+1)}$[a,b]

Runge Phenomenon:

- as n → ∞, f(x)-P_n(x)→∞, P_n(x) diverges from f(x) for all values of x such that 0.726 ≤ x ≤ 1 (except at the points of interpolation x_i)
- consider the error term for polynomial interpolation as an example of this

Lagrange Interpolation Polynomials:

$f(x_k)=P(x_k)$ for k = 0, 1, 2...n

$$P(x)=f(x_0)L_{n,0}(x)+\ldots f(x_n)L_{n,n}(x)=\sum_{k=0}^{n}f(x_k)L_{n,k}(x)$$

$$L_{n,k}(x)=\frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)}$$

$$L_{n,k}(x)=\prod_{\substack{i=0\\i\ne k}}^{n}\frac{(x-x_i)}{(x_k-x_i)}$$

Order of Convergence:

$$\lim_{n\to\infty}\frac{|P_{n+1}-P|}{|P_n-P|^{\alpha}}=\lambda$$

λ is positive → α is order of convergence

Cubic Spline Interpolantion:

S(x) is a cubic spline interpolant for f(x) if:

(a) S(x) is a cubic polynomial, denoted by S_j(x), on each subinterval [x_j, x_{j+1}], 0≤ j ≤n-1

(b) $S_j(x_j)$ = f(x_j), for 0≤ j ≤n-1 and $S_{n-1}(x_n)$=f(x_n)

(c) $S_{j+1}(x_{j+1})$ = $S_j(x_{j+1})$, for 0≤ j ≤n-2

(d) $S_{j+1}'(x_{j+1})$ = $S_j'(x_{j+1})$, for 0≤ j ≤n-2

(e) $S_{j+1}''(x_{j+1})$ = $S_j''(x_{j+1})$, for 0≤ j ≤n-2

(f) and either one of:

(i) S″(x_0) = S″(x_n) = 0 (free/natural boundary condition)

(ii) S′(x_0) = f′(x_0) and S′(x_n) = f′(x_n) (clamped boundary condition)

There are infinite solutions for conditions (a)-(e) with 4n-2 conditions to be satisfied in 4n unknowns. If (f) is satisfied, there are only 4n conditions in 4n unknowns and a unique S(x).

Numerical Differentiation Formulas:

Given f(x) and $\hat{x}$. Write a Lagrange interpolating polynomial, then differentiate it and its error term. Solve $f(\hat{x})$.

$$f'(x)=\sum_{k=0}^{n}f(x_k)L_k'(x)+f^{(n+1)}(\xi(x))\frac{d}{dx}\left[\frac{(x-x_0)(x-x_1)(x-x_n)}{(n+1)!}\right]$$

simplified error term when x=x_i:

$$f^{(n+1)}(\xi(x_j))\frac{d}{dx}\left[\frac{(x-x_0)(x-x_1)(x-x_n)}{(n+1)!}\right]_{x=x_j}=\frac{f^{(n+1)}(\xi(x_j))}{(n+1)!}\left[\prod_{\substack{k=0\\k\ne j}}^{n}(x_j-x_k)\right]$$

For n = 1, j = 0:

$$f'(x_0)=\frac{f(x_1)-f(x_0)}{x_1-x_0}\text{ with error term }\frac{f''(\xi(x))}{2}(x_0-x_1)$$

For n = 2 (middle of 3 data points), j = 1:

$$f'(x_1)=f(x_0)\left[\frac{x_1-x_2}{(x_0-x_1)(x_0-x_2)}\right]+f(x_1)\left[\frac{2x_1-x_1-x_2}{(x_1-x_0)(x_1-x_2)}\right]$$
$$+f(x_2)\left[\frac{x_1-x_0}{(x_2-x_0)(x_2-x_1)}\right]$$

with error term $\dfrac{f'''(\xi(x_1))}{6}(x_1-x_0)(x_1-x_2)$ where ξ_1 lies between x_0 and x_2

For equally spaced data, x_1-x_0 = x_2-x_1 = h:

$$f'(x_1)=\frac{1}{2h}(f(x_2)-f(x_0))-\frac{h^2}{6}f'''(\xi_1)\text{ where }x_0-h\le\xi_1\le x_0+h$$

For n = 2, j = 0 or 2 (equally spaced data):

$$f'(x_0)=\frac{1}{2h}(-3f(x_0)+4f(x_0+h)-f(x_0+2h))+\frac{h^2}{3}f'''(\xi_0)$$

where $x_0\le\xi_0\le x_0 2h$

Numerical Differentiation Using Taylor's Theorem:

$$f(x_o+h)=f(x_0)+hf'(x_0)+\frac{h^2}{2}f''(x_0)+\frac{h^{n+1}}{(n+1)!}f'''(\xi_1)$$

for –h, etc, replace all h by –h.Write Taylor for all points, do linear combination to cancel as many derivatives as possible.

Richardson's Extrapolation:

Case 1:

$$M=N_1(h)+K_1h+K_2h^2+K_3h^3\ldots$$

M = exact value, N_1(h) = computed approx. using stepsize h, K_1h...= truncation error of O(h)

Use same formula to approx M, but replace h by h/2 (or h/some value). Determine linear combination of two formulas for M so that largest term in truncation error (O(h)) cancels out. N calculation for extrapolation table:

$$N_j\left(\frac{h}{2^j}\right)=N_{j-1}\left(\frac{h}{2^{j+1}}\right)+\frac{N_{j-1}\left(\frac{h}{2^{i+1}}\right)-N_{j-1}\left(\frac{h}{2^i}\right)}{2^{j-1}-1}\text{ for j}\ge 2$$

Case 2:

$$M=N_1(h)+K_1h^2+K_2h^4+K_3h^6\ldots$$

M has truncation error of O(h^2).

Replace h by h/2 (or h/some value) and follow steps in case 1.

N calculation for extrapolation table:

$$N_j\left(\frac{h}{2^i}\right)=N_{j-1}\left(\frac{h}{2^{i+1}}\right)+\frac{N_{j-1}\left(\frac{h}{2^{i+1}}\right)-N_{j-1}\left(\frac{h}{2^i}\right)}{4^{i-1}-1}\text{ for j}\ge 2$$

has truncation error of O(h^{2j}).

Newton-Cotes Closed Quadrature Formulas:

Trapezoidal rule (n=1):

$$\int_a^b f(x)dx=\frac{h}{2}[f(x_0)+f(x_1)]-\frac{h^3}{12}f''(\xi),$$

$h=x_1-x_0$ degree $=1$

Simpson's rule (n=2):

$$\int_a^b f(x)dx=\frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)]-\frac{h^5}{90}f^4(\xi)$$

$h=(b-a)/2$ degree $=3$

Simpsons Rule: (deg of precision is 3)

$f(x)$	$\int_a^b f(x)dx$	$\frac{h}{3}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$
1	$b-a$	$\frac{b-a}{6}[1+4(1)+1]=b-a$
x	$\frac{b^2-a^2}{2}$	$\frac{b-a}{6}\left[a+4\left(\frac{a+b}{2}\right)+b\right]=\frac{b^2-a^2}{2}$
x^2	$\frac{b^3-a^3}{3}$	$\frac{b-a}{6}\left[a^2+4\left(\frac{a+b}{2}\right)^2+b^2\right]=\frac{b^3-a^3}{3}$
x^3	$\frac{b^4-a^4}{4}$	$\frac{b-a}{6}\left[a^3+4\left(\frac{a+b}{2}\right)^3+b^3\right]=\frac{b^4-a^4}{4}$
x^4	$\frac{b^5-a^5}{5}$	$\frac{b-a}{6}\left[a^4+4\left(\frac{a+b}{2}\right)^4+b^4\right]\ne\frac{b^5-a^5}{5}$

Composite Simpson's Rule:

m applications of Simpson's rule on [a,b] requires [a,b] be subdivided into an even # (2m) of subintervals each of length:

$$h=\frac{b-a}{2m}$$

$$\int_a^b f(x)dx=\frac{h}{3}\left[f_0+2\sum_{j=1}^{m-1}f_{2j}+4\sum_{j=1}^{m}f_{2j-1}+f_{2m}\right]$$

Case m = 2:

Requires 5 quadrature points, 4 subintervals.

$$\int_a^b f(x)dx=\frac{h}{3}[f_0+4f_1+2f_2+4f_3+f_4]$$

Case m = 4:

Requires 9 quadrature points, 8 subintervals.

$$\int_a^b f(x)dx=\frac{h}{3}[f_0+4f_1+2f_2+4f_3+2f_4+4f_5+2f_6+4f_7+f_8]$$

Truncation error: $E_{2m}(f)=-\dfrac{(b-a)h^4}{180}f^4(\mu)$

Romberg Integration: is the application of Richardson's Extrapolation to the composite trapezoidal rule approximations

$R_{k,1}$ is the composite trapezoidal rule approx. to $\int_a^b f(x)dx$ using 2^{k-1} sub intervals

$$R_{k,j}=R_{k,j-1}+\frac{R_{k,j-1}-R_{k-1,j-1}}{4^{j-1}-1}$$

for k = 2,3,4,... and j = 2,3,...,k (use for columns 2,3,...)

$$\begin{array}{llll} R_{1,1} & & & \\ R_{2,1} & R_{2,2} & & \\ R_{3,1} & R_{3,2} & R_{3,3} & \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

$$R_{1,1}=\frac{h}{2}[f_0+f_1]\quad,\ \text{where } h=b-a$$

$$R_{2,1}=h\left[\frac{f_0}{2}+f_1+\frac{f_2}{2}\right]\quad,\ \text{where } h=\frac{b-a}{2}$$

$$R_{3,1}=h\left[\frac{f_0}{2}+f_1+f_2+f_3+\frac{f_4}{2}\right]\quad,\ \text{where } h=\frac{b-a}{4}$$

$$R_{2,2}=R_{2,1}+\frac{R_{2,1}-R_{1,1}}{3}$$

Truncation Error: First column = O(h^2), Second Column = O(h^4), Third O(h^6) ...

Error Term, First column: $-\dfrac{b-a}{12}h^2f''(\mu)$, where $a<\mu<b$

Adaptive Quadrature:

S_1 = (non-composite) Simpson's rule approx to $\int_a^b f(x)dx$

S_2 = composite Simpson's rule approx using 2 applications of Simpson's rule

$$\int_a^b f(x)dx=S_2-\frac{h^5}{1440}f^4(\tilde{\mu})\approx S_2+\frac{1}{15}(S_2-S_1)$$

$$\left|\int_a^b f(x)dx-S_2\right|\approx\frac{1}{15}|S_2-S_1|$$

If $|S_2-S_1|<15\varepsilon$, $\left|\int_a^b f(x)dx-S_2\right|<\varepsilon$

else, select smaller h and repeat.

DE Equations: An initial-value problem has a unique solution if f(t,y(t)) is continuous on D={(t,y)|a≤t≤b, -∞<y<∞} and if f satisfies a **Lipshitz** condition on D:

$|f(t,y_1)-f(t,y_2)| \le L|y_1-y_2|$ for some constant L and all $(t,y_i)\varepsilon D$

Euler's Method: is obtained from this truncated Taylor polynomial approx. by replacing $y(t_{i+1})$ by its numerical approx. w_{i+1}, and similarly replacing $y(t_i)$ by w_i:

$w_0 = \alpha$ (initial condition)

$w_{i+1} = w_i + h \cdot f(t_i, w_i)$

Geometric interpretation of Euler's Method:

$w_0 = \alpha = y(t_0)$

$w_1 = w_0 + h \cdot f(t_0, w_0) = w_0 + h \cdot y'(t_0)$

$\frac{w_1 - w_2}{h} = y'(t_0)$

$w_2 = w_1 + h \cdot f(t_1, w_1)$

Global Truncation Error at t_i is $|y_i-w_i|$

If the global truncation error is $O(h^k)$, numerical method for computing w_i is said to be of the order *k*, the larger the order of k, the faster the rate of convergence.

A difference method is said to be convergent (wrt the differential equation it approx.) if:

$$\lim_{h\to 0}\max_{1\le i\le N}|y_i - w_i| = 0$$

$$\max_{1\le i\le N}|y_i - w_i| \le \frac{hM}{2L}\left[e^{L(b-a)}-1\right]$$

Euler's method is of order 1.

Local Truncation Error: the error incurred in one step of a numerical method assuming that the value at the previous mesh point is exact.

For Euler's Method,

$v_{i+1} = y(t_i) + h \cdot f(t_i, y(t_i))$

$|y_{i+1} - v_{i+1}| = \left|\frac{h^2}{2}y''(\xi_i)\right| \le \frac{h^2}{2}M \quad , \quad y''(\xi_i) \le M$

Local truncation error is $O(h^2)$

If the local truncation error is $O(h^{k+1})$ then the global is $O(h^k)$ – method is order k

Disadvantages of Eulers: not sufficiently accurate. Global truncation is O(h), means that h must be small for high accuracy approx.

Taylor's Method of order n

$$w_{i+1} = w_i + h\cdot f(t_i,w_i) + \frac{h^2}{2}f'(t_i,w_i) + \ldots + \frac{h^n}{n!}f^{(n-1)}(t_i,w_i) \quad , \quad \text{int } n \ge 1$$

Truncation error is $O(h^{n+1})$: just above remainder term:

$$\left|\frac{h^{n+1}}{(n+1)!}\right| y^{(n+1)}(\xi_i)$$

Global Truncation is $O(h^n)$, so order n

Euler's method is just special case where n = 1

Runge-Kutta:

Advantage of Taylor method: high order (high accuracy) truncation error.

Disadvantage: high derivatives

R-K are higher order formulas that require function evaluations of only f(t,y(t)), and not its derivatives, by using Taylor for 2 variables.

General form:

$$w_{i+1} = w_i + \sum_{j=1}^{m} a_j k_j$$

Where $k_1 = h \cdot f(t_i, w_i)$ and $k_j = h \cdot f\left(t_i + \alpha_j h, w_i + \sum_{l=1}^{j-1}\beta_{jl}k_l\right)$

Choose parameters $\{a_i\}$,$\{alpha_i\}$,$\{\beta_{ij}\}$ so that the general R-K formula is identical to the Taylor series expansion

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2}y_i'' + \cdots$$

Runge-Kutta-Fehlberg:

Variable stepsize implementation of R-K that insures that the global truncation error is within ε.

$|y(t_{i+1})-w_{i+1}|<\varepsilon$

The global truncation error is estimated by finding 2 approximations, using 2 different R-K formulas, w_{i+1} with $O(h^{n+1})$ and $\hat{w}_{i+1}$ with $O(h^{n+2})$. The global error estimate is: $|\hat{w}_{i+1}-w_{i+1}|/h$

Direct Methods for Solving Linear Systems

Ax=b, n equations, n unknowns

case 1

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$

$\vdots$

$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$

or case 2, $P(x) = a_o + a_1 x + a_2 x^2$

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \end{bmatrix}$$

forward elimination

$m_{21} = \frac{a_{21}}{a_{11}},\ E_2 \leftarrow E_2 - m_{21}E_1$

$m_{31} = \frac{a_{31}}{a_{11}},\ E_3 \leftarrow E_3 - m_{31}E_1$

$m_{32} = \frac{a_{32}}{a_{12}},\ E_3 \leftarrow E_3 - m_{32}E_1$

Make upper triangular and back substitute

Algorithm

forward elimination

```
For i=1 to n-1
    For j=(i+1) to n
        mult←aji/aii
        For k=i+1 to n
            ajk← ajk-mult· aik
    bj←bj-mult·bi
```

Back substitution

$x_n \leftarrow b_n/a_{nn}$

For i=n-1 to 1

$x_i \leftarrow \left[b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right] / a_{ii}$

Fails with divide by zero (a_{ii}=0)

Partial Pivoting to reduce error

Find largest pivot

```
p←i
for k=1 to n
    if |aki|>|api| then p←k
```

row swap if necessary

```
 if p≠i
For k=i to n
    temp←aik
    aik←apk
    apk←temp
temp←bi
bi←bp
bp←temp
```

continue with gaussian

Requires $n^3/3$ mults/divides (adds/subs as well)

Matlab Ax=b type A\b for Gausian with PPivot

$\det A = (-1)^m a_{11}a_{22}\ldots a_{nn}$ where m=# of row swaps

To find inverse solve [A|I]

Better method, if need $A^{-1}b$, solve for x in Ax=b

If $A^{-1}B$ is needed, more efficient to solve AX=B for X

Stability

Stable if there exists small perturbations

E,e such that $\hat{x}$ is close to the exact solution

(A+E)y=b+e

small perturbation means $\frac{\|E\|}{\|A\|}, \frac{\|e\|}{\|b\|}$ are small

$\|A\| = \sqrt{\sum\sum a_{ij}^2} \quad \|x\| = \sqrt{\sum x_i^2}$

Gaussian elimination with PPivoting is almost always stable

Condition of problem Ax=b

Ill-conditioned if there exists one small perturbation of the data (A+E)y=b+e for which the exact value of y

Can be measured by the condition number of A: $\|A\|\|A^{-1}\|$

Matlab cond(A)

Various Formulae:

Matrix algebra: Ax=B → $x=A^{-1}B$

Taylor approximations:

$\sin(x) \approx x - \frac{x^3}{6} \qquad e^x \approx 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \ldots + \frac{x^n}{n!}$

$\cos(x) \approx 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6$

$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \qquad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$

Trig Identities:

sin2θ=2sinθcosθ

sin(a-b)=sinacosb-cosasinb

cos(a-b)=cosacosb+sinasinb

$\cos 2\theta = \cos^2\theta - \sin^2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$

Examples:

Determine second order (n=2) Taylor polynomial approx for $f(x)=x^{1/4}$ expanded about $x_0=1$. Include remainder term.

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3$$

$f(x_0) = 1 \qquad f'(x_0) = \frac{1}{4}x_0^{-3/4} = \frac{1}{4}$

$f''(x_0) = -\frac{3}{16}x_0^{-7/4} = -\frac{3}{16} \qquad f'''(x_0) = \frac{21}{64}x_0^{-11/4} = \frac{21}{64}$

$\therefore f(x) = 1 + \frac{1}{4}(x-1) - \frac{3}{32}(x-1)^2 + \frac{7}{128}(\xi(x))^{-11/4}(x-1)^3$

Determine a good upper bound for the truncation error of the taylor poly approx. above when 0.95≤ x ≤1.06 by bounding the remainder term. Give 4 sig digits.

$|f(x) - P(x)| \le \max\left|\frac{7}{128}(\xi(x))^{-11/4}(x-1)^3\right|$

$\le \frac{7}{128}\left|(0.95)^{-11/4}(1.06-1)^3\right| \le 1.360 * 10^{-5}$

Let $P(x)=x^4+x^2-3$. If Muller's method is used to approx. a zero of P(x) using initial approx. 0,1,2, give the Lagrange form of the interpolating polynomial.

$x_0 = 0, x_1 = 1, x_2 = 2$

$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-1)(x-2)}{(-1)(-2)}$

$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x)(x-2)}{(1)(-1)}$

$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x)(x-1)}{(2)(1)}$

$f(x_0) = P(x_0) = -3$

$f(x_1) = P(x_1) = -1$

$f(x_2) = P(x_2) = 17$

$\therefore P(x) = -3L\ (x) - L_1(x) + 17L_2(x)$

What values of x, where x>1 is the following expression subject to subtractive cancellation (which will produce a very inaccurate result using floating point arithmetic)? $f(x) = \sqrt{x} - \sqrt{x-1}$

X large and positive

How should f(x) be evaluated in floating point arithmetic in order to avoid subtractive cancellation?

$$f(x) = \left(\sqrt{x} - \sqrt{x-1}\right) * \frac{\sqrt{x}+\sqrt{x-1}}{\sqrt{x}+\sqrt{x-1}} = \frac{1}{\sqrt{x}+\sqrt{x-1}}$$

Apply Newton's method to $f(x) = x^2 - \frac{1}{c}$ in order to determine an interitive formula for computing $\frac{1}{\sqrt{c}}$

$f(x) = x^2 - \frac{1}{c} \qquad f'(x) = 2x$

$$P_n = P_{n-1} - \frac{P_{n-1}^2 + \frac{1}{c}}{2P_{n-1}} = \frac{P_{n-1}^2 + \frac{1}{c}}{2P_{n-1}} \text{ or } \frac{1}{2}\left(P_{n-1} + \frac{1}{cP_{n-1}}\right)$$

$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \ldots$

Use this form of Taylor's theorem to determine the quadratic approx to sin(1+h).

$\sin(1+h) \approx \sin(1) + h\cos(1) + \frac{h^2}{2}(-\sin(1))$

Apply the Newton-Raphson method to $f(x) = x^2 - \frac{R}{x}$ in order to determine an iterative formula for computing $\sqrt[3]{R}$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - \frac{R}{x_n}}{2x_n + \frac{R}{x_n^2}} = x_n - \left(\frac{x_n^3 - R}{x_n}\right)\left(\frac{x_n^2}{2x_n^3 + R}\right)$$

$$= x_n\left(1 - \frac{x_n^3 - R}{2x_n^3 + R}\right) = x_n\left(\frac{2x_n^3 + R - x_n^3 + R}{2x_n^3 + R}\right) = x_n\left(\frac{x_n^3 + 2R}{2x_n^3 + R}\right)$$

A good approx to a zero of $P(x)=x^4+x^3-6x^2-7x-7$ is x_0=2.64. If x_0 is used as an approx to a zero of P(x), use synthetic division (Homer's algorithm) to determine the associated deflated polynomial.

$a_0 = -7, a_1 = -7, a_2 = -6, a_3 = 1, a_4 = 1$

$b_4 = a_4 = 1$

$b_3 = a_3 + b_4 x_0 = 3.64$

$b_2 = a_2 + b_3 x_0 = 3.6096$

$b_1 = a_1 + b_2 x_0 = 2.529344$

$b_0 = -0.322532$

$\therefore$ deflated polynomial is $x^3 + 3.64x^2 + 3.6096x + 2.529344$

If a, b, c, d, e, f have known values then $\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix}$ is a system of 2 linear equations in the 2 unknowns x,y. If ad-bc ≠ 0, then the solution is $x = \frac{de-bf}{ad-bc}$ and $y = \frac{af-ce}{ad-bc}$. Consider the system: $\begin{bmatrix} 0.96 & -1.23 \\ 4.91 & -6.29 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.27 \\ -1.38 \end{bmatrix}$ Show that the problem of computing the solution $\begin{bmatrix} x \\ y \end{bmatrix}$ is ill conditioned.

Almost any perturbation of the 6 constants in the data will do. For example, $\begin{bmatrix} 0.961 & -1.23 \\ 4.89 & -6.29 \end{bmatrix}\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} -0.27 \\ -1.38 \end{bmatrix}$ has the exact solution $\approx \begin{bmatrix} -0.03 \\ 0.196 \end{bmatrix}$, whereas the given system has solution $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Let $f(x) = -\frac{1}{2} - \frac{x}{3} - \frac{x^2}{24}$. F(x) is accurate when x is close to 0. Show that the computation of fl(f(0.123)) is unstable.

given problem $x = 0.123$ → computed solution -0.4629

perturbed problem $\hat{x} = 0.123 + \varepsilon$ → $f(\hat{x}) \approx -\frac{1}{2} - \frac{\hat{x}}{3} - \frac{\hat{x}^2}{24}$ for x close to 0

$f(\hat{x}) = -\frac{1}{2} - \frac{0.123+\varepsilon}{3} - \frac{(0.123+\varepsilon)^2}{24}$

$f(\hat{x}) = -0.54163 - 0.34358\varepsilon + O(\varepsilon^2)$

$f(\hat{x}) \approx -0.5416$ for all ε such that $\left|\frac{\varepsilon}{0.123}\right|$ is small.

Since this is not close to -0.4629 for all small ε, the computation is unstable.

$fl(fl(w*x) - fl(y*z))$ for w=16.00, x=43.61, y=12.31, z=56.68

Using idealized, rounding, floating point arithmetic (base 10, k=4), the above equation evaluates to 0.1. The exact value is 0.0292. Use the definition of stability to show (by using only a perturbation in y) that the computation is stable.

To show that its stable, find a value ε for which the exact value of (16.00)(43.61)-(12.31+ε)(56.68) is approx equal to 1. Solving for ε gives $\varepsilon = \frac{16*43.61 - 0.1}{56.68} - 12.31 = -0.0012491$

Thus, for example, if y =12.31 is perturbed to $\hat{y} = y + \varepsilon = 12.31 - 0.00125$ then the exact value of $w*x - \hat{y}*z = 0.10005$, which is close to 0.1.

And since $\left|\frac{\varepsilon}{0.123}\right| = \frac{0.00125}{12.31}$ is small, the computation is stable.

Examples

Problem

Suppose that the absolute errors in three consecutive approximations with some iterative method are $e_n = 0.07$; $e_{n+1} = 0.013$; $e_{n+2} = 0.00085$ Use the definition of convergence to estimate the order of convergence of the iterative method.

Solution

Definition of the order of convergence $\propto$: $\lim_{n \Rightarrow \infty} e_{n+1}/ e_n^{\propto} = k$; if n is sufficiently large then, $e_{n+1}/ e_n^{\propto} = k = e_{n+2}/ e_{n+1}^{\propto} \Rightarrow e_{n+1}/ e_{n+2} = k = (e_n/ e_{n+1})^{\propto}$; solve for $\propto$ ($\propto$=1.62)

Problem

a) Compute two approximations computed to $\int x/(1+x^2)\,dx$, using the noncomposite Simpsons rule(with h=0.1) and using the composite Simpsons rule (with h=0.05). bounds: [0,0.2]

b) The truncation error term for the two approximations computed in a) to obtain an $O(h^4)$ approximation to the value of the given definite intergal.

Solution

a) Simpsons rule(h=0.1) $I_1 = h/3\,[f(0) + 4f(0.1) + f(0.2)] = 0.01961158$: Composite

Simpsons rule(h=0.05) $I_2 = h/3\,[f(0) + 4f(0.05) + 2f(0.1) + 4f(0.15) + f(0.2)] = 0.0196143$

b) width h= 0.1

(1) $I = I_1 + kh^4 + O(h^6)$

(2) $I = I_2 + k(h/2)^4 + O(h^6)$

16*(2) - (1) solve for $I \Rightarrow I = I_2 + (I_2 - I_1)/15 + O(h^6)$

$\therefore I = 0.019103533$

Problem

The open-newton cotes quadrature formula(for n=1) that approximates $\int x g(x)\,dx$ is $3h/2\,(g(x_o)+ g(x_1))$ where $h = (x_o - x_1)/3$ and $x_i = x + (i+1)h$ for i= 0,1,2,…; recall that Euler's method can be derived by intergrating this differential equation $y'(t) = f(t,y(t))$ over $[t_I, t_{I+1}]$ and using a simple "rectangular" rule to approximate $\int f(t,y(t))\,dt$. Derive a formula for approximating th solution to $y'(t) = f(t,y(t))$ by intergrating this differential equation over $[t_I, t_{I+1}]$ and approximating $\int f(t,y(t))\,dt$ by the above open newton cotes quadrature formula for n=1.

Solution

$\int y'(t) = \int f(t,y(t)) \therefore y(t_{I+3}) = y(t_I) + \int f(t,y(t)) \approx y(t_I) + 3h/2\,(f(t_{I+1},y(t_{I+1}))+ f(t_{I+2},y(t_{I+2})))$ which suggests $\omega_{I+3} = \omega_i + 3h/2\,(f(t_{I+1},\omega_{I+1}))+ f(t_{I+2},\omega_{I+2})))$

1. Let P(x) denote the (linear) polynomial of degree 1 that interpolates f(x) = cos x at the points $x_0 = -0.1$ and $x_1 = 0.1$ (where x is in radians). Use the error term of polynomial interpolation to determine an upper bound for P(x) - f(x), where $x \in [-0.1, 0.1]$. Do not construct P(x).

Solution: $|P(x) - f(x)| = |\, f''(E)\,(x - 0.1)(x + 0.1) / 2!\,|$ with $-0.1 < E < 0.1$

$= |\,{-\cos(E)}\,(x^2 - 0.01) / 2\,| \;<=\; [\cos(0) / 2]\ \mathrm{Max}(-0.1 < E < 0.1)|x^2 - 0.01|$

$= 0.5\,|0 - 0.01| = 0.005$

2. Lagrange form of P(x): $P(x) = L_{2,0}(x)y_0 + L_{2,1}(x)y_1 + L_{2,2}(x)y_2$

$= (x - x_1)(x - x_2)y_0 / (x_0 - x_1)(x_0 - x_2) + (x - x_0)(x - x_2)y_1 / (x_1 - x_0)(x_1 - x_2)$

$+ (x - x_0)(x - x_1)y_2 / (x_2 - x_0)(x_2 - x_1)$

1. Consider evaluation of $f(x) = 1 / [1 - \tanh(x)]$, where $\tanh(x) = (e^x - e^{-x}) / (e^x + e^{-x})$

If f(x) is to be evaluated in floating-point arithmetic (e.g., 4 decimal digit, idealized, rounding floating-point), for each of the following ranges of values of x, specify whether

the computed floating-point result will be accurate or inaccurate. No justification is required.

(a) *x* is large and positive (for example, *x* > 4) Ans: Inaccurate

(b) *x* is close to 0 (for example, *x* £ 0. 001) Ans: Accurate

(c) *x* is large and negative (for example, *x* < -4) Ans: Accurate

2. Consider $g(h) = [\sin(1 + h) - \sin(1)] / h$, $h \neq 0$, where the arguments for sin are in radians . When h is close to 0, evaluation of g(h) is inaccurate in floating-point arithmetic. In (a) and (d) below, use 4 decimal digit, idealized, rounding floating-point arithmetic. If x is a floating-point number, assume that fl(sin x) is determined by rounding the exact value of sin x to 4 significant digits.

(a) Evaluate fl(g(h)) for h = 0. 00351:

fl(1+h)= fl(1+0.00351)= fl(1.00351)=1.004

fl(sin(1+h))= fl(sin(1.004))= fl(0.843625...)=0.8436

fl(sin(1))= fl(0.841470...)=0.8415

fl(sin(1+h)-sin(1))= fl(0.8436-0.8415)=0.002100

fl((sin(1+h)-sin(1))/h)= fl(0.002100/0.00351)= fl(0.598290...)=0.5983

Note: exact value is 0.538824...

(b) Taylor's Theorem can be expressed in two equivalent forms: The way as defined, or by using a change of variable (replacing x by x0 + h, so that h = x - x0 is the independent

variable), $f(x0 + h) = f(x0) + hf'(x0) + h^2f''(x_0)/2! + h^3f'''(x_0) / 3! + \ldots$

Using the latter form of Taylor's Theorem (without the remainder term), determine the

quadratic (in h) Taylor polynomial approximation to sin(1+h). Note: leave your answer

in terms of cos(1) and sin(1); do not evaluate these numerically.

Solution: $\sin(1+h) \approx \sin(1) + h(\cos(1)) + h^2\,(-\sin(1)) / 2$

(c) Use the Taylor polynomial approximation from (b) to obtain a polynomial approximation, say p(h), to g(h).

Solution: $p(h) = [\sin(1) + h\cos(1) - (h^2 / 2)\sin(1)) - \sin(1)] / h = \cos(1) - h\sin(1)/2$

(d) Show that p(h) is much better than g(h) for floating-point evaluation when h is small by evaluating fl(p(0.00351)):

Solution:

fl(cos(1)) = fl(0.540302...) = 0.5403

fl(h/2) = fl(0.00351/2) = 0.001755

fl(sin(1)) = fl(0.841471...) = 0.8415

fl((h/2)sin(1)) = fl(0.001755 x 0.8415) = fl(0.0014768325) = 0.001477

fl(cos(1)-(h/2)sin(1)) = fl(0.5403 - 0.001477) = fl(0.538823) = 0.5388

3. If a, b,c,d,e, *f* have known values, then [[*a b*] [*c d*]] * [*x y*] = [*e f*] is a system of 2 linear equations in the 2 unknowns *x* and y. If $ad - bc \neq 0$, then the solution is

$x = (de - bf) / (ad - bc)$ and $y = (af - ce) / (ad - bc)$. Consider the linear system

[[0.96 -1.23] [4.91 -6.29]] * [*x y*] = [-0.27 -1.38] Show that the problem of computing the solution [*x y*] is ill-conditioned.

Solution: Almost any perturbation of *a, b, c, d,* and / or *e,f* will do: For example,

[[0.961 –1.23] [4.89 –6.29]] * [x y] = [-0.27 –1.38] has solution ≈ [-0.03 0.196]

whereas the given system has solution [1 1]

4. (a) Let R denote any positive number. Apply the Newton-Raphson method to

$f(x) = x^2 - (R / x)$ in order to determine an iterative formula for computing $\sqrt[3]{R}$ Simplify the formula so that it is in the form: $x_n\,[g(x_n) / h(x_n)]$

where g(*xn*) and h(*xn*) are simple polynomials in *xn*.

Solution: $x_{n+1} = x_n - f(x_n)/f(x_n) = x_n - (x^2 - (R/x_n)) / (2x_n + R/x_n) = x_n[1- (x_n^3 - R)/ (2x_n^3 + R)] = x_n(2x_n^3 - R - x_n^3 + R)/(2x_n^3 + R) = x_n\,(x_n^3 + 2R)/(2x_n^3 + R)$

(b) Consider the case *R* = 2. Given some initial value *x* 0 , if the iterative formula in

(a) converges to $\sqrt[3]{2}$, what will be the order of convergence? Very briefly justify your

answer, referring to any results from your class notes or the textbook.

Solution: $f'(x) = 2x + R/x^2 = 2x + 2/x^2$, so at the root $x = \sqrt[3]{2}$

$f'(\sqrt[3]{2}) = 2\,\sqrt[3]{2} + 2 / 2^{2/3} \neq 0 \Rightarrow \sqrt[3]{2}$ is a simple zero $\Rightarrow$ quadratic convergence

5. A good approximation to one of the zeros of $P(x) = x^4 + x^3 - 6x^2 - 7x - 7$ is x0 = 2.64 . If x0 is used as an approximation to a zero of P(x), use synthetic division

(that is, Horner's algorithm) to determine the associated deflated polynomial.

Note: do not do any computations with the Newton-Raphson method.

Solution: $b_4 = 1 b_3 = a_3 + b_4x_0 = 1 + 1(2.64) = 3.64$

$b_2 = a_2 + b_3x_0 = -6 + 3.64(2.64) = 3.6096$ $b_1 = 2.529344$

$b_0 = -0.322532$

Deflated Poly: $x^3 + 3.64x^2 + 3.6096x + 2.529344$

6. Let P(x) denote the (linear) polynomial of degree 1 that interpolates f(x) = cos x

at the points x0 = -0.1 and x1 = 0.1 (where x is in radians). Use the error term of polynomial interpolation to determine an upper bound for |P(x) – f(x)|, where $x \in [-0.1, 0.1]$. Do not construct P(x).

Solution: $|P(x) - f(x)| = |f''(\xi)(x-0.1)(x+0.1)/2!|$ with $-0.1 < \xi < 0.1$

$= |{-\cos(\xi)}(x^2 - 0.01)/2| <= \cos(0)/2\ \max(-0.1 <= \xi <= 0.1)|x^2 - 0.01|$

$= 0.5\,|0 - 0.01| = 0.005$

Sample Final:

1. $f(x) = \sqrt{x} - \sqrt{(x-1)}$, where x > 1. Formula is inaccurate for large positive x. No problem when x ≈ 1. How should f(x) be evaluated in floating point arithmetic to avoid the subtractive cancellation? $f(x) = [\sqrt{x} - \sqrt{(x-1)}]\ x\ (\sqrt{x} + \sqrt{(x-1)}) / (\sqrt{x} - \sqrt{(x-1)})$

$= 1 / (\sqrt{x} + \sqrt{(x-1)})$

2. Let $f(x) = [(\sin(x) - e^x) + 1] / x^2$, $x \neq 0$, x in radians.

To 4 significant digits, the exact value of f(0.123) is -0.5416, and fpn computation is inaccurate. In order to obtain a better formula for approximating f(x) when x is close to 0, use the Taylor polynomial approximations for e^x and sin(x) (both expanded

about x0 = 0) in order to obtain a quadratic polynomial approximation for f(x).

Solution: $\sin(x) \approx x - x^3/6$ $e^x \approx 1 + x + x^2/2 + x^3/6 + x^4/24$

$f(x) \approx [x - x^3/6 - (1 + x + x^2/2 + x^3/6 + x^4/24) + 1] / x^2 = -0.5 - x/3 - x^2/24$

b) Show that this polynomial is unstable for fl(f(0.123))

Solution: $x = 0.123 + \xi$, where $\xi / 0.123$ is small. Given the problem where x = 0.123 gives a solution of –0.4629, now compare to the perturbed problem where $x = 0.123 + \xi$. $f(x) = -0.5 - (0.123 + \xi)/3 - (0.123 + \xi)^2/24 = -0.54163 - 0.34358\xi + O(\xi)^2$

≈ 0.5416 for all small ξ.

3. a) Apply Newton-Raphson to $f(x) = x^2 - 1/c$ to determine an iterative formula for computing 1/sqrt(c). Solution: $f(x) = x^2 - 1/c$; $f'(x) = 2x$

$p_n = p_{n-1} - (p_{n-1}^2 - 1/c) / 2p_{n-1} = (p_{n-1}^2 + 1/c) / 2p_{n-1} = 0.5(p_{n-1} + 1/cp_{n-1})$

b) For arbitrary c > 0, let p0 be the initial approximation to 1 / sqrt(c) , let {p1 , p2 , p3, ...} be the sequence of computed approximations to 1 / sqrt(c) using the iterative formula from (a), and let $e_n = p_n - 1 / \mathrm{sqrt}(c)$ for n = 0,1,2,3,... Show using algebra that $e_n = e_{n-1}^2 / 2p_{n-1}$. Also, it followed that $\lim(n = \infty)\ |e_n| / |e_{n-1}|^2 = \lim(n = \infty)\ 1/2p_{n-1} = \sqrt{c} /2$

Solution: $e_n = p_n - 1/\mathrm{sqrt}(c) = [p_{n-1}^2 + 1/c]/2p_{n-1} - 1/\mathrm{sqrt}(c) = [\mathrm{sqrt}(c)p_{n-1}^2 + 1/c - 2p_{n-1}] / 2p_{n-1}\mathrm{sqrt}(c) = \mathrm{sqrt}(c)[p_{n-1}^2 - 2p_{n-1}/\mathrm{sqrt}(c) + 1/c] / 2p_{n-1}\mathrm{sqrt}(c) = [p_{n-1} - 1/\mathrm{sqrt}(c)]^2 / 2p_{n-1} = e_{n-1}^2 / 2p_{n-1}$

4. Determine a0, b0, d0, a1, b1, c1 and d1 so that

$S(x) = [a0 + b0x - 3x^2 + d0x^3,\ -1 \le x \le 0;\ a1 + b1x + c1x^2 + d1x^3,\ 0 \le x \le 1]$

is the natural cubic spline function such that S(-1) = 1, S(0) = 2 and S(1) = -1.

Clearly identify the 8 conditions that the unknowns must satisfy, and then solve for the 7 unknowns.

Solution: $S_0'(x) = b_0 - 6x + 3d_0x^2$ $S_1'(x) = b_1 + 2c_1x + 3d_1x^2$

$S_0''(x) = -6 + 6d_0x S_1'(x) = 2c_1 + 6d_1x$

The 8 conditions are: $S_0(-1) = 1 \Rightarrow a_0 - b_0 - 3 - d_0 = 1 \Rightarrow a_0 - b_0 - d_0 = 4$

$S_1(0) = 2 \Rightarrow a_1 = 2$

$S_1(1) = -1 \Rightarrow a_1 + b_1 + c_1 + d_1 = -1 \Rightarrow b_1 + c_1 + d_1 = -3$

$S_1(0) = S_0(0) \Rightarrow a_1 = a_0 \Rightarrow a_0 = 2$

$S_1'(0) = S_0'(0) \Rightarrow b_1 = b_0$

$S_1''(0) = S_0''(0) \Rightarrow 2c_1 = -6 \Rightarrow c_1 = -3$

$S_0''(-1) = 0 \Rightarrow -6 - 6d_0 = 0 \Rightarrow d_0 = -1$

$S_1''(1) = 0 \Rightarrow 2c_1 + 6d_1 = 0 \Rightarrow -6 + 6d_1 = 0 \Rightarrow d_1 = 1$

From the first condition, $b_0 = a_0 - d_0 - 4 = -1$

From the fifth condition, $b_1 = b_0 \Rightarrow b_1 = -1$

5. Determine the degree of precision of the quadrature formula (3h/4)[3f(h) + f(3h)] which is an approximiation to $\int_0^{3h} f(x)dx$. Show all work:

Solution: f(x) = 1 $\int_0^{3h} dx = 3h$ 3h/4[3 + 1] = 3h

f(x) = x $\int_0^{3h} x dx = 9h^2 / 2$ $3h/4[3h + 3h] = 9h^2 / 2$

$f(x) = x^2$ $\int_0^{3h} x^2\,dx = 9h^3$ $3h/4[3h^2 + 9h^2] = 9h^3$

$f(x) = x^3$ $\int_0^{3h} x^3\,dx = 81h^4 / 4$ $3h/4[3h^3 + 27h^3] = 45h^4 / 2 \neq 81h^4 / 4$

thus the degree of precision is 2

6. Consider approximating $\int_a^b f(x)\,dx$ using Romberg integration. Denote the Romberg table by

$R_{1,1}$

$R_{2,1}$ $R_{2,2}$

$R_{3,1}$ $R_{3,2}$ $R_{3,3}$ where $R_{k,1}$ = the trapezoidal rule approximation to $\int_a^b f(x)dx$ using 2^{k-1} subintervals on [a,b]. Use $R_{1,1}$ and $R_{2,1}$ and Richardson extrapolation to show that $R_{2,2}$ is equal to Simpsons Rule approximation to $\int_a^b f(x)dx$

Solution: $R_{22} = [4R_{21} - R_{11}] / 3$ or $R_{21} + [R_{21} - R_{11}] / 3$

$= 4[(b-a)(f_0 + 2f_1 + f_2)/4 - (b-a)(f_0 + f_2)/2] / 3$

$= (b-a)[2(f_0 + 2f_1 + f_2) - (f_0 + f_2)] / 6 = (b-a)[f_0 + 4f_1 + f_2] / 6$

$= h/3\,(f_0 + 4f_1 + f_2)$ with $h = (b - a) / 2$

7. Approximate $\int_{-1}^{1} \cos(x) / \mathrm{sqrt}(1 - x^2)\,dx$ using Gauss-Legendre and with n = 3. The argument for cos is in radians. **Solution:** Gauss-Legendre: (Roots, Coefficients): (0.7754967, 5/9); (0.0, 8/9); (-0.7745967, 5/9)

$f(x) = \cos(x) / \mathrm{sqrt}(1 - x^2)$

$5/9\,f(x_1) + 8/9\,f(x_2) + 5/9\,f(x_3) = 5\cos(0.7745967)/9\mathrm{sqrt}(1 - (0.7745967)^2)$

$+ 8\cos(0) / 9\mathrm{sqrt}(1 - 0^2) + 5\cos(-0.7745967)/ 9\mathrm{sqrt}(1 - (-0.7745967)^2)$

= 2.144494

8. a) Consider the initial-value problem $y'(t) = (1/t)(y^2 + y)$ and y(1) = -2. Use the Taylor method of order n = 2 with h = 0.1 to approximate y(1.1). Show all of your work and the iterative formula.

Solution: $f(t, y(t)) = (1/t)[(y(t))^2 + y(t)]$ so $y''(t) = \partial f/\partial t + \partial f/\partial y = (-1/t^2)(y^2 + y) + (1/t)(2y + 1)((y^2 + y) / t) = 2y^2(y + 1) / t^2$

Taylor method of order 2 is $w_{i+1} = w_i + hf(t_i, w_i) + h^2f'(t_i, w_i)/2$

$= w_i + h(w_i^2 + w_i)/t_i + (h^2 2w_i^2 / 2t_i^2)(w_i + 1)$

$= w_i + hw_i(w_i + 1)/t_i + (h^2w_i^2/t_i^2)(w_i + 1)$

So, $w_1 = w_0 + hw_0(w_0 + 1)/t_0 + h^2w_0^2(w_0 + 1)/t_0^2$

$= -2 + 0.1(-2)(-2 + 1) / 1 + 0.01(4)(-2 + 1) / 1^2 = -1.84$

b) Approximate y(1.1) using h = 0.1 and the following second order Runge-Kutta method: $w_{i+1} = w_i + (h/2)[f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))]$

Solution: $w_1 = w_0 + (h/2)[f(t_0, w_0) + f(t_1, w_1 + hf(t_0, w_0))]$

$= -2 + (0.1/2)[f(1, -2) + f(1.1, -2 + 0.1f(1, -2))]$

$= -2 + 0.05[(4 - 2) + f(1.1, -2 + 0.1(2))] = -2 + 0.05[2 + ((-1.8)^2 - 1.8)/1.1]$

$= -2 + 0.05(2 + 1.309090) = 1.8345$

c) Order of the local truncation error of the Runge-Kutta method is $O(h^3)$

8. For the following data, $(x_i, f(x_i))$: (-1,0); (1,1); (2,3); Suppose a function g(x) of the form $g(x) = c_0 + c_1e^{-x} + c_2e^x$ is to be determined so that g(x) interpolates the above data at the specified points x_i. Write a system of linear equations in matrix/vector form Ac = b whose solution will give the values of the unknown c_0, c_1 and c_2 that solve this interpolation. Note: Leave answer in terms of e, and do not solve the system.

Solution: $[[1\ e\ e^{-1}]\ [1\ e^{-1}\ e]\ [1\ e^{-2}\ e^2]] * [c_0\ c_1\ c_2] = [0\ 1\ 3]$

Problem

Suppose that the absolute errors in three consecutive approximations with some iterative method are $e_n = 0.07$; $e_{n+1} = 0.013$; $e_{n+2} = 0.00085$ Use the definition of convergence to estimate the order of convergence of the iterative method.

Solution

Definition of the order of convergence $\propto$: $\lim_{n \Rightarrow \infty} e_{n+1}/ e_n^{\propto} = k$; if n is sufficiently large then, $e_{n+1}/ e_n^{\propto} = k = e_{n+2}/ e_{n+1}^{\propto} \Rightarrow e_{n+1}/ e_{n+2} = k = (e_n/ e_{n+1})^{\propto}$; solve for $\propto$

Problem

c) Compute two approximations computed to $\int x/(1+x^2)\,dx$, using the noncomposite Simpsons rule(with h=0.1) and using the composite Simpsons rule (with h=0,05).

d) The truncation error term for the two approximations computed in a) to obtain an $O(h^4)$ approximation to the value of the given definite integral.

Solution

c) Simpsons rule(h=0.1) $I_1 = h/3\,[f(0) + 4f(0.1) + f(0.2)] = 0.01961158$: Composite Simpsons rule(h=0.05) $I_2 = h/3\,[f(0) + 4f(0.05) + 2f(0.1) + 4f(0.15) + f(0.2)] = 0.0196143$

d) width h= 0.1

(1) $I = I_1 + kh^4 + O(h^6)$

(2) $I = I_2 + k(h/2)^4 + O(h^6)$

16*(2) - (1) solve for $I \Rightarrow I = I_2 + (I_2 - I_1)/15 + O(h^6)$

$\therefore I = 0.019103533$

Problem

The open-newton cotes quadrature formula(for n=1) that approximates $\int x g(x)\,dx$ is $3h/2\,(g(x_o)+ g(x_1))$ where $h = (x_o - x_1)/3$ and $x_i = x + (i+1)h$ for i= 0,1,2,…; recall that Euler's method can be derived by intergrating this differential equation $y'(t) = f(t,y(t))$ over $[t_I, t_{I+1}]$ and using a simple "rectangular" rule to approximate $\int f(t,y(t))\,dt$. Derive a formula for approximating th solution to $y'(t) = f(t,y(t))$ by intergrating this differential equation over $[t_I, t_{I+1}]$ and approximating $\int f(t,y(t))\,dt$ by the above open newton cotes quadrature formula for n=1.

Solution

$\int y'(t) = \int f(t,y(t)) \therefore y(t_{I+3}) = y(t_I) + \int f(t,y(t)) \approx y(t_I) + 3h/2\,(f(t_{I+1},y(t_{I+1}))+ f(t_{I+2},y(t_{I+2})))$ which suggests $\omega_{I+3} = \omega_i + 3h/2\,(f(t_{I+1},\omega_{I+1}))+ f(t_{I+2},\omega_{I+2})))$

ezplot('0.5*exp(x/3)-sin(x)', [-4, 5])
fplot('3*x^3-x-1', [-1 2], '-')

fzero to compute the zeros of f(x):
fzero(' 0.5*exp(x/3)-sin(x)', [-5, -3])
Change bounds for each interval a zero is located.

Polynomial Operations:
The coefficients of a polynomial are stored in a vector *p*
(**starting with the coefficient of the highest power of *x***).
$p(x) = x^3 - 2x - 5$
p = [1 0 -2 -5];
r = roots(p)
Will compute and store all of the roots of the polynomial in r.
Alternate form: r = roots([1 0 -2 -5])
If r is any vector, p = poly(r), will result in p being a vector whose entries are the coefficients of a polynomial $p(x)$ that has as its roots the entries of r.
Example:
r = [1 2 3 4];
p = poly(r)
Results in p = [1 -10 35 -50 24],
$p(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$
The function *polyval* evaluates a polynomial at a specified value. If p = [1 -10 35 -50 24], then polyval(p, -1) gives the value of $p(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$ at $x = -1$, namely 120. MATLAB uses HORNER'S ALGORITHM for *polyval*.

The MATLAB function INTERP1 can be used to do linear and cubic polynomial interpolation. If X denotes a vector of values, Y = F(X) a set of corresponding function values, then z = interp1(X, Y, x) or z = interp1(X, Y, x, 'linear'). z = interp1(X, Y, x, 'cubic') determines z based on a cubic polynomial approximation.

For the cubic spline with clamped boundary conditions, the data to be interpolated should be stored in vectors, X and Y, where Y has 2 more entries than X and the first and last entries of Y are the two boundary conditions. If *S(x)* denotes the cubic spline interpolant, and *z* is a given number, then the value of *S(z)* can be computed by entering spline(X, Y, z). To determine the coefficients of the spline, first determine the pp (piecewise polynomial) form of the spline by entering pp = spline(X, Y). Then enter [breaks, coefs] = unmkpp(pp).
breaks: a vector of the knots (or nodes) of the spline,
coefs: an array, the i-th row of which contains the coefficients of the i-th spline

Quad - Numerically evaluate integral, adaptive Simpson quadrature. Use array operators .*, ./ and .^
using tol = 10^{-5}
$$\int_{0.1}^{2} \sin(1/x)\,dx$$
[Q, fnc] = quad (@f, 0.1, 2, 10^-5)
function y=f(x)
y = sin(1./x);

Ode45: is variable stepsize Runge-Kutta method that uses two Runge-Kutta formulas of orders 4 and 5. Solve non-stiff differential equations, medium order method.

$y'(t) = -y(t) - 5e^{-t}\sin(5t)$, $y(0) = 1$ on [0,3]
[t, y] = ode45('f', [0 3], 1)
function z=f(t,y)
z = -y-5*exp(-t)*sin(5*t);

Linspace: Linearly spaced vector.

LINSPACE(X1, X2) generates a row vector of 100 linearly equally spaced points between X1 and X2.
LINSPACE(X1, X2, N) generates N points between X1 and X2. For N < 2, LINSPACE returns X2.

Gaussian Elimination Algorithm

(forward elimination)
For i=1,2,…n-1 do
  For j=i+1,i+2,…,n do
    mult ← a_{ji}/a_{ii}
    for k=i+1,i+2,…,n do
      a_{jk} ← a_{jk}-mult*a_{ik}
    b_j ← b_j-mult*b_i

(back substitution)
x_n ← b_n/a_{nn}
for i=n-1,n-2,…,1 do
  x_i ← [b_i-sum($a_{ij}x_j$, from j=i+1 to n)]/a_{ii}

Gaussian Elimination with partial pivoting
(forward elimination)
for i=1,2,…,n-1 do
  (find the largest pivot)
  p←i
  for k=i+1,i+2,…,n do
    if $|a_{k,i}|>|a_{p,i}|$ then
      p←k
  if p!=i then
    for k=i,i+1,…,n do
      temp←$a_{i,k}$
      $a_{i,k}$←$a_{p,k}$
      $a_{p,k}$←temp
    temp←b_i
    b_i←b_p
    b_p←temp
  for j=i+1,i+2,…,n do
    mult←$a_{j,i}/a_{ii}$
    for k=i+1,i+2,…,n do
      $a_{j,k}$←$a_{j,k}$-mult*$a_{i,k}$
    b_j←b_j-mult*b_i
(back-substitution)
x_n←$b_n/a_{n,n}$
for i=n-1,n-2,…,1 do
  x_i←[b_i-sum($a_{ij}x_j$, from j=i+1 to n)]/a_{ii}

Composite Trapezoidal Rule: Input – upper and lower limits of the integral a and b, max # of iterations maxiter, and tolerance tol.

function **trap**(a, b, maxiter, tol)
```
m = 1;
x = linspace(a, b, m+1);
y = f(x);
approx = trapz(x, y);
disp(' m integral approximation');
fprintf('  %5.0f %16.10f \n ', m, approx);
for i = 1 : maxiter
m = m*2 ;
oldapprox = approx ;
x = linspace ( a , b , m+1 ) ;
y = f(x);
approx = trapz(x, y);
fprintf(' %5.0f %16.10f \n ', m, approx);
if abs( 1-oldapprox/ approx) < tol
return
end
end
fprintf('The iteration did not converge in %g', maxiter), fprintf(' iterations.')
```

Secant: To find the zero of the function. Input – initial approx. p0, p1, tolerance tol, and max # of iterations N.
function p = **secant**(p0, p1, N, tol)
```
i=2;
q0=f(p0);
q1=f(p1);
while i<=N
   p=p1-q1*(p1-p0)/(q1-q0);
   if abs(p-p1)<tol
      return;
   end
   i=i+1;
   p0=p1;
   q0=q1
   p1=p;
   q1=f(p);
end
fprintf('failed to converge in %g',Nzero),fprintf(' iterations\n')
```

Bisection Method
function p = **bisection**(p0, p1, N, tol)
```
i=1;
FA=f(a);
while i<=N
   p=a+(b-a)/2;
   FP=f(p);
   if ((FP= =0)|((b-a)/2<TOL))
      return;
   end
   i=i+1;
   if FA*FP>0
      a=p;
      FA=FP;
   else
      b=p;
   end
end
fprintf('failed to converge in %g',Nzero),fprintf(' iterations\n')
```

Use ***polyval*** to evaluate $q(x)$ at the first zero of $q(x)$ computed by MATLAB in (a). This should give a result very close to 0.

r = [1 1 1 1 1 1 1 1];
p = **poly**(r)
roots_of_p = **roots**(p)
q = p + [0 0 0 0.001 0 0 0 0 0]
qr = roots(q)

polyval(q,qr(1))

Use INTERP1 to approximate z = P(1.3), where P(x) is the piecewise linear interpolating polynomial to y = sin x at the 11 equally-spaced points

```
for i = 1:11
   x(i)=(i-1)*pi/20;
   y(i)=sin(x(i));
end
```

z = **interp1**(x, y, 1.3, 'linear');
fprintf(' z = %18.10f\n',z);
fprintf(' sin(1.3) = %18.10f\n',sin(1.3));
fprintf(' abs(sin(1.3)-z) = %18.10f\n',abs(sin(1.3)-z));

Polynewton computes one zero of a polynomial.
Input variables:
n the polynomial degree
a a vector of the coefficients of the polynomial $P(x)$
pzero the initial approximation to a zero
Nzero the maximum number of iterations allowed
tol relative error tolerance used to test for convergence
and the output variables are
p the final computed approximation to a zero of $P(x)$
b the final vector of values *b* computed by Horner's algorithm, from which the deflated polynomial can be obtained.

polynewton(4, [-3.3 -1 0 2 5], 1.3, 20, 1e-8);
Will output to the screen each computed approximation to the zero.
[x, y] = polynewton(4, [-3.3 -1 0 2 5], 1.3, 20, 1e-8);
Will output to the screen each computed approximation to a zero, will store the final computed approximation in the variable *x*, and will store the final computed vector of values *b* from Horner's algorithm in the vector y.

Assignment #3
plot a graph of the function $f(x) = \frac{e^{x/3}}{2} - \sin x$ $f(x)$ in [−4, 5].

>> ezplot('0.5*exp(x/3)-sin(x)',[-4,5])

fzero to compute the 3 zeros of
>> fzero('0.5*exp(x/3)-sin(x)',[-5,-3])
ans = -3.3083

h=fzero('10*(0.5*pi*1^2-1^2*asin(x/1)-x*sqrt(1^2-x^2))-12.4',[0,1])
h = 0.1662; >> r=1; >> x=r-h; x = 0.8338

Newton.m
To find a zero of the function. Input – Initial approx. pzero, maximum interations Nzero, tolerance tol.
function p = **newton**(pzero,Nzero,tol)
```
i=1;
while i <= Nzero
   p = pzero-f(pzero)/fp(pzero);
   fprintf('i = %g',i),fprintf(' approximation = %18.10f\n',p)
   if abs(1-pzero/p)<tol
      return
   end
   i=i+1;
   pzero=p;
end
fprintf('failed to converge in %g',Nzero),fprintf(' iterations\n')
```
f.m
function y=f(x)
y=32*(x-sin(x))-35;
fp.m
function y=f(x)
y=32*(1-cos(x));

Multiplicity of the roots.

```
function test_mult(x)
root = newton (pi/2, 40, 1e-8);
fprintf('\nroot used = %18.10f\n',root);
for i = 0:x
   fprintf('m = %g',i),fprintf(' p = %18.10f\n',f_diff(root,i));
end
```

function [p,b] = **polynewton**(n, a, pzero, Nzero, tol)
```
i=1;
while i<=Nzero
   [b, c] = horner(pzero, n, a);
   p = pzero - b(1)/c(2);
   fprintf('i = %g',i),fprintf(' approximation = %18.10f\n',p)
   if abs(1-pzero/p)<tol
      return
   end
   i=i+1;
   pzero=p;
End
fprintf('failed to converge in %g',Nzero),fprintf(' iterations\n')
```

Newtons method to find reciprocal
$$f(x) = \frac{1}{x} - R$$
$$f'(x) = \frac{1}{-x^2}$$
$$p = p_0 - \frac{f(p_0)}{f'(p_0)}$$
$$p = p_0 - \frac{\frac{1}{p_0} - R}{\frac{1}{-p_0^2}}$$
$$p = p_0 + p_0 - R \cdot p_0^2$$
$$p = 2 \cdot p_0 - R \cdot p_0^2$$

Matlab version of Newtons method for computing **1/R**

function p = **reciprocal**(R,pzero,Nzero,tol)
```
i=1;
while i<=Nzero
   p=2*pzero-R*pzero*pzero;
   fprintf('i = %g',i),fprintf(' approximation = %18.10f\n',p)
   y(i)=p;
   if abs(1-pzero/p)<tol
      return
   end
   i=i+1;
   pzero=p;
end
fprintf('failed to converge in %g',Nzero),fprintf(' iterations\n')
```

Horner: to evaluate the polynomial. Input – location x0, degree n, coefficients a. Outputs: b = P(x0), c = P'(x0)

function [b, c] = **horner**(x0, n, a)
```
b(n+1)=a(n+1);
c(n+1)=a(n+1);
for j = n: -1: 2
b(j) = a(j) + b(j+1)*x0;
c(j) = b(j) + c(j+1)*x0;
end
b(1) = a(1) + b(2)*x0;
```