Outline

1. Numerical Integration
2. Numerical Differentiation
3. Richardson Extrapolation

Main Ideas

- ***Quadrature*** based on polynomial interpolation:
  - Methods:
    - Method of undetermined coefficients (e.g., Adams-Bashforth)
    - Lagrangian interpolation
  - Rules:
    - Midpoint, Trapezoidal, Simpson, Newton-Cotes
    - Gaussian Quadrature
- ***Quadrature*** based on piecewise polynomial interpolation
  - Composite trapezoidal rule
  - Composite Simpson
  - Richardson extrapolation
- ***Differentiation***
  - Taylor series / Richardson extrapolation
  - Derivatives of Lagrange polynomials
  - Derivative matrices

Quadrature Examples

- $\mathcal{I} \;=\; \int_a^b e^{\cos x}\, dx$

Differential equations:

$$u(x) \;=\; \sum_{j=1}^{n} u_j \phi_j(x) \;\in\; X^N$$

Find $u(x)$ such that

$$r(x) \;=\; \frac{du}{dx} \;-\; f(x) \;\perp\; X^N$$

Integration

- For $f: \mathbb{R} \to \mathbb{R}$, definite integral over interval $[a, b]$

$$I(f) = \int_a^b f(x)\,dx$$

is defined by limit of Riemann sums

$$R_n = \sum_{i=1}^{n} (x_{i+1} - x_i)\, f(\xi_i)$$

- Riemann integral exists provided integrand f is bounded and continuous almost everywhere
- Absolute condition number of integration with respect to perturbations in integrand is $b - a$
- Integration is inherently well-conditioned because of its smoothing effect

Numerical Quadrature

- *Quadrature rule* is weighted sum of finite number of sample values of integrand function
- To obtain desired level of accuracy at low cost,
  - How should sample points be chosen?
  - How should their contributions be weighted?
- Computational work is measured by number of evaluations of integrand function required

Quadrature Rules

- An n-point quadrature rule has form

$$Q_n(f) = \sum_{i=1}^{n} w_i\, f(x_i)$$

- Points x_i are called *nodes* or *abscissas*
- Multipliers w_i are called *weights*
- Quadrature rule is
  - *open* if $a < x_1$ and $x_n < b$
  - *closed* if $a = x_1$ and $x_n = b$
- Can also have $(x_1\, x_2 \ldots x_n) \leq a < b$ (Adams-Bashforth timesteppers)

Quadrature Rules, continued

- Quadrature rules are based on polynomial interpolation
- Integrand function f is sampled at finite set of points
- Polynomial interpolating those points is determined
- Integral of interpolant is taken as estimate for integral of original function
- In practice, interpolating polynomial is not determined explicitly but used to determine weights corresponding to nodes
- If Lagrange is interpolation used, then weights are given by

$$w_i = \int_a^b \ell_i(x), \quad i = 1, \dots, n$$

Method of Undetermined Coefficients

- Alternative derivation of quadrature rule uses *method of undetermined coefficients*
- To derive n-point rule on interval $[a, b]$, take nodes $x_1, \ldots, x_n$ as given and consider weights $w_1, \ldots, w_n$ as coefficients to be determined
- Force quadrature rule to integrate first n polynomial basis functions exactly, and by linearity, it will then integrate any polynomial of degree $n - 1$ exactly
- Thus we obtain system of *moment equations* that determines weights for quadrature rule

Quadrature Overview

$$I(f) \;:=\; \int_a^b f(t)\,dt \;\approx\; \sum_{i=1}^{n} w_i\, f_i, \;=:\; Q_n(f) \qquad f_i \;:=\; f(t_i)$$

- Idea is to minimize the number of function evaluations.
- *Small n is good.*
- Several strategies:
  - global rules
  - composite rules
  - composite rules + extrapolation
  - adaptive rules

Global (Interpolatory) Quadrature Rules

- Generally, approximate $f(t)$ by polynomial interpolant, $p(t)$.

$$f(t) \;\approx\; p(t) \;=\; \sum_{i=1}^{n} l_i(t)\, f_i$$

$$I(f) \;=\; \int_a^b f(t)\, dt \;\approx\; \int_a^b p(t)\, dt \;=:\; Q_n(f)$$

$$Q_n(f) \;=\; \int_a^b \left(\sum_{i=1}^{n} l_i(t)\, f_i \right) dt \;=\; \sum_{i=1}^{n} \left(\int_a^b l_i(t)\, dt \right) f_i \;=\; \sum_{i=1}^{n} w_i\, f_i.$$

$$\boxed{w_i \;=\; \int_a^b l_i(t)\, dt}$$

- We will see two types of global (interpolatory) rules:
  - Newton-Cotes — interpolatory on uniformly spaced nodes.
  - Gauss rules — interpolatory on optimally chosen point sets.

Examples

- Midpoint rule: $l_i(t) = 1$

$$w_1 \;=\; \int_a^b l_1(t)\,dt = \int_a^b 1\,dt = (b-a)$$

- Trapezoidal rule:

$$l_1(t) = \frac{t-b}{a-b}, \qquad\qquad l_2(t) = \frac{t-a}{b-a}$$

$$w_1 = \int_a^b l_1(t)\,dt = \;\tfrac{1}{2}(b-a)\; = \int_a^b l_2(t)\,dt = w_2$$

- Simpson's rule:

$$w_1 = \int_a^b l_1(t)\,dt = \;\tfrac{b-a}{6}\; = \int_a^b l_3(t)\,dt = w_3$$

$$w_2 = \int_a^b l_2(t)\,dt \;=\; \int_a^b \left(\frac{t-a}{b-a}\right)\left(\frac{t-b}{a-b}\right)dt \;=\; \frac{2}{3}(b-a)$$

- These are the Newton-Cotes quadrature rules for n=1, 2, and 3, respectively.
  - The midpoint rule is *open*.
  - Trapezoid and Simpson's rules are *closed*.

Finding Weights: Method of Undetermined Coefficients

Example 1: Find w_i for $[a,b] = [1,2]$, $n = 3$.

- First approach: $f \;=\; 1,\; t,\; t^2$.

$$
\begin{aligned}
I(1) &= \sum_{i=1}^{3} w_i \cdot 1 \;=\; 1 \\
I(t) &= \sum_{i=1}^{3} w_i \cdot t_i \;=\; \frac{1}{2} t^2 \Big|_1^2 \\
I(t^2) &= \sum_{i=1}^{3} w_i \cdot t_i^2 \;=\; \frac{1}{3} t^3 \Big|_1^2
\end{aligned}
$$

Results in 3×3 matrix for the w_is.

Finding Weights: Method of Undetermined Coefficients

- Second approach: Choose f so that some of the coefficients multiplying the w_is vanish.

$$\begin{aligned} I_1 &= w_1(1-\frac{3}{2})(1-2) &= \int_1^2 (t-\frac{3}{2})(t-2)\,dt \\ I_2 &= w_2(\frac{3}{2}-1)(\frac{3}{2}-2) &= \int_1^2 (t-1)(t-2)\,dt \\ I_3 &= w_3(2-1)(2-\frac{3}{2}) &= \int_1^2 (t-1)(t-\frac{3}{2})\,dt \end{aligned}$$

Corresponds to the Lagrange interpolant approach.

Method of Undetermined Coefficients

Example 2: Find w_i for $[a,b] = [0,1]$, $n = 3$, but using $f_i = f(t_i) = f(i)$, with i = -2, -1, and 0. (The t_i's are outside the interval (a,b).)

- Result should be exact for $f(t) \in \mathbb{P}_0$, $\mathbb{P}_1$, and $\mathbb{P}_2$.
- Take f=1, f=t, and f=t^2.

$$\begin{aligned} \sum w_i &= 1 = \int_0^1 1\,dt \\ -2w_{-2} - w_{-1} &= \frac{1}{2} = \int_0^1 t\,dt \\ 4w_{-2} + w_{-1} &= \frac{1}{3} = \int_0^1 t^2\,dt \end{aligned}$$

- Find

$$w_{-2} = \frac{5}{12} \qquad w_{-1} = -\frac{16}{12} \qquad w_0 = \frac{23}{12}.$$

- This example is useful for finding integration coefficients for explicit time-stepping methods that will be seen in later chapters.

Example: Undetermined Coefficients

- Derive 3-point rule $Q_3(f) = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3)$ on interval $[a, b]$ using monomial basis

- Take $x_1 = a$, $x_2 = (a + b)/2$, and $x_3 = b$ as nodes

- First three monomials are 1, x, and x^2

- Resulting system of moment equations is

$$\begin{aligned}
w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1 &= \int_a^b 1\,dx = x|_a^b = b - a \\
w_1 \cdot a + w_2 \cdot (a + b)/2 + w_3 \cdot b &= \int_a^b x\,dx = (x^2/2)|_a^b = (b^2 - a^2)/2 \\
w_1 \cdot a^2 + w_2 \cdot ((a + b)/2)^2 + w_3 \cdot b^2 &= \int_a^b x^2\,dx = (x^3/3)|_a^b = (b^3 - a^3)/3
\end{aligned}$$

Example, continued

- In matrix form, linear system is

$$\begin{bmatrix} 1 & 1 & 1 \\ a & (a+b)/2 & b \\ a^2 & ((a+b)/2)^2 & b^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} b-a \\ (b^2-a^2)/2 \\ (b^3-a^3)/3 \end{bmatrix}$$

- Solving system by Gaussian elimination, we obtain weights

$$w_1 = \frac{b-a}{6}, \qquad w_2 = \frac{2(b-a)}{3}, \qquad w_3 = \frac{b-a}{6}$$

which is known as *Simpson's rule*

Method of Undetermined Coefficients

- More generally, for any n and choice of nodes $x_1, \dots, x_n$, *Vandermonde system*

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} b-a \\ (b^2-a^2)/2 \\ \vdots \\ (b^n-a^n)/n \end{bmatrix}$$

  determines weights $w_1, \dots, w_n$

Stability of Quadrature Rules

- Absolute condition number of quadrature rule is sum of magnitudes of weights,

$$\sum_{i=1}^{n} |w_i|$$

- If weights are all nonnegative, then absolute condition number of quadrature rule is $b - a$, same as that of underlying integral, so rule is stable

- If any weights are negative, then absolute condition number can be much larger, and rule can be unstable

Conditioning

- Absolute condition number of integration:

$$
\begin{aligned}
I(f) &= \int_a^b f(t)\,dt \\
I(\hat{f}) &= \int_a^b \hat{f}(t)\,dt \\
\left|I(f) - I(\hat{f})\right| &= \left|\int_a^b (f - \hat{f})\,dt\right| \;\leq\; |b-a| \; ||f - \hat{f}||_\infty
\end{aligned}
$$

- Absolute condition number is $|b-a|$.

Conditioning

- Absolute condition number of quadrature:

$$\left|Q_n(f) - Q_n(\hat{f})\right| \;\leq\; \left|\sum_{i=1}^{n} w_i \left(f_i - \hat{f}_i\right)\right| \;\leq\; \left|\sum_{i=1}^{n} w_i\right| \max_i \left|f_i - \hat{f}_i\right|$$

$$\leq\; \left|\sum_{i=1}^{n} w_i\right| \, ||f - \hat{f}||_\infty$$

$$C \;=\; \sum_{i=1}^{n} |w_i|$$

- If $Q_n(f)$ is interpolatory, then $\sum w_i \;=\; (b-a)$:

$$Q_n(1) \;=\; \sum_{i=1}^{n} w_i \cdot 1 \;\equiv\; \int_a^b 1\, dt \;=\; (b-a).$$

- If $w_i \geq 0$, then $C \;=\; (b-a)$.
- Otherwise, $C > (b-a)$ and can be arbitrarily large as $n \longrightarrow \infty$.

Newton-Cotes Quadrature

Newton-Cotes quadrature rules use equally spaced nodes in interval $[a, b]$

- *Midpoint rule*

$$M(f) = (b-a)f\left(\frac{a+b}{2}\right)$$

- *Trapezoid rule*

$$T(f) = \frac{b-a}{2}(f(a)+f(b))$$

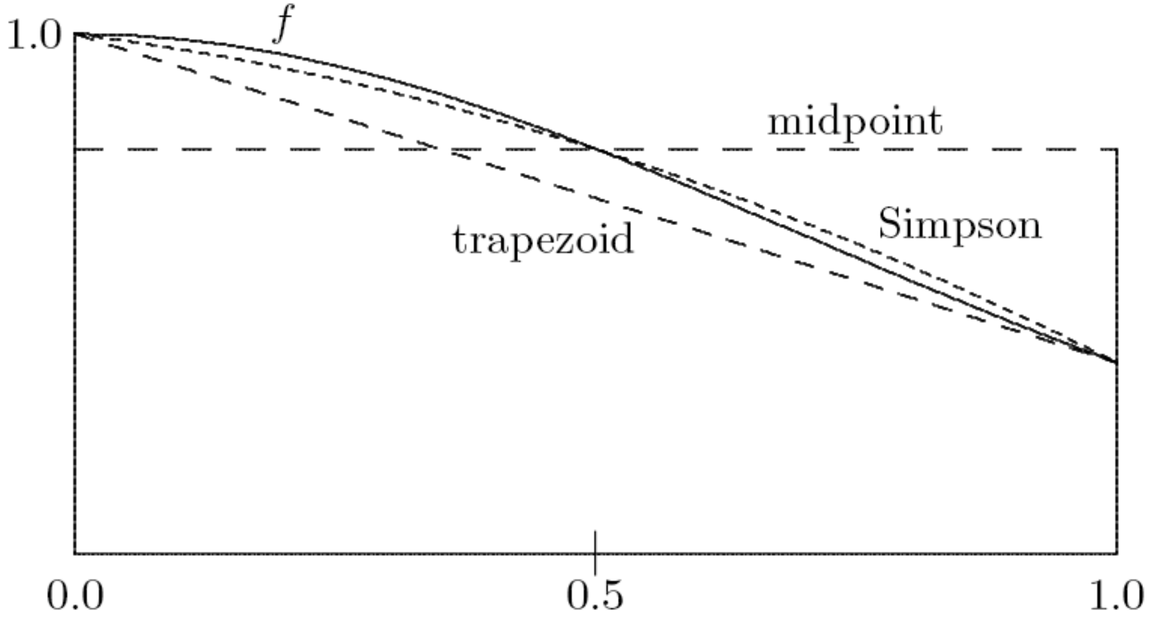
- *Simpson's rule*

$$S(f) = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$

Example: Newton-Cotes Quadrature

Approximate integral $I(f) = \int_0^1 \exp(-x^2)\,dx \approx 0.746824$

$$\begin{aligned} M(f) &= (1-0)\exp(-1/4) \approx 0.778801 \\ T(f) &= (1/2)[\exp(0) + \exp(-1)] \approx 0.683940 \\ S(f) &= (1/6)[\exp(0) + 4\exp(-1/4) + \exp(-1)] \approx 0.747180 \end{aligned}$$



< interactive example >

Error Estimation

- Expanding integrand f in Taylor series about midpoint $m = (a+b)/2$ of interval $[a, b]$,

$$\begin{aligned} f(x) &= f(m) + f'(m)(x-m) + \frac{f''(m)}{2}(x-m)^2 \\ &\quad + \frac{f'''(m)}{6}(x-m)^3 + \frac{f^{(4)}(m)}{24}(x-m)^4 + \cdots \end{aligned}$$

- Integrating from a to b, odd-order terms drop out, yielding

$$\begin{aligned} I(f) &= f(m)(b-a) + \frac{f''(m)}{24}(b-a)^3 + \frac{f^{(4)}(m)}{1920}(b-a)^5 + \cdots \\ &= M(f) + E(f) + F(f) + \cdots \end{aligned}$$

where $E(f)$ and $F(f)$ represent first two terms in error expansion for midpoint rule

Error Estimation, continued

- If we substitute $x = a$ and $x = b$ into Taylor series, add two series together, observe once again that odd-order terms drop out, solve for $f(m)$, and substitute into midpoint rule, we obtain

$$I(f) = T(f) - 2E(f) - 4F(f) - \cdots$$

- Thus, provided length of interval is sufficiently small and $f^{(4)}$ is well behaved, midpoint rule is about twice as accurate as trapezoid rule

- Halving length of interval decreases error in either rule by factor of about $1/8$

Error Estimation, continued

- Difference between midpoint and trapezoid rules provides estimate for error in either of them

$$T(f) - M(f) = 3E(f) + 5F(f) + \cdots$$

so

$$E(f) \approx \frac{T(f) - M(f)}{3}$$

- Weighted combination of midpoint and trapezoid rules eliminates $E(f)$ term from error expansion

$$\begin{aligned} I(f) &= \frac{2}{3}M(f) + \frac{1}{3}T(f) - \frac{2}{3}F(f) + \cdots \\ &= S(f) - \frac{2}{3}F(f) + \cdots \end{aligned}$$

which gives alternate derivation for Simpson's rule and estimate for its error

Example: Error Estimation

- We illustrate error estimation by computing approximate value for integral $\int_0^1 x^2\,dx = 1/3$

$$\begin{aligned} M(f) &= (1-0)(1/2)^2 = 1/4 \\ T(f) &= \frac{1-0}{2}(0^2+1^2) = 1/2 \\ E(f) &\approx (T(f)-M(f))/3 = (1/4)/3 = 1/12 \end{aligned}$$

- Error in $M(f)$ is about $1/12$, error in $T(f)$ is about $-1/6$

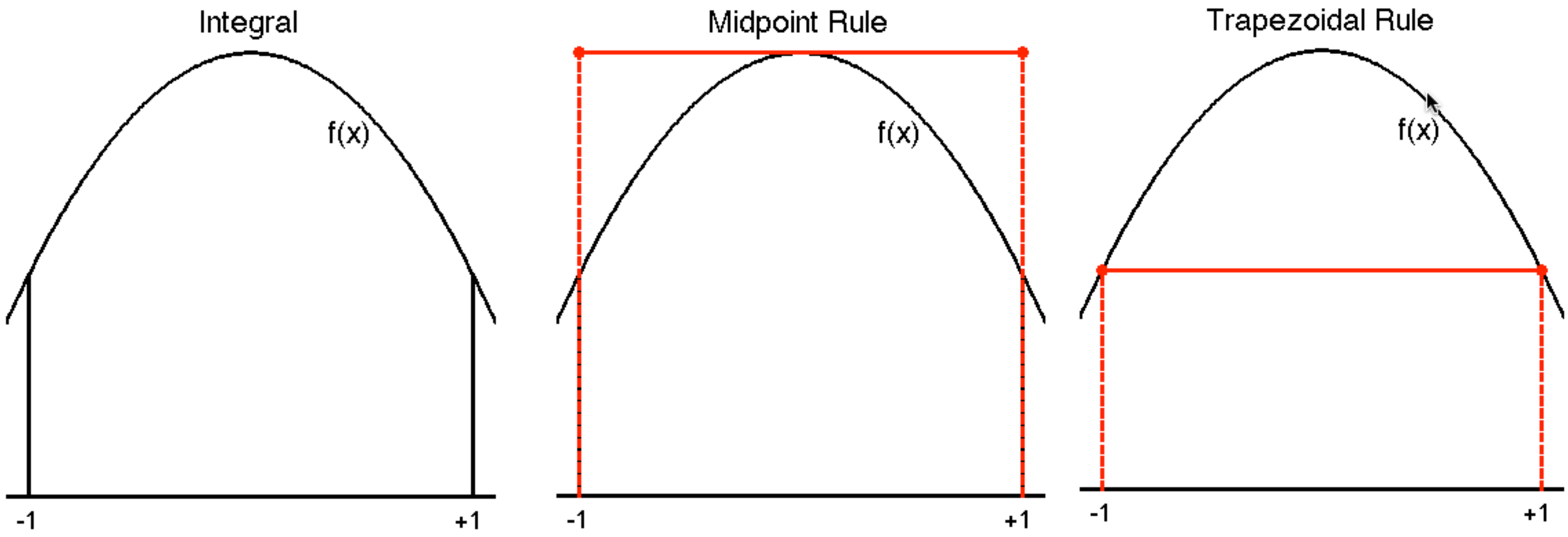
- Also,

$$S(f) = (2/3)M(f)+(1/3)T(f) = (2/3)(1/4)+(1/3)(1/2) = 1/3$$

which is exact for this integral, as expected

Example

$$
\begin{aligned}
f(x) &= 2 - x^2 \\
I(f) &= \int_{-1}^{1} f(x)\,dx = 2x - \left.\frac{x^3}{3}\right|_{-1}^{1} = 3\frac{1}{3}. \\
M(f) &= 2 \cdot f(0) = 2 \cdot 2 = 4. \qquad (|\text{error}|{=}2/3) \\
T(f) &= 2 \cdot \frac{f(-1)+f(1)}{2} = 2. \qquad (|\text{error}|{=}4/3)
\end{aligned}
$$



- Error for *midpoint rule* is generally ½ that of *trapezoidal rule.*

Accuracy of Quadrature Rules

- Quadrature rule is of *degree* d if it is exact for every polynomial of degree d, but not exact for some polynomial of degree $d+1$
- By construction, n-point interpolatory quadrature rule is of degree at least $n-1$
- Rough error bound

  $$|I(f) - Q_n(f)| \le \tfrac{1}{4}\, h^{n+1}\, \|f^{(n)}\|_\infty$$

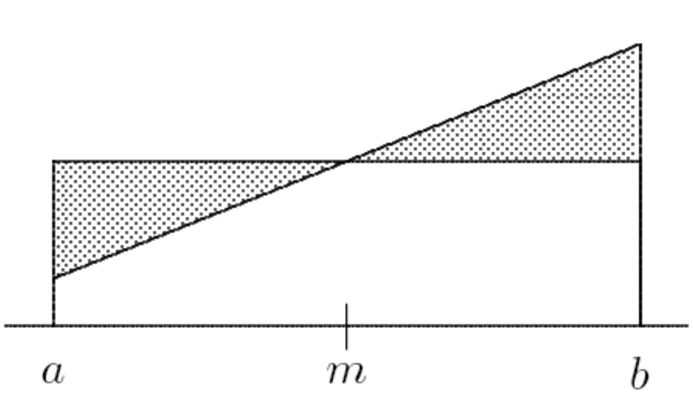
  where $h = \max\{x_{i+1} - x_i \,:\; i = 1, \ldots, n-1\}$, shows that $Q_n(f) \to I(f)$ as $n \to \infty$, provided $f^{(n)}$ remains well behaved
- Higher accuracy can be obtained by increasing n or by decreasing h

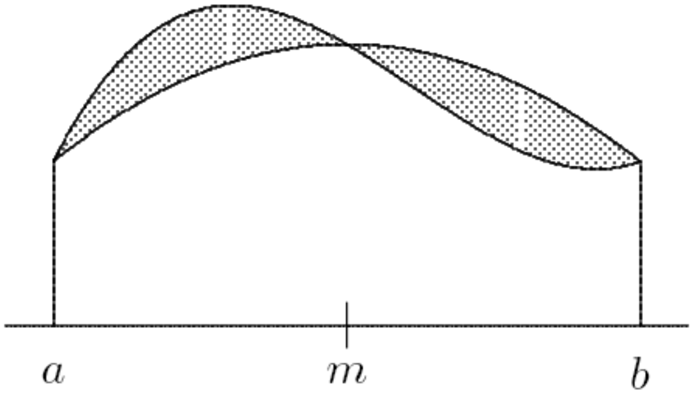
Accuracy of Newton-Cotes Quadrature

- Since n-point Newton-Cotes rule is based on polynomial interpolant of degree $n-1$, we expect rule to have degree $n-1$
- Thus, we expect midpoint rule to have degree 0, trapezoid rule degree 1, Simpson's rule degree 2, etc.
- From Taylor series expansion, error for midpoint rule depends on second and higher derivatives of integrand, which vanish for linear as well as constant polynomials
- So midpoint rule integrates linear polynomials exactly, hence its degree is 1 rather than 0
- Similarly, error for Simpson's rule depends on fourth and higher derivatives, which vanish for cubics as well as quadratic polynomials, so Simpson's rule is of degree 3

Accuracy of Newton-Cotes Quadrature

- In general, odd-order Newton-Cotes rule gains extra degree beyond that of polynomial interpolant on which it is based
- n-point Newton-Cotes rule is of degree $n - 1$ if n is even, but of degree n if n is odd
- This phenomenon is due to *cancellation* of positive and negative errors





< interactive example >

Newton-Cotes Weights

$n\backslash j$	1	2	3	4	5	6
2	$\frac{1}{2}$	$\frac{1}{2}$				
3	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{1}{3}$			
4	$\frac{3}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	$\frac{3}{8}$		
5	$\frac{14}{45}$	$\frac{64}{45}$	$\frac{8}{15}$	$\frac{64}{45}$	$\frac{14}{45}$	
6	$\frac{95}{288}$	$\frac{125}{96}$	$\frac{125}{144}$	$\frac{125}{144}$	$\frac{125}{96}$	$\frac{95}{288}$

Drawbacks of Newton-Cotes Rules

- Newton-Cotes quadrature rules are simple and often effective, but they have drawbacks

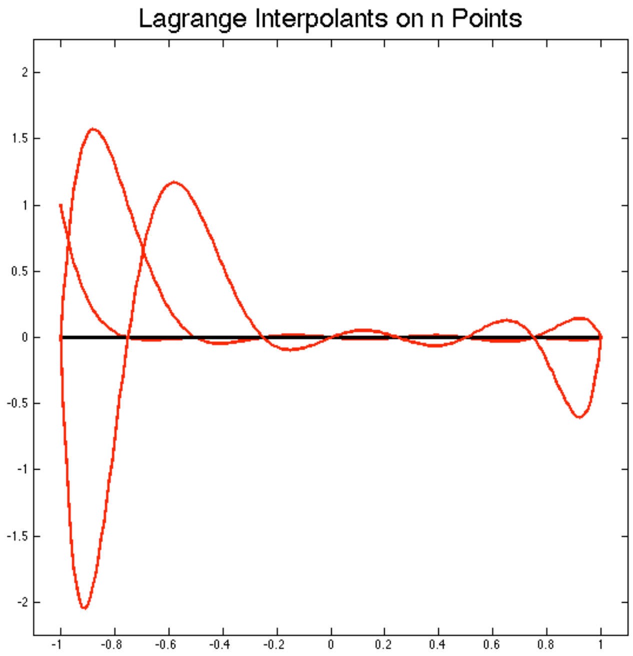
- Using large number of equally spaced nodes may incur erratic behavior associated with high-degree polynomial interpolation (e.g., weights may be negative)

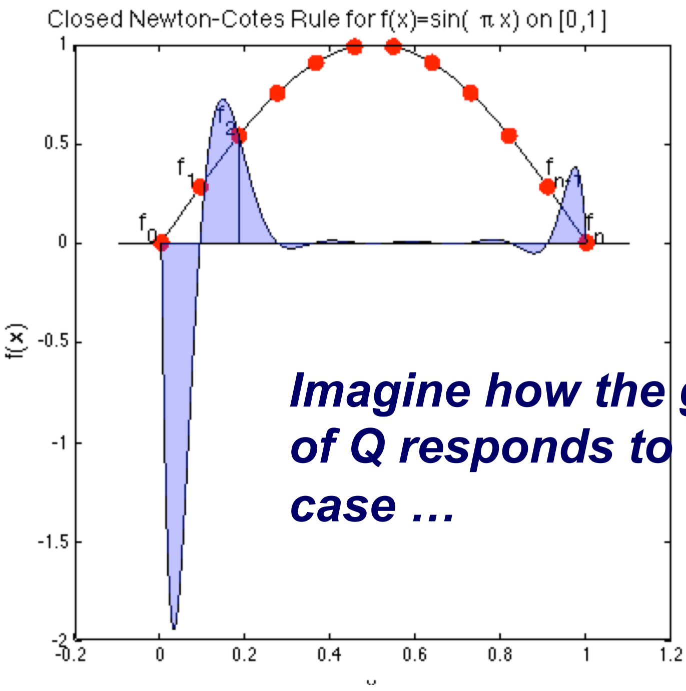
- Indeed, every n-point Newton-Cotes rule with $n \geq 11$ has at least one negative weight, and $\sum_{i=1}^{n} |w_i| \to \infty$ as $n \to \infty$, so Newton-Cotes rules become arbitrarily ill-conditioned

- Newton-Cotes rules are not of highest degree possible for number of nodes used

Newton-Cotes Formulae: What Could Go Wrong?

- Demo: newton_cotes.m and newton_cotes2.m
- Newton-Cotes formulae are interpolatory.
- For high n, Lagrange interpolants through uniform points are ill-conditioned.
- In quadrature, this conditioning is manifest through negative quadrature weights (bad).



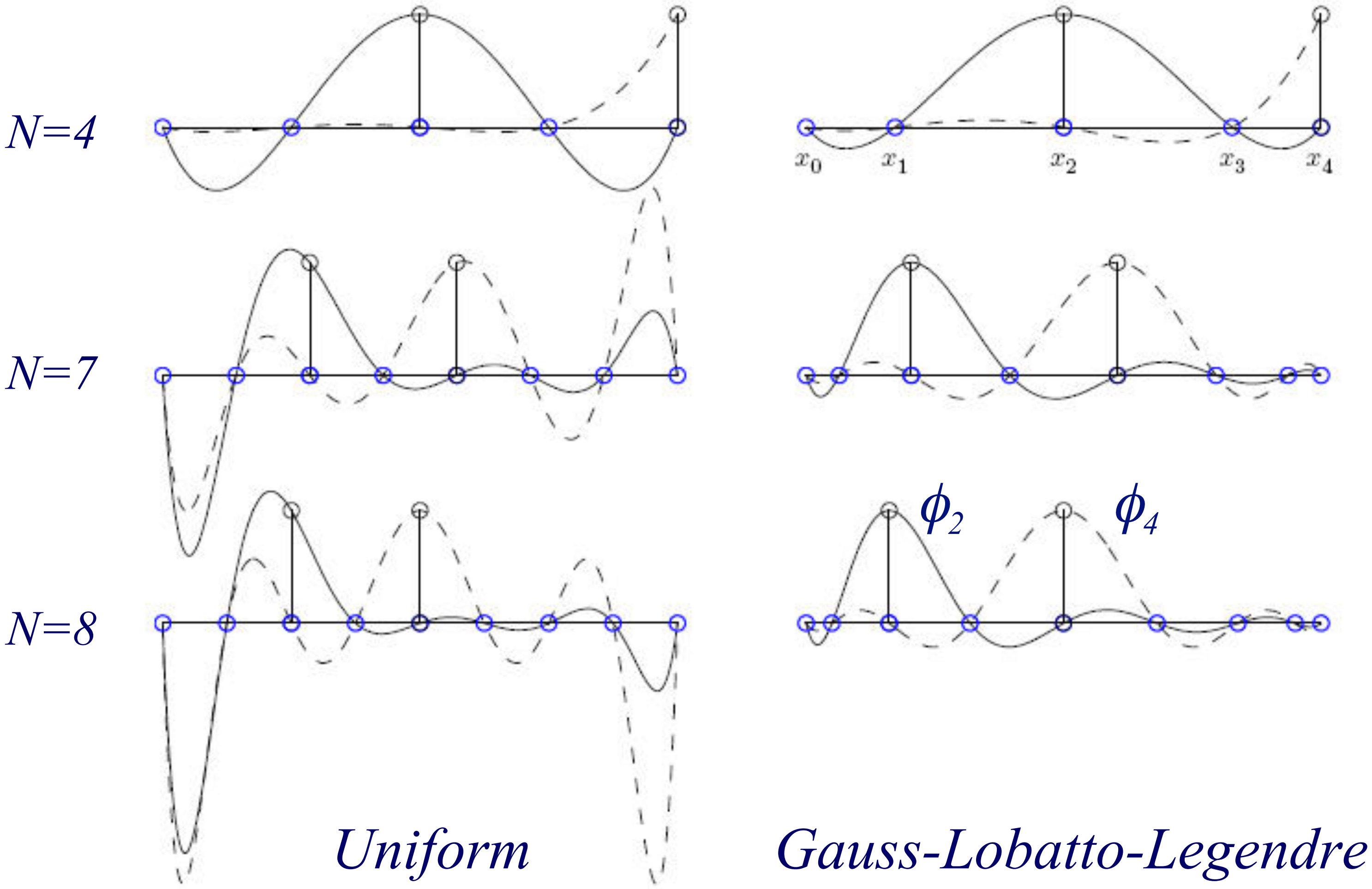


Imagine how the gradient of Q responds to f_i in this case …

Gaussian Quadrature

- *Gaussian quadrature rules* are based on polynomial interpolation, but nodes as well as weights are chosen to maximize degree of resulting rule
- With $2n$ parameters, we can attain degree of $2n - 1$
- Gaussian quadrature rules can be derived by method of undetermined coefficients, but resulting system of moment equations that determines nodes and weights is nonlinear
- Also, nodes are usually irrational, even if endpoints of interval are rational
- Although inconvenient for hand computation, nodes and weights are tabulated in advance and stored in subroutine for use on computer

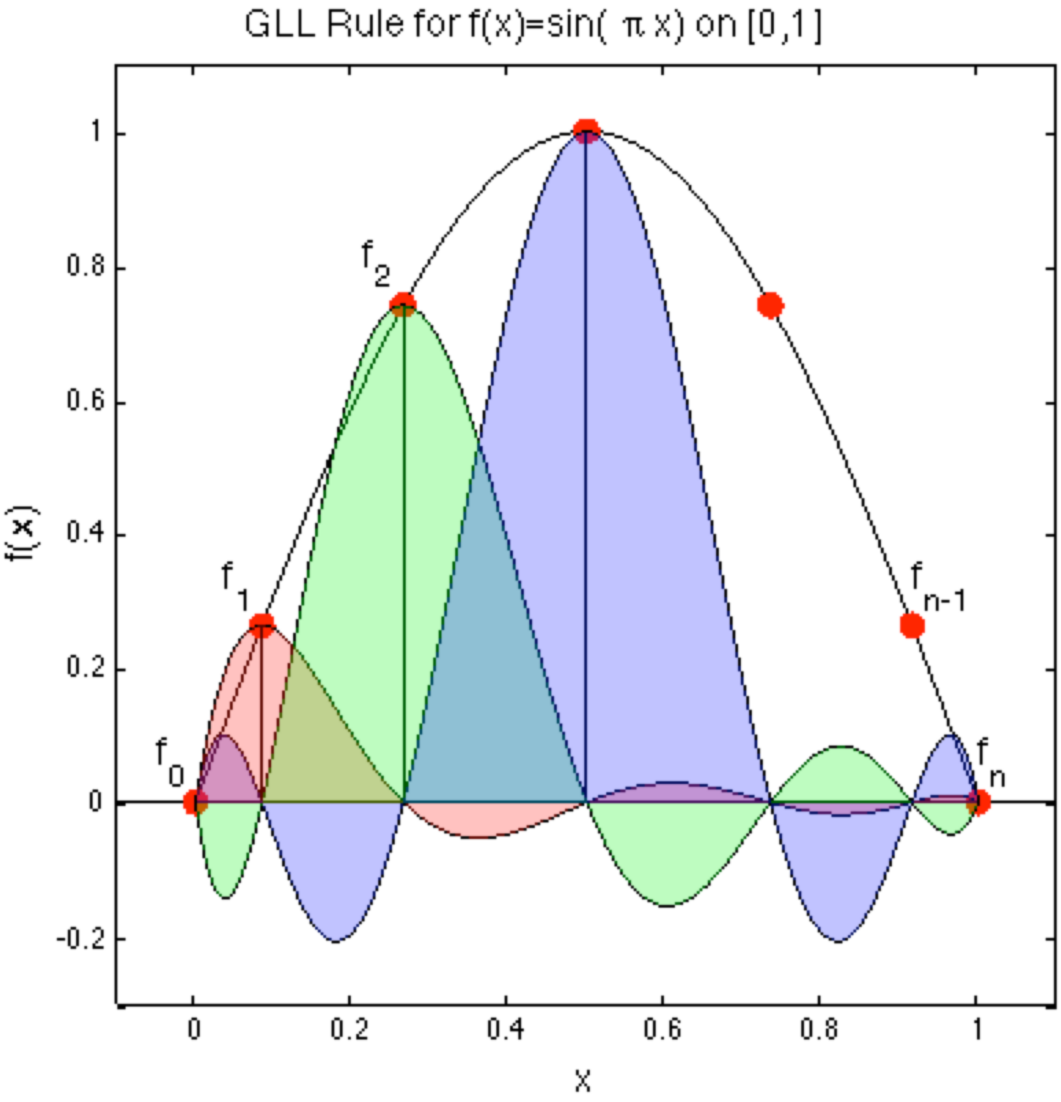
Lagrange Polynomials: Good and Bad Point Distributions



- ***We can see that for N=8 one of the uniform weights is close to becoming negative.***

Lagrange Polynomials: Good and Bad Point Distributions

gll_txt.m



- ***All weights are positive.***

Example: Gaussian Quadrature Rule

- Derive two-point Gaussian rule on $[-1, 1]$,

  $$G_2(f) = w_1 f(x_1) + w_2 f(x_2)$$

  where nodes x_i and weights w_i are chosen to maximize degree of resulting rule

- We use method of undetermined coefficients, but now nodes as well as weights are unknown parameters to be determined

- Four parameters are to be determined, so we expect to be able to integrate cubic polynomials exactly, since cubics depend on four parameters

Example, continued

- Requiring rule to integrate first four monomials exactly gives moment equations

$$\begin{aligned}
w_1 + w_2 &= \int_{-1}^{1} 1\,dx = x|_{-1}^{1} = 2 \\
w_1 x_1 + w_2 x_2 &= \int_{-1}^{1} x\,dx = (x^2/2)|_{-1}^{1} = 0 \\
w_1 x_1^2 + w_2 x_2^2 &= \int_{-1}^{1} x^2\,dx = (x^3/3)|_{-1}^{1} = 2/3 \\
w_1 x_1^3 + w_2 x_2^3 &= \int_{-1}^{1} x^3\,dx = (x^4/4)|_{-1}^{1} = 0
\end{aligned}$$

Example, continued

- One solution of this system of four nonlinear equations in four unknowns is given by

$$x_1 = -1/\sqrt{3}, \quad x_2 = 1/\sqrt{3}, \quad w_1 = 1, \quad w_2 = 1$$

- Another solution reverses signs of x_1 and x_2
- Resulting two-point Gaussian rule has form

$$G_2(f) = f(-1/\sqrt{3}) + f(1/\sqrt{3})$$

  and by construction it has degree three

- In general, for each n there is unique n-point Gaussian rule, and it is of degree $2n-1$
- Gaussian quadrature rules can also be derived using orthogonal polynomials

Topics for Today

- ❑ Gaussian quadrature
- ❑ Composite trapezoidal rule
- ❑ Richardson extrapolation
- ❑ Romberg integration
- ❑ Mechanical integration

- ❑ Numerical differentiation (start)

- ❑ Project proposals – almost done (should be out tonight)

Gauss Quadrature, I

Consider
$$I \;:=\; \int_{-1}^{1} f(x)\, dx.$$

Find w_i, x_i $i = 1, \dots, n$, to maximize degree of accuracy, M.

- Cardinality, $|\,.\,|$:
$$\begin{aligned} |\,\mathbb{P}_M\,| \;&=\; M \;+\; 1 \\ |\,w_i\,| \;+\; |\,x_i\,| \;&=\; 2n \\ M + 1 \;&=\; 2n \qquad \Longleftrightarrow \qquad M \;=\; 2n - 1 \end{aligned}$$

- Indeed, it is possible to find x_i and w_i such that *all polynomials of degree* $\leq M = 2n - 1$ are integrated *exactly.*

- The n nodes, x_i, are the zeros of the nth-order Legendre polynomial.

- The weights, w_i, are the integrals of the cardinal Lagrange polynomials associated with these nodes:
$$w_i \;=\; \int_{-1}^{1} l_i(x)\, dx, \qquad l_i(x) \in \mathbb{P}_{n-1}, \qquad l_i(x_j) = \delta_{ij}.$$

- Error scales like $\quad |I - Q_n| \sim C \dfrac{f^{(2n)}(\xi)}{(2n)!} \quad$ (Q_n exact for $f(x) \in \mathbb{P}_{2n-1}$.)

- n nodes are roots of orthogonal polynomials

Change of Interval

- Gaussian rules are somewhat more difficult to apply than Newton-Cotes rules because weights and nodes are usually derived for some specific interval, such as $[-1, 1]$
- Given interval of integration $[a, b]$ must be transformed into standard interval for which nodes and weights have been tabulated
- To use quadrature rule tabulated on interval $[\alpha, \beta]$,

$$\int_{\alpha}^{\beta} f(x)\, dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

to approximate integral on interval $[a, b]$,

$$I(g) = \int_{a}^{b} g(t)\, dt$$

we must change variable from x in $[\alpha, \beta]$ to t in $[a, b]$

Change of Interval, continued

- Many transformations are possible, but simple linear transformation

$$t = \frac{(b-a)x + a\beta - b\alpha}{\beta - \alpha}$$

  has advantage of preserving degree of quadrature rule

- Generally, the translation is much simpler:

$$t_i \;=\; a + \frac{\xi_i + 1}{2}(b-a)$$

- When $\xi = -1$, $t = a$. When $\xi = 1$, $t = b$.
- Here ξ_i, i=1,. . . ,n, are the Gauss points on (-1,1).

Use of Gauss Quadrature

- Generally, the translation is much simpler:

$$t_i \;=\; a + \frac{\xi_i + 1}{2}(b-a)$$

- When $\xi = -1$, $t = a$. When $\xi = 1$, $t = b$.
- Here ξ_i, i=1,...,n, are the Gauss points on (-1,1).
- So, you simply *look up**the (ξ_i, w_i) pairs, use the formula above to get t_i, then evaluate

$$Q_n \;=\; \frac{(b-a)}{2}\sum_i w_i f(t_i)$$

*(*that is, call a function)*

Use of Gauss Quadrature

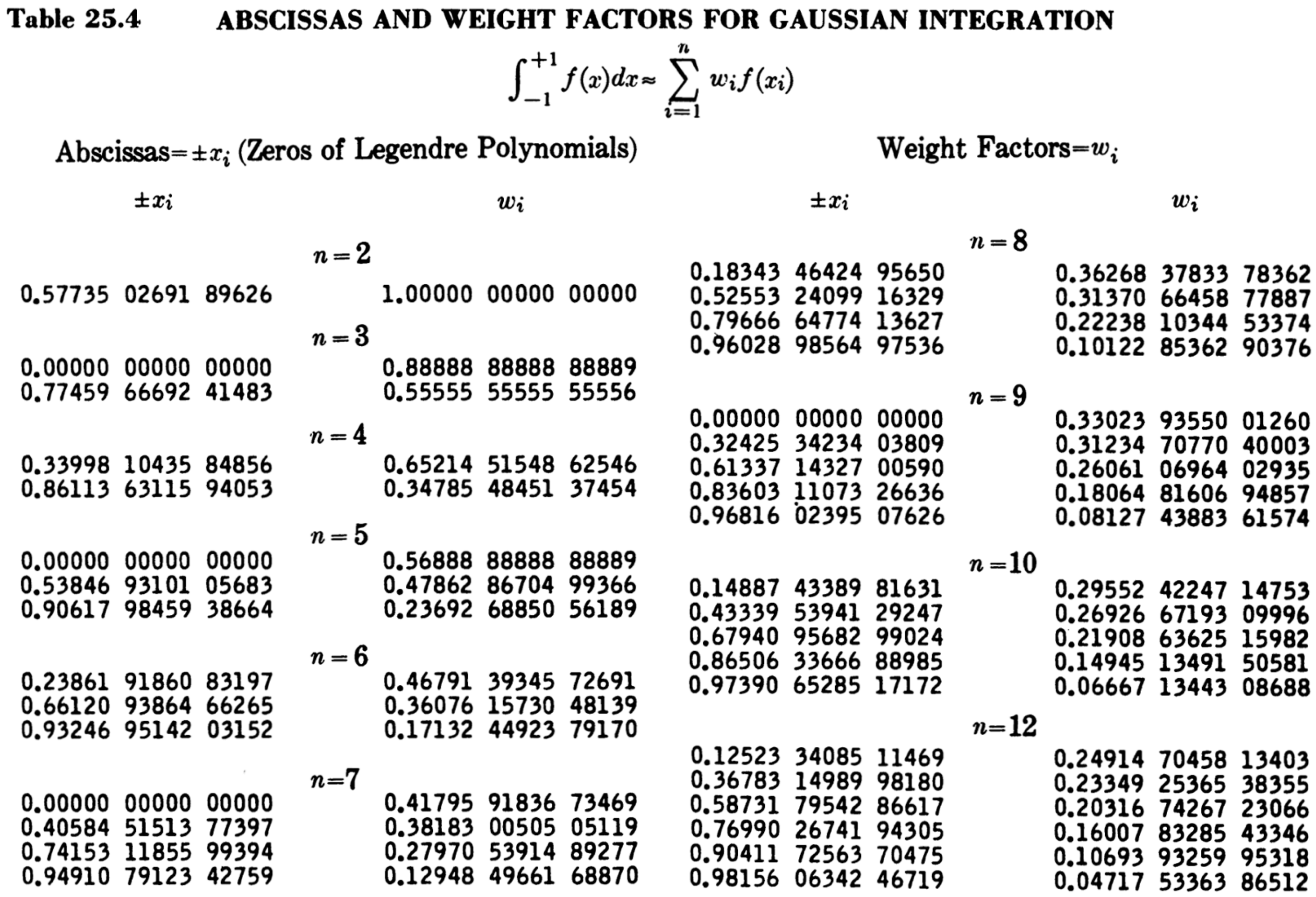


Table 25.4 **ABSCISSAS AND WEIGHT FACTORS FOR GAUSSIAN INTEGRATION**

$$\int_{-1}^{+1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

Abscissas$=\pm x_i$ (Zeros of Legendre Polynomials) Weight Factors$=w_i$

	$\pm x_i$	w_i
$n=2$	0.57735 02691 89626	1.00000 00000 00000
$n=3$	0.00000 00000 00000	0.88888 88888 88889
	0.77459 66692 41483	0.55555 55555 55556
$n=4$	0.33998 10435 84856	0.65214 51548 62546
	0.86113 63115 94053	0.34785 48451 37454
$n=5$	0.00000 00000 00000	0.56888 88888 88889
	0.53846 93101 05683	0.47862 86704 99366
	0.90617 98459 38664	0.23692 68850 56189
$n=6$	0.23861 91860 83197	0.46791 39345 72691
	0.66120 93864 66265	0.36076 15730 48139
	0.93246 95142 03152	0.17132 44923 79170
$n=7$	0.00000 00000 00000	0.41795 91836 73469
	0.40584 51513 77397	0.38183 00505 05119
	0.74153 11855 99394	0.27970 53914 89277
	0.94910 79123 42759	0.12948 49661 68870
$n=8$	0.18343 46424 95650	0.36268 37833 78362
	0.52553 24099 16329	0.31370 66458 77887
	0.79666 64774 13627	0.22238 10344 53374
	0.96028 98564 97536	0.10122 85362 90376
$n=9$	0.00000 00000 00000	0.33023 93550 01260
	0.32425 34234 03809	0.31234 70770 40003
	0.61337 14327 00590	0.26061 06964 02935
	0.83603 11073 26636	0.18064 81606 94857
	0.96816 02395 07626	0.08127 43883 61574
$n=10$	0.14887 43389 81631	0.29552 42247 14753
	0.43339 53941 29247	0.26926 67193 09996
	0.67940 95682 99024	0.21908 63625 15982
	0.86506 33666 88985	0.14945 13491 50581
	0.97390 65285 17172	0.06667 13443 08688
$n=12$	0.12523 34085 11469	0.24914 70458 13403
	0.36783 14989 98180	0.23349 25365 38355
	0.58731 79542 86617	0.20316 74267 23066
	0.76990 26741 94305	0.16007 83285 43346
	0.90411 72563 70475	0.10693 93259 95318
	0.98156 06342 46719	0.04717 53363 86512

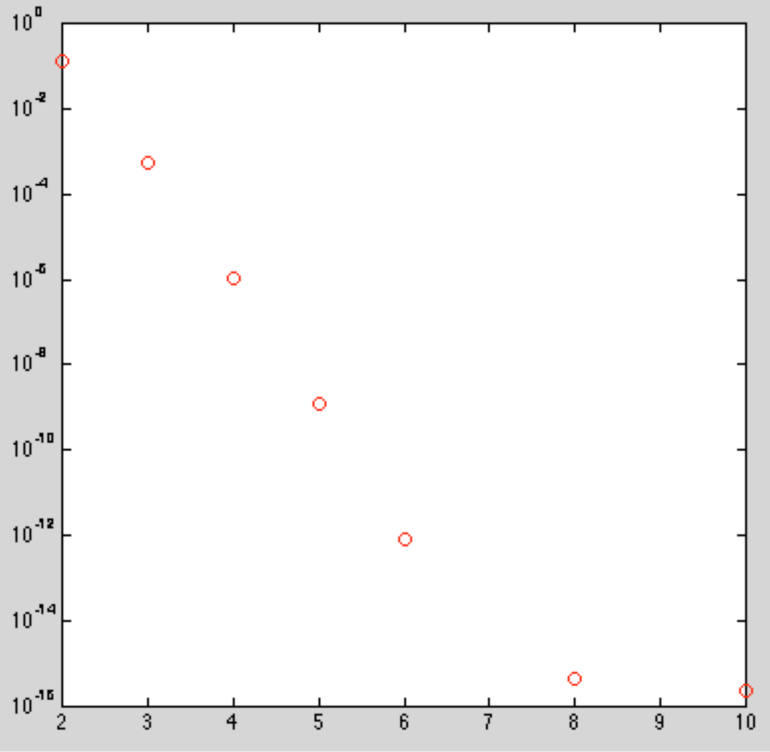
- There is a lot of software, in most every language, for computing the nodes and weights for all of the Gauss, Gauss-Lobatto, Gauss-Radau rules (Chebyshev, Legendre, etc.)

Gaussian Quadrature

- Gaussian quadrature rules have maximal degree and optimal accuracy for number of nodes used
- Weights are always positive and approximate integral always converges to exact integral as $n \to \infty$
- Unfortunately, Gaussian rules of different orders have no nodes in common (except possibly midpoint), so Gaussian rules are *not* progressive *(except for Gauss-Chebyshev)*
- Thus, estimating error using Gaussian rules of different order requires evaluating integrand function at full set of nodes of both rules

Gauss Quadrature Example

```
a=0; b=1;

for n=2:10;
  [z,w]=zwgll(n-1);  % Gauss-Lobatto-Legendre pts/wts

  t=a + 0.5*(z+1)*(b-a);
  f=exp(t);

  I   = w'*f*(b-a)/2.;
  err= abs(I-exact);
  [n I err ]

  semilogy(n,err,'ro'); hold on
end;
```



Gauss Quadrature Derivation

Gauss Quadrature: I

- Suppose $g(x) \in \mathbb{P}_m$ with zeros x_0, x_1, $\ldots$, x_{m-1}.

$$\begin{aligned} g(x) &= Ax^m + a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 \\ &= \underbrace{A(x-x_0)(x-x_1)\cdots(x-x_{m-1}).}_{\in \mathbb{P}_m} \end{aligned}$$

- Let $n < m-1$ and consider the first $n+1$ zeros:

$$\begin{aligned} g(x) &= \underbrace{(x-x_0)(x-x_1)\cdots(x-x_n)}_{=:\, q_{n+1}(x)} \underbrace{A\,(x-x_{n+1})\cdots(x-x_{m-1})}_{=:\, r(x) \in \mathbb{P}_{m-(n+1)}} \\ &= q_{n+1}(x)\, r(x) \end{aligned}$$

- Consider $f(x) \;\in\; \mathbb{P}_m, \;\; m \;>\; n+1,$

$$\begin{aligned} p_n(x) \;&\in\; \mathbb{P}_n, \qquad \text{(Lagrange interpolating polynomial.)} \\ p_n(x_i) \;&=\; f(x_i),\; i \;=\; 0,\ldots,n. \end{aligned}$$

- Then, let

$$\begin{aligned} g(x) \;&:=\; f(x) \;-\; p_n(x) \;\in\; \mathbb{P}_m \\ g(x_i) \;&=\; f(x_i) \;-\; p_n(x_i) \;=\; 0, \;\; i \;=\; 0,\ldots,n, \\ g(x) \;&=\; (x-x_0)(x-x_1)\cdots(x-x_n)\,A\,(x-x_{n+1})\cdots(x-x_{m-1}) \\ &=\; q_{n+1}(x)\,r(x), \end{aligned}$$

with $r(x) \in \mathbb{P}_{m-(n+1)}$.

- It follows that, for any $f(x) \in \mathbb{P}_m$, $m > n+1$, we can write

$$f(x) \;=\; p_n(x) \;+\; q_{n+1}(x)\,r(x),$$

with $r(x) \in \mathbb{P}_{m-(n+1)}$.

- Notice that

$$\begin{aligned}
\int_{-1}^{1} f(x)\,dx &= \int_{-1}^{1} p_n(x)\,dx \;+\; \int_{-1}^{1} q_{n+1}(x)\,r(x)\,dx \\
&= \int_{-1}^{1} \left(\sum_{i=0}^{n} f_i\, l_i(x) \right) dx \;+\; \int_{-1}^{1} q_{n+1}(x)\,r(x)\,dx \\
&= \sum_{i=0}^{n} f_i \int_{-1}^{1} l_i(x)\,dx \;+\; \int_{-1}^{1} q_{n+1}(x)\,r(x)\,dx \\
&= \sum_{i=0}^{n} f_i\, w_i \;+\; \int_{-1}^{1} q_{n+1}(x)\,r(x)\,dx,
\end{aligned}$$

with $w_i := \int_{-1}^{1} l_i(x)\,dx$.

- Note that the quadrature rule is *exact* iff

$$\int_{-1}^{1} q_{n+1}(x)\,r(x)\,dx \;=\; 0.$$

Orthogonal Polynomials

- The Legendre polynomials of degree k, $\pi_k(x) \in \mathbb{P}_k$ are *orthogonal polynomials* satisfying

$$(\pi_i, \pi_j) \;:=\; \int_{-1}^{1} \pi_i(x)\,\pi_j(x)\,dx \;=\; \delta_{ij}$$

- $\{\pi_0,\, \pi_1,\, \ldots\, , \pi_k\}$ form a *basis* (a spanning set) for $\mathbb{P}_k$.
- That is, for any $p_k(x) \in \mathbb{P}_k$ there is a unique set of coefficients γ_k such that

$$p_k(x) \;=\; \gamma_0\,\pi_0(x) \,+\, \gamma_1\,\pi_1(x) \,+\, \ldots \,+\, \gamma_k\,\pi_k(x).$$

- Note that, if $j > k$, then

$$(\pi_j, p_k) \;=\; \sum_{i=0}^{k} \gamma_i\,(\pi_j, \pi_i) \;=\; 0.$$

- Returning to quadrature, our rule will be exact if

$$\int_{-1}^{1} q_{n+1}(x)\, r(x)\, dx \;=\; 0.$$

- We get to choose the nodes, x_0, x_1, $\ldots, x_n$, that define q_{n+1}.
- If we choose the nodes to be the *zeros* of $\pi_{n+1}(x)$, then the integral will vanish if $r(x) \in \mathbb{P}_k$, with $k < n+1$.
- Recall that $r \in \mathbb{P}_{m-(n+1)}$, which implies

$$\begin{aligned} m-(n+1) &< n+1 \\ m &< 2n+2. \\ m &\leq 2n+1. \end{aligned}$$

- Thus, using $n+1$ nodes, x_0, x_1, $\ldots, x_n$, (the zeros of π_{n+1}), we have a quadrature rule that is **exact** for all polynomials of up to degree $2n+1$.

Gauss Quadrature: II

- A common way to state the Gauss quadrature problem is, *Find* $n+1$ *weights,* w_i*, and points,* x_i*, that will maximize the degree of polynomial,* m*, for which the quadrature rule will be exact.*
- Since you have $2n+2$ degrees of freedom (w_i, x_i), $i = 0, \ldots, n$ then you should be able to be exact for a space of functions having cardinality $2n+2$.
- The cardinality of $\mathbb{P}_m$ is $m+1$.
- Setting $m+1 = 2n+2$ we find $m = 2n+1$.

Gauss Lobatto Quadrature

- This is similar to the Gauss quadrature problem stated previously, save that we constrain $x_0 = -1$ and $x_n = +1$, which means we now have only $2n$ degrees of freedom.
- We should expect $m = 2n - 1$, and this is indeed the case when we choose the x_is to be the zeros of $(1 - x^2)P_n'(x)$.
- As before, the weights are the integrals of the Lagrange cardinal functions, $l_i(x)$.

Arbitrary Domains

- Usually, we're interested in more general integrals, i.e.,

$$\tilde{I} \;:=\; \int_a^b f(x)\,dx.$$

- The integral can be computed using a change of basis, $f[x(\xi)]$, with

$$x(\xi) \;:=\; a \;+\; \frac{b-a}{2}\,(1+\xi)\,.$$

- Note that

$$dx \;=\; \frac{b-a}{2}\,d\xi \;=\; \mathcal{J}\,d\xi,$$

  where $\mathcal{J}$ is the *Jacobian* associated with the map from $[-1,1]$ to $[a,b]$.

- With this change of basis, we have

$$\begin{aligned}
\tilde{I} &= \frac{b-a}{2} \int_{-1}^{1} f[x(\xi)]\, d\xi \;=\; \frac{b-a}{2} \int_{-1}^{1} \hat{f}(\xi)\, d\xi \\
&\approx \frac{b-a}{2} \left(\sum_{i=0}^{n} w_i \hat{f}(\xi_i) \right) \\
&\approx \frac{b-a}{2} \left(\sum_{i=0}^{n} w_i f(x_i) \right) \\
&\approx \frac{b-a}{2} \left(\sum_{i=0}^{n} w_i f_i \right).
\end{aligned}$$

- The quadrature weights stay the same.
- The quadrature points are given by $x_i = a + \dfrac{1}{2}(b-a)(1+\xi_i)$.

- Here, the ξ_is are the zeros of $\pi_{n+1}(\xi)$ or $(1-\xi^2)\pi'_n(\xi)$, depending on whether one wants Gauss-Legendre (GL) or Gauss-Lobatto-Legendre (GLL) quadrature.

- There are closed form expressions for the quadrature weights:

$$\underline{\text{Gauss}} \qquad\qquad\qquad \underline{\text{Gauss-Lobatto}}$$

$$w_i \;=\; \frac{2}{(1-\xi_i)^2 P'_{n+1}(\xi_i)} \qquad\qquad w_i \;=\; \frac{2}{n(n+1)[P_n(\xi)]^2},$$

  where P_k is the kth-order Legendre polynomial normalized such that $P_k(1) = 1$.

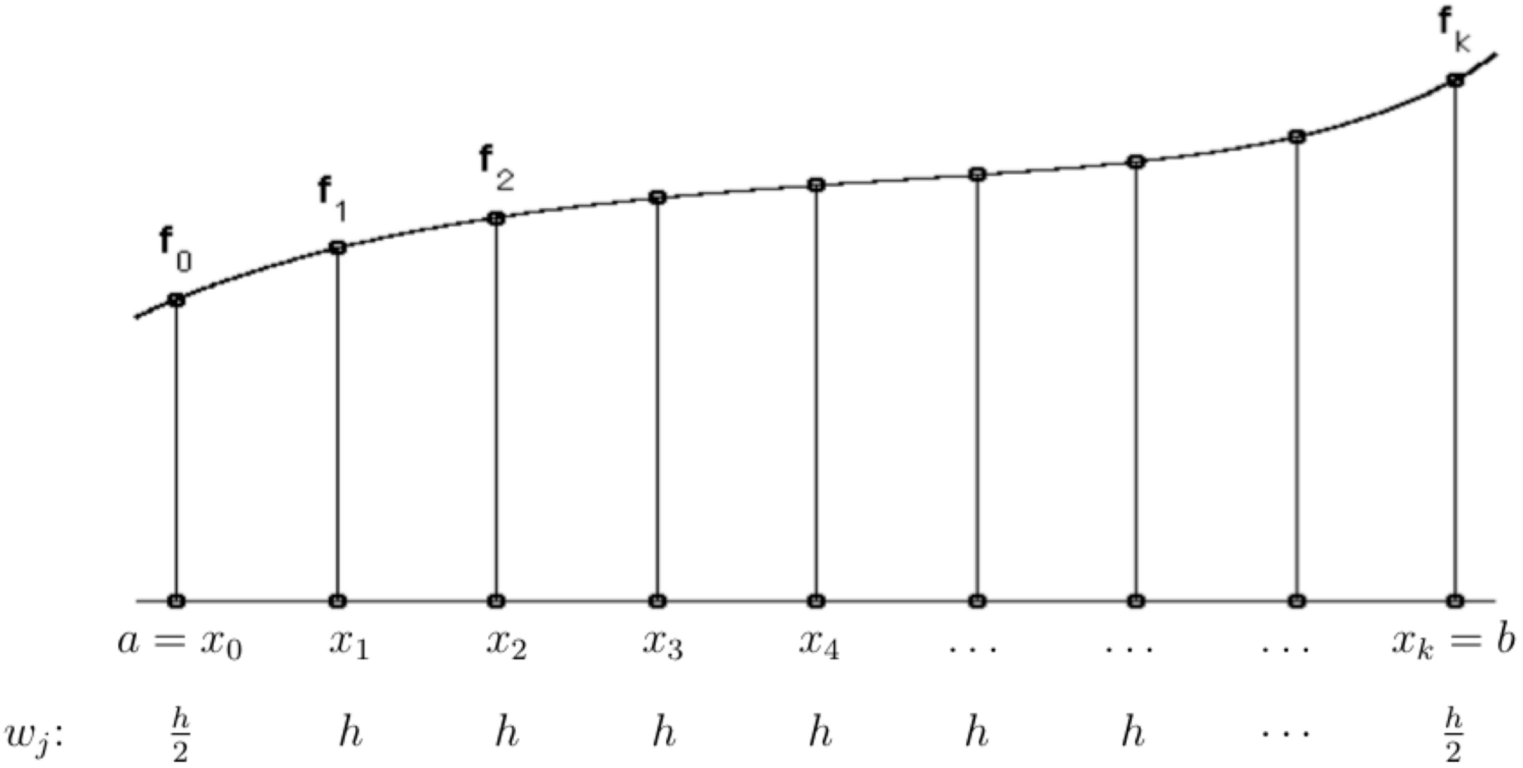
- Most algorithms for finding the ξ_is are based on solving an eigenvalue problem for a tridiagonal matrix (the companion matrix).
- The matlab script `zwgll.m` returns the set (ξ_i, w_i) for $n+1$ node points.

```
function [z,w] = zwgll(p);

% Compute the p+1 Gauss-Lobatto-Legendre nodes z on [-1,1],
% i.e. the zeros of the first derivative of the Legendre
% polynomial of degree p plus -1 and 1 and the p+1 weights w.

n = p+1;

z(1:n)=0; w(1:n)=0; z(1)=-1; z(n)= 1;

if p>1,
  if p==2
   z(2)=0;
  else
   M=zeros(p-1,p-1);
   for i=1:p-2,
    M(i,i+1)=(1/2)*sqrt((i*(i+2))/((i+1/2)*(i+3/2)));
    M(i+1,i)=M(i,i+1);
   end;

   [V,D]=eig(M);
   z(2:p)=sort(eig(M));
  end;
end;

%compute the weights w

w(1)=2/(p*(n)); w(n)=w(1);
for i=2:p,
  x=z(i); z0=1; z1=x;
  for j=1:p-1,
    z2=x.*z1*(2*j+1)/(j+1)-z0*j/(j+1);
    z0=z1; z1=z2;
  end;
  w(i)=2/(p*(n)*z2*z2);
end;

z=z'; w=w';
```

Composite Rules

- Main Idea: Use your favorite rule on each panel and sum across all panels.

- Particularly good if f(x) not differentiable at the knot points.



Composite Quadrature

- Alternative to using more nodes and higher degree rule is to subdivide original interval into subintervals, then apply simple quadrature rule in each subinterval
- Summing partial results then yields approximation to overall integral
- This approach is equivalent to using piecewise interpolation to derive *composite quadrature rule*
- Composite rule is always stable if underlying simple rule is stable
- Approximate integral converges to exact interval as number of subintervals goes to infinity provided underlying simple rule has degree at least zero

Composite Quadrature Rules

❑ Composite Trapezoidal (Q_{CT}) and Composite Simpson (Q_{CS}) rules work by breaking the interval [a,b] into panels and then applying either trapezoidal or Simpson method to each panel.

❑ Q_{CT} is the most common, particularly since Q_{CS} is readily derived via Richardson extrapolation at no extra work.

❑ Q_{CT} can be combined with Richardson extrapolation to get higher order accuracy, not quite competitive with Gauss quadrature, but a significant improvement over Q_{CT} alone. (Using ***multiple rounds of Richardson extrapolation*** is known as Romberg integration.)

❑ **For periodic functions, Q_{CT} *is* a Gauss quadrature rule.**

Examples: Composite Quadrature

- Subdivide interval $[a, b]$ into k subintervals of length $h = (b - a)/k$, letting $x_j = a + jh$, $j = 0, \dots, k$
- *Composite midpoint rule*

$$M_k(f) = \sum_{j=1}^{k} (x_j - x_{j-1})\, f\left(\frac{x_{j-1} + x_j}{2}\right) = h \sum_{j=1}^{k} f\left(\frac{x_{j-1} + x_j}{2}\right)$$

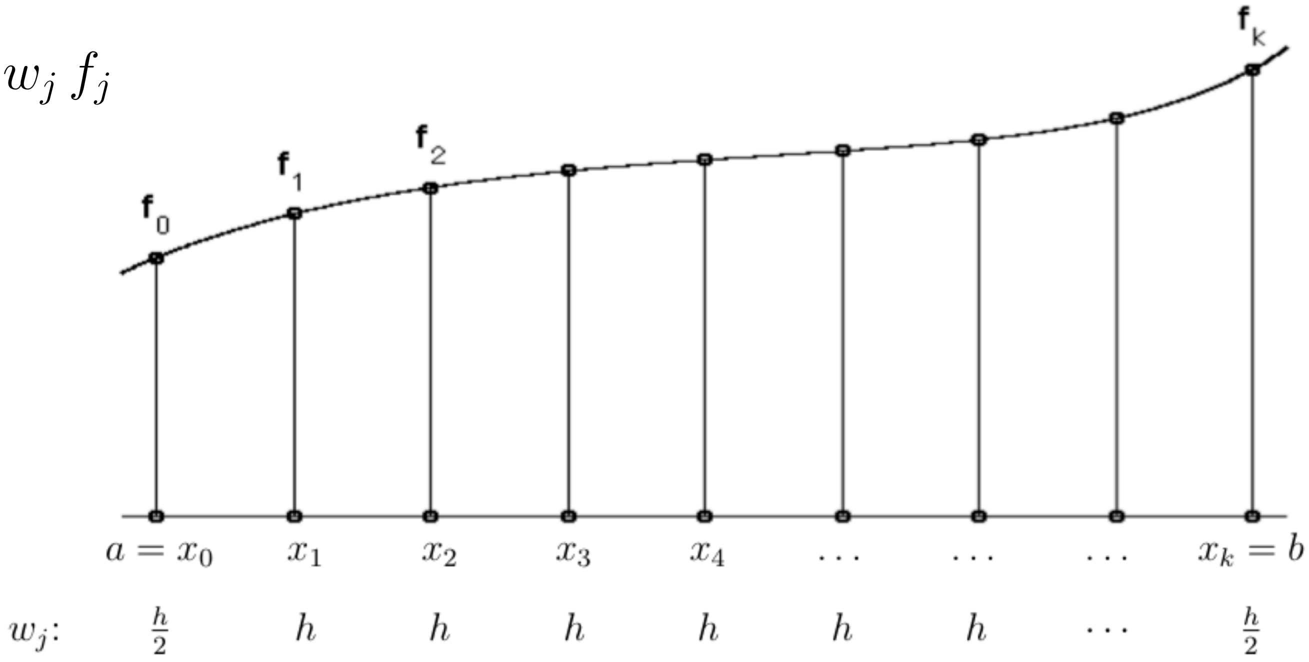
- *Composite trapezoid rule*

$$\begin{aligned} T_k(f) &= \sum_{j=1}^{k} \frac{(x_j - x_{j-1})}{2}\, (f(x_{j-1}) + f(x_j)) \\ &= h\, \left(\tfrac{1}{2} f(a) + f(x_1) + \cdots + f(x_{k-1}) + \tfrac{1}{2} f(b)\right) \end{aligned}$$

Implementation of Composite Trapezoidal Rule

Assuming uniform spacing $h = (b-a)/k$,

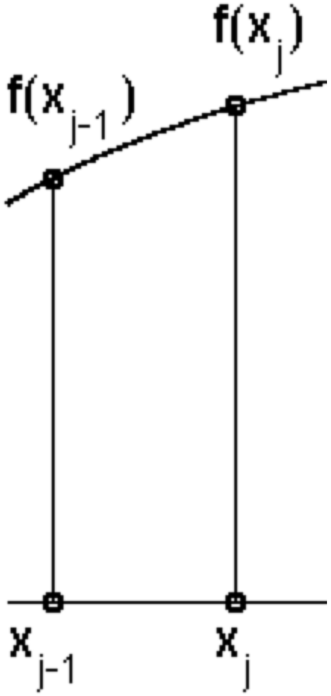
$$
\begin{aligned}
Q_{CT} &:= \sum_{j=1}^{k} Q_j \;=\; \sum_{j=1}^{k} \frac{h}{2}(f_{j-1} + f_j) \\
&= \frac{h}{2} f_0 + h f_1 + h f_2 + \ldots + \ldots + h f_{k-1} + \frac{h}{2} f_k \\
&= \sum_{j=1}^{k} w_j \, f_j
\end{aligned}
$$



Composite Trapezoidal Rule

$$
\begin{aligned}
I_j \;:=\; \int_{x_{j-1}}^{x_j} f(x)\,dx \;&=\; \frac{h}{2}(f_{j-1}+f_j) \;+\; O(h^3) \\
&=:\; Q_j \;+\; O(h^3) \\
&=\; Q_j \;+\; c_j h^3 + \textit{higher order terms.}
\end{aligned}
$$

$$c_j \le \frac{1}{12} \max_{[x_{j-1},x_j]} |f''(x)|$$



$$I \;=\; \int_a^b f(x)\,dx \;=\; \underbrace{\sum_{j=1}^{k} Q_j}_{Q_{CT}} \;+\; \sum_{j=1}^{k} c_j h^3 \;+\; h.o.t.$$

$$|I - Q_{CT}| \;+\; h.o.t. = h^3 \sum_{j=1}^{k} c_j \;\le\; h^3 k \max_j |c_j| \;=\; h^2 \max_j |c_j| \; \textit{(b-a)}$$

Recall, hk = (b-a).

Other Considerations with Composite Trapezoidal Rule

Composite Trapezoidal Rule: (uniform spacing, n panels)

$$T_h := h\left[\frac{f_0+f_1}{2} + \frac{f_1+f_2}{2} + \cdots + \frac{f_{n-1}+f_n}{2}\right]$$

$$= h\left[\sum_{i=1}^{n-1} f_i + \frac{1}{2}(f_0+f_n)\right],$$

$$= \sum_{i=0}^{n} w_i\, f_i, \qquad w_0 = w_n = \frac{h}{2},\ w_i = h \text{ otherwise.}$$

Stable ($w_i > 0$).

Composite Midpoint Rule: (uniform spacing, n panels)

$$
\begin{aligned}
M_h &:= h\left[f_{\frac{1}{2}} + f_{\frac{3}{2}} + \cdots + f_{n-\frac{1}{2}}\right] \\
&= h\left[\sum_{i=1}^{n} f_{i-\frac{1}{2}}\right] = h\sum_{i=1}^{n} f(x_0 + ih - \frac{h}{2}) \\
&= \sum_{i=0}^{n} w_i\, f_i, \qquad w_i = h.
\end{aligned}
$$

(Note number of function evaluations.)

Accuracy? $|\tilde{I} - I_h| =??$

Single Panel: $x \in [x_{i-1}, x_i]$.

- Taylor series about $x_{i-\frac{1}{2}}$:

$$f(x) \;=\; f_{i-\frac{1}{2}} + (x - x_{i-\frac{1}{2}})f'_{i-\frac{1}{2}} + \frac{(x - x_{i-\frac{1}{2}})^2}{2} f''_{i-\frac{1}{2}} + \frac{(x - x_{i-\frac{1}{2}})^3}{3!} f'''_{i-\frac{1}{2}} + \cdots$$

- Integrate:

$$\begin{aligned}
\tilde{I}_i := \int_{x_{i-1}}^{x_i} f(x)\,dx \;&=\; hf_{i-\frac{1}{2}} + \left.\frac{(x - x_{i-\frac{1}{2}})^2}{2}\right|_{x_{i-1}}^{x_i} f'_{i-\frac{1}{2}} + \left.\frac{(x - x_{i-\frac{1}{2}})^3}{3!}\right|_{x_{i-1}}^{x_i} f''_{i-\frac{1}{2}} + \cdots \\
&=\; hf_{i-\frac{1}{2}} + 0 + \frac{2h^3}{3!\,2^3} f''_{i-\frac{1}{2}} + 0 + \frac{2h^5}{5!\,2^5} f^{iv}_{i-\frac{1}{2}} + 0 + \frac{2h^7}{7!\,2^7} f^{vi}_{i-\frac{1}{2}} + \cdots \\
&=\; hf_{i-\frac{1}{2}} + \frac{h^3}{24} f''_{i-\frac{1}{2}} + \frac{h^5}{1920} f^{iv}_{i-\frac{1}{2}} + \frac{h^7}{322560} f^{vi}_{i-\frac{1}{2}} + \cdots \\
&=\; M_i + \frac{h^3}{24} f''_{i-\frac{1}{2}} + \frac{h^5}{1920} f^{iv}_{i-\frac{1}{2}} + \frac{h^7}{322560} f^{vi}_{i-\frac{1}{2}} + \cdots
\end{aligned}$$

- Therefore, midpoint rule error (single panel) is:

$$M_i \;=\; \tilde{I} - \frac{h^3}{24} f''_{i-\frac{1}{2}} - \frac{h^5}{1920} f^{iv}_{i-\frac{1}{2}} - \frac{h^7}{322560} f^{vi}_{i-\frac{1}{2}} - \cdots$$

- Locally, midpoint rule is $O(h^3)$ accurate.

Trapezoidal Rule: (single panel)

- Taylor series about $x_{i-\frac{1}{2}}$.

$$f(x_{i-1}) \;=\; f_{i-\frac{1}{2}} - \frac{h}{2} f'_{i-\frac{1}{2}} + \left(\frac{h}{2}\right)^2 \frac{f''_{i-\frac{1}{2}}}{2!} - \left(\frac{h}{2}\right)^3 \frac{f'''_{i-\frac{1}{2}}}{3!} + \cdots$$

$$f(x_i) \;=\; f_{i-\frac{1}{2}} + \frac{h}{2} f'_{i-\frac{1}{2}} + \left(\frac{h}{2}\right)^2 f''_{i-\frac{1}{2}} + \left(\frac{h}{2}\right)^3 f'''_{i-\frac{1}{2}} + \cdots$$

- Take average:

$$\begin{aligned}
\frac{h}{2}\left[f_{i-1} + f_i\right] &= M_h + \frac{h^3}{2!\,2^2} f''_{i-\frac{1}{2}} + \frac{h^5}{4!\,2^4} f^{iv}_{i-\frac{1}{2}} + \frac{h^7}{6!\,2^6} f^{vi}_{i-\frac{1}{2}} + \cdots \\
&= \tilde{I}_i + \frac{h^3}{2!\,2^2}\left(1 - \frac{1}{3}\right) f''_{i-\frac{1}{2}} + \frac{h^5}{4!\,2^4}\left(1 - \frac{1}{5}\right) f^{iv}_{i-\frac{1}{2}} + \frac{h^7}{6!\,2^6}\left(1 - \frac{1}{7}\right) f^{vi}_{i-\frac{1}{2}} + \cdots \\
&= \tilde{I}_i + \frac{h^3}{12} f'' + \frac{h^5}{480} f^{iv} + \frac{h^7}{53760} f^{vi} + \cdots \\
&= \tilde{I}_i + c_2 h^3\, f''_{i-\frac{1}{2}} + c_4 h^5\, f^{iv}_{i-\frac{1}{2}} + c_6 h^7\, f^{vi}_{i-\frac{1}{2}} + \cdots
\end{aligned}$$

- Leading-order error for trapezoid rule is twice that of midpoint.

Composite Rule: Sum trapezoid rule across n panels:

$$\begin{aligned}
I_{CT} \;&:=\; h\left[\sum_{i=1}^{n-1} f_i + \frac{1}{2}(f_0 + f_n)\right] \\
&=\; \sum_{i=1}^{n} \frac{h}{2}[f_{i-1} + f_i] \\
&=\; \sum_{i=1}^{n} \left[\tilde{I}_i \;+\; c_2 h^3\, f''_{i-\frac{1}{2}} \;+\; c_4 h^5\, f^{iv}_{i-\frac{1}{2}} \;+\; c_6 h^7\, f^{vi}_{i-\frac{1}{2}} \;+\; \cdots\right] \\
&=\; \tilde{I} \;+\; c_2 h^2 \left[h \sum_{i=1}^{n} f''_{i-\frac{1}{2}}\right] \;+\; c_4 h^4 \left[h \sum_{i=1}^{n} f^{iv}_{i-\frac{1}{2}}\right] \;+\; \cdots \\
&=\; \tilde{I} \;+\; \frac{h^2}{12}\left[\int_a^b f''\,dx \;+\; \text{h.o.t.}\right] \;+\; c_4 h^4 \left[\int_a^b f^{iv}\,dx \;+\; \text{h.o.t.}\right] \;+\; \cdots \\
&=\; \tilde{I} \;+\; \frac{h^2}{12}\,[f'(b) - f'(a)] \;+\; O(h^4).
\end{aligned}$$

- Global truncation error is $O(h^2)$ and has a particularly elegant form.
- Can estimate $f'(a)$ and $f'(b)$ with $O(h^2)$ accurate formula to yield $O(h^4)$ accuracy.
- With care, can also precisely define the coefficient for h^4, h^6, and other terms (Euler-Maclaurin Sum Formula).

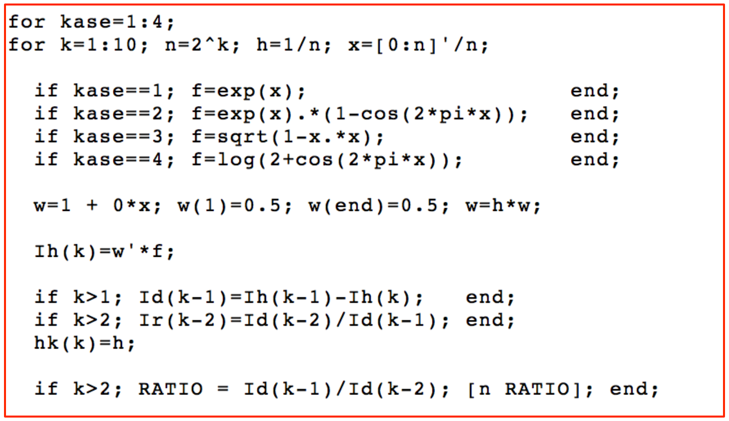
Examples.

- Apply (composite) trapezoidal rule for several endpoint conditions, $f'(a)$ and $f'(b)$:

  1. Standard case (nothing special).
  2. Lucky case ($f'(a) = f'(b) = 0$).
  3. Unlucky case ($f'(b) = -\infty$).
  4. Really lucky case ($f^{(k)}(a) = f^{(k)}(b)$, $k = 1, 2, \ldots$).

- Functions on $[a, b] = [0, 1]$:

$$\begin{array}{lrcl} (1) & f(x) & = & e^x \\ (2) & f(x) & = & e^x\,(1 - \cos 2\pi x) \\ (3) & f(x) & = & \sqrt{1 - x^2} \\ (4) & f(x) & = & \log(2 + \cos 2\pi x) \\ (5) & f(x) & = & e^{-30x^2} \\ (6) & f(x) & = & e^{\cos 30\pi x}. \end{array}$$



```
for kase=1:4;
for k=1:10; n=2^k; h=1/n; x=[0:n]'/n;

  if kase==1; f=exp(x);                         end;
  if kase==2; f=exp(x).*(1-cos(2*pi*x));        end;
  if kase==3; f=sqrt(1-x.*x);                   end;
  if kase==4; f=log(2+cos(2*pi*x));             end;

  w=1 + 0*x; w(1)=0.5; w(end)=0.5; w=h*w;

  Ih(k)=w'*f;

  if k>1; Id(k-1)=Ih(k-1)-Ih(k);   end;
  if k>2; Ir(k-2)=Id(k-2)/Id(k-1); end;
  hk(k)=h;

  if k>2; RATIO = Id(k-1)/Id(k-2); [n RATIO]; end;
```

- `quad1.m` example.

Trapezoidal Rule Example

Quadrature: Trapezoidal Rule

Let $$\mathcal{I} \;:=\; \int_a^b f(x)\,dx \;\approx\; \sum_{j=0}^{N} w_j f(x_j) \;=:\; Q_N.$$

For trapezoidal rule (with uniform spacing, say),

$$x_j \;=\; a \,+\, j\cdot\Delta x, \;\; \Delta x \;:=\; (b-a)/N,$$

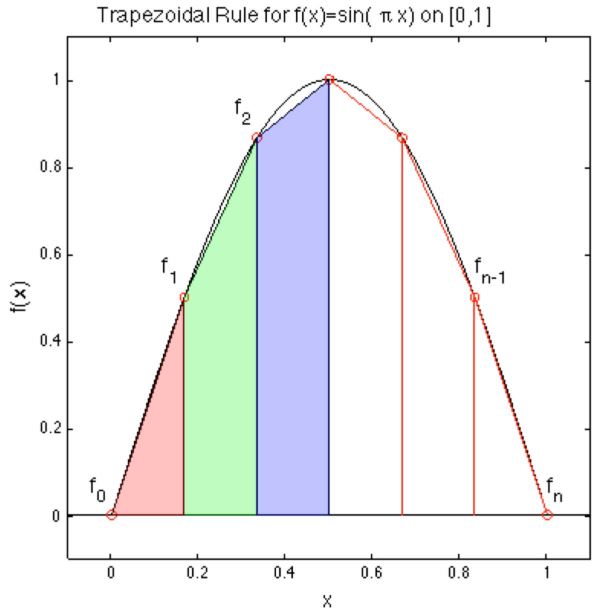
$$w_j \;=\; \Delta x,\; j \;=\; 1,\, 2,\, \ldots, n-1$$

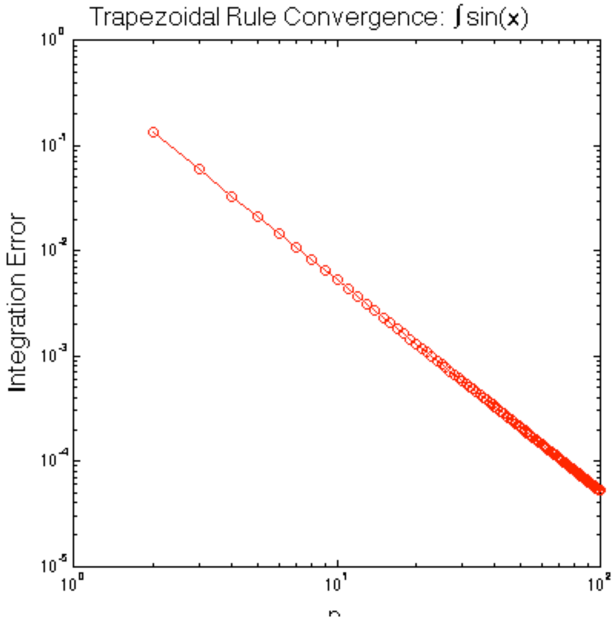
$$w_0 \;=\; w_n \;=\; \frac{1}{2}\Delta x.$$

- Convergence is $O(N^{-2})$:

$$|\,\mathcal{I} \,-\, Q_N\,| \;\sim\; CN^{-2}$$

trap_v_gll.m, trap_txt.m





Trapezoidal Rule vs. Gauss-Lobatto-Legendre Quadrature

Gauss-Lobatto-Legendre Quadrature

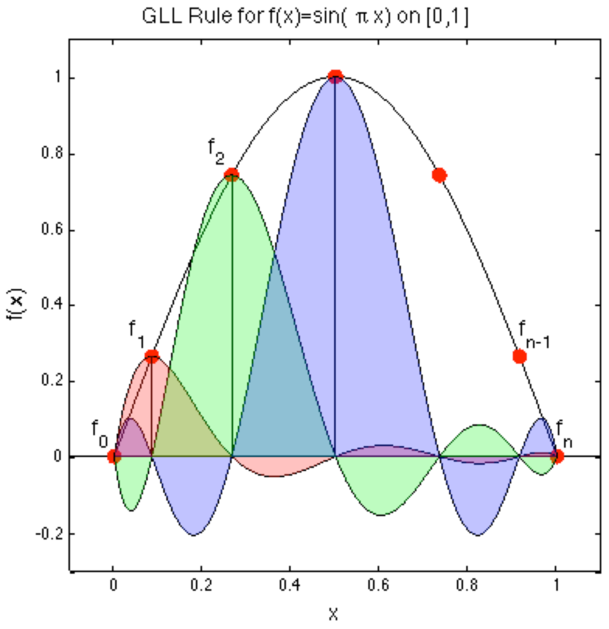
Let $\mathcal{I} \;:=\; \int_a^b f(x)\,dx \;\approx\; \sum_{j=0}^{N} w_j f(x_j) \;=:\; Q_N.$

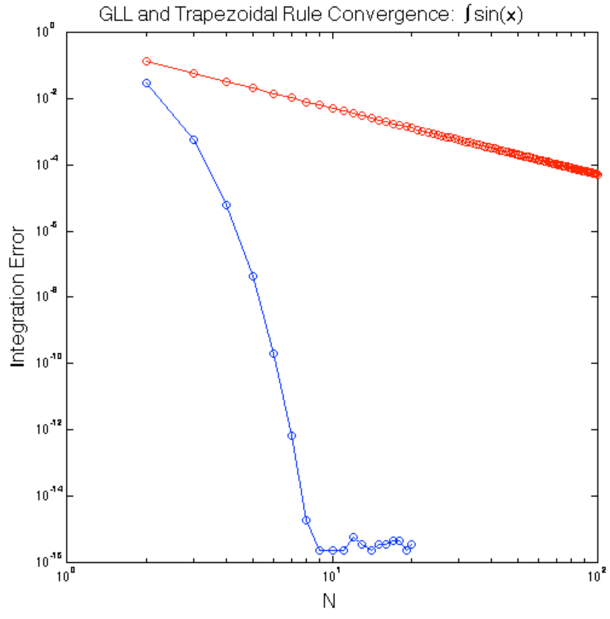
$$
\begin{aligned}
x_j &= a + \frac{b-a}{2}(\xi_j + 1) \\
\xi_j &= \text{GLL quadrature points} \;=\; \text{zeros of } (1-\xi^2)\,L_N'(\xi) \\
w_j &= \frac{b-a}{2}\int_{-1}^{1} h_j(\xi)\,d\xi \\
&= \frac{b-a}{2}\rho_j, \quad \rho_j \;:=\; \text{GLL quadrature weight on } [-1,1]
\end{aligned}
$$

- For smooth functions, convergence is $O(e^{-\sigma N})$:

$$|\mathcal{I} - Q_N| \;\sim\; Ce^{-\sigma N}, \quad \sigma > 0.$$

trap_v_gll.m, gll_txt.m





Trapezoidal Rule vs. Gauss-Lobatto-Legendre Quadrature

Gauss-Lobatto-Legendre Quadrature

Let $\mathcal{I} \;:=\; \int_a^b f(x)\,dx \;\approx\; \sum_{j=0}^{N} w_j f(x_j) \;=:\; Q_N.$

$$x_j \;=\; a \;+\; \frac{b-a}{2}\,(\xi_j \;+\; 1)$$

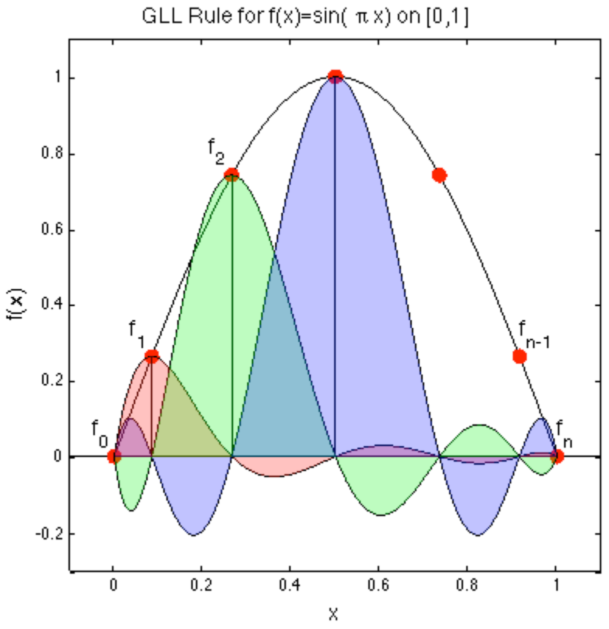
$$\xi_j \;=\; \text{GLL quadrature points} \;=\; \text{zeros of}\,(1-\xi^2)\,L'_N(\xi)$$

$$w_j \;=\; \frac{b-a}{2}\int_{-1}^{1} h_j(\xi)\,d\xi$$

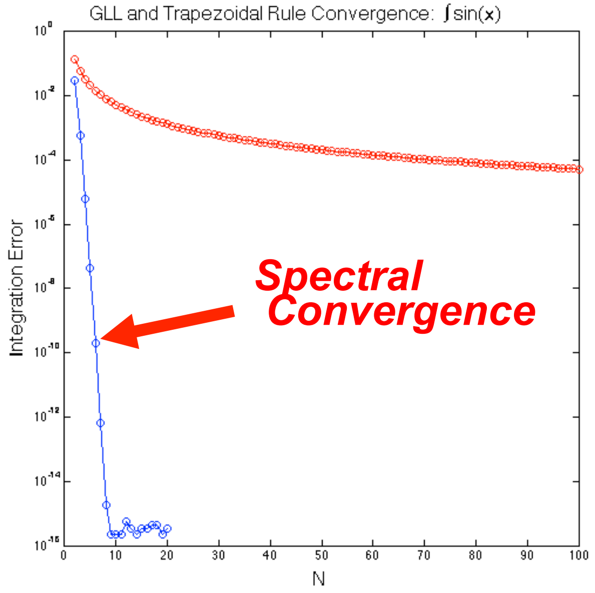
$$\;=\; \frac{b-a}{2}\rho_j, \quad \rho_j \;:=\; \text{GLL quadrature weight on } [-1,1]$$

- For smooth functions, convergence is $O(e^{-\sigma N})$:

$$|\,\mathcal{I} \,-\, Q_N\,| \;\sim\; Ce^{-\sigma N}, \quad \sigma > 0.$$



GLL Rule for f(x)=sin(π x) on [0,1]

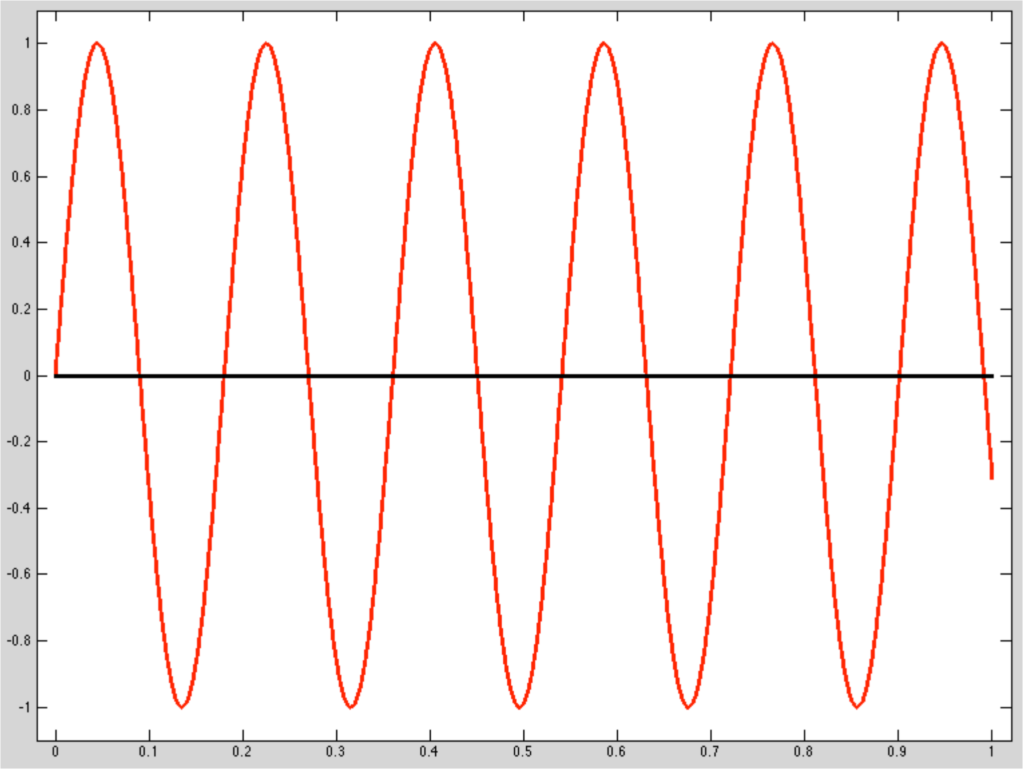


GLL and Trapezoidal Rule Convergence: ∫sin(x)

Spectral Convergence

Working with 1D Nodal Bases

- What is the convergence behavior for highly oscillatory functions?



trap_v_gll_k.m

Strategies to improve to $O(h^4)$ or higher?

- **Endpoint Correction.**
  - Estimate $f'(a)$ and $f'(b)$ to $O(h^2)$ using available f_i data.
  - **How?**
  - **Q:** What happens if you don't have at least $O(h^2)$ accuracy?
  - **-** Requires knowing the c_2 coefficient. :(

trap_endpoint.m

- **Richardson Extrapolation.**

$$
\begin{aligned}
I_h &= \tilde{I} + c_2 h^2 + O(h^4) \\
I_{2h} &= \tilde{I} + 4c_2 h^2 + O(h^4) \\
&\quad \text{(Reuses } f_i,\ i\text{=even!)} \\
I_R &= \left[\frac{4}{3} I_h - \frac{1}{3} I_{2h} \right] \\
&= I_{\text{SIMPSON}}\,!
\end{aligned}
$$

Composite Trapezoidal + Richardson Extrapolation

For composite trapezoidal rule, can show that if $f \in C^{2K+1}$ then

$$I \;=\; I_{CT} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots + \tilde{c}_{2K} h^{2K} + O(h^{2K+1})$$

Suggests the following strategy:

$$(1)\; I \;=\; I_{CT(h)} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots$$

$$(2)\; I \;=\; I_{CT(2h)} + \tilde{c}_2 (2h)^2 + \tilde{c}_4 (2h)^4 + \tilde{c}_6 (2h)^6 + \ldots$$

Take 4 ×(1)-(2) (*eliminate* $O(h^2)$ *term*):

$$4I - I \;=\; 4I_{CT(h)} - I_{CT(2h)} + c_4' h^4 + c_6' h^6 + \ldots$$

$$I \;=\; \frac{4}{3} I_{CT(h)} - \frac{1}{3} I_{CT(2h)} + \hat{c}_4 h^4 + \hat{c}$$

$$= \; I_{S(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + h.o.t.$$

This is Richardson Extrapolation.

Understand it.

Example: f=x²

Here,

$$I_{S(2h)} \;:=\; \frac{4}{3} I_{CT(h)} - \frac{1}{3} I_{CT(2h)}$$

Composite Trapezoidal + Richardson Extrapolation

For composite trapezoidal rule, can show that if $f \in C^{2K+1}$ then

$$\boxed{I \;=\; I_{CT} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots + \tilde{c}_{2K} h^{2K} + O(h^{2K+1})}$$

Suggests the following strategy:

$$\begin{aligned}
(1)\; I &= I_{CT(h)} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots \\
(2)\; I &= I_{CT(2h)} + \tilde{c}_2 (2h)^2 + \tilde{c}_4 (2h)^4 + \tilde{c}_6 (2h)^6 + \ldots
\end{aligned}$$

Take 4 ×(1)-(2) (*eliminate* $O(h^2)$ *term*):

$$\begin{aligned}
4I - I &= 4I_{CT(h)} - I_{CT(2h)} + c_4' h^4 + c_6' h^6 + \ldots \\
I &= \frac{4}{3} I_{CT(h)} - \frac{1}{3} I_{CT(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + \ldots \\
&= I_{S(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + h.o.t.
\end{aligned}$$

Here,

$$\boxed{I_{S(2h)} \;:=\; \frac{4}{3} I_{CT(h)} - \frac{1}{3} I_{CT(2h)}}$$

Composite Trapezoidal + Richardson Extrapolation

Can in fact show that if $f \in C^{2K+1}$ then

$$I = Q_{CT} + \boxed{\tilde{c}_2 h^2} + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots + \tilde{c}_{2K} h^{2K} + O(h^{2K+1})$$

Original error – O(h^2)

Suggests the following strategy:

$$\begin{aligned}(1)\ I &= Q_{CT(h)} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots \\ (2)\ I &= Q_{CT(2h)} + \tilde{c}_2 (2h)^2 + \tilde{c}_4 (2h)^4 + \tilde{c}_6 (2h)^6 + \ldots\end{aligned}$$

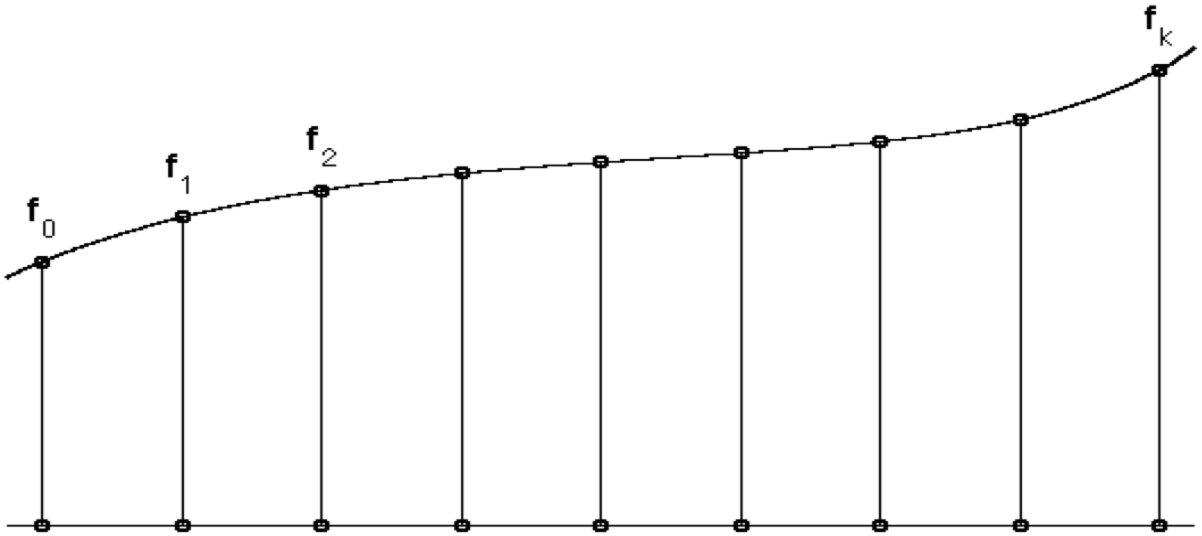
Take 4 ×(1)-(2) (*eliminate* $O(h^2)$ *term*):

$$\begin{aligned}4I - I &= 4Q_{CT(h)} - Q_{CT(2h)} + c_4' h^4 + c_6' h^6 + \ldots \\ I &= \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + \ldots \\ &= Q_{S(2h)} + \boxed{\hat{c}_4 h^4} + \hat{c}_6 h^6 + h.o.t.\end{aligned}$$

Here, $\quad Q_{S(2h)} \equiv \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)}$

New error – O(h^4)

Richardson Extrapolation + Composite Trapezoidal Rule



		$a = x_0$	x_1	x_2	x_3	x_4	$\dots$	$\dots$	$\dots$	$x_k = b$	
$h:$	w_j:	$\frac{h}{2}$	h	h	h	h	h	h	$\cdots$	$\frac{h}{2}$	
$2h:$	$\tilde{w}_j$:	$\frac{2h}{2}$	0	$2h$	0	$2h$	0	$2h$	$\cdots$	$\frac{2h}{2}$	$(k \text{ even})$
$\frac{4}{3}w_j - \frac{1}{3}\tilde{w}_j$:		$\frac{h}{3}$	$\frac{4h}{3}$	$\frac{2h}{3}$	$\frac{4h}{3}$	$\frac{2h}{3}$	$\frac{4h}{3}$	$\frac{2h}{3}$	$\cdots$	$\frac{2h}{3}$	

- ***Richardson + Composite Trapezoidal = Composite Simpson***
- But we never compute it this way.
- Just use $Q_{CS} = (4\, Q_{CT(h)} – Q_{CT(2h)}) / 3$
- No new function evaluations required!

Repeated Richardson Extrapolation

(Romberg Integration)

- We can repeat the extrapolation process to get rid of the $O(h^4)$ term.
- And repeat again, to get rid of $O(h^6)$ term.

$$
\begin{aligned}
T_{k,0} &= \text{Trapezoidal rule with} h = (b-a)/2^k \\
T_{k,j} &= \frac{4^j\, T_{k,j-1} - T_{k-1,j-1}}{4^j - 1}
\end{aligned}
$$

$$
\begin{array}{lcccc}
h & T_{0,0} & & & \\
h/2 & T_{1,0} & T_{1,1} & & \\
h/4 & T_{2,0} & T_{2,1} & T_{2,2} & \\
h/8 & T_{3,0} & T_{3,1} & T_{3,2} & T_{3,3} \\
 & O(h^2) & O(h^4) & O(h^6) & O(h^8)
\end{array}
$$

- Idea works just as well if errors are of form $c_1h + c_2h^2 + c_3h^3 + \ldots$, but tabular form would involve 2^j instead of 4^j

Repeated Richardson Extrapolation
(Romberg Integration)

- We can repeat the extrapolation process to get rid of the O(h^4) term.
- And repeat again, to get rid of O(h^6) term.

```
exact = exp(1)-1;
n=16;
x=0:n; x=x'/n; h=x(2)-x(1); f=exp(cos(5*x)); f=exp(-x.*x); f=exp(x);

T=zeros(5,5);

T(1,1)=16*h*(sum(f(1:16:end))-(f(1)+f(n+1))/2); % n must be divisible by 16
T(2,1)= 8*h*(sum(f(1: 8:end))-(f(1)+f(n+1))/2);
T(3,1)= 4*h*(sum(f(1: 4:end))-(f(1)+f(n+1))/2);
T(4,1)= 2*h*(sum(f(1: 2:end))-(f(1)+f(n+1))/2);
T(5,1)= 1*h*(sum(f(1: 1:end))-(f(1)+f(n+1))/2); % Finest approximation

format compact; format longe

T(:,1:1)

for j=2:5; for i=j:5; j1=j-1;
  T(i,j)=((4^j1)*T(i,j-1)-T(i-1,j-1))/(4^j1 - 1);
end;end;
```

Richardson Example

$$I = \int_0^1 e^x \, dx$$

Initial values, all created from same 17 values of f(x).

```
1.859140914229523
1.753931092464825
1.727221904557517
1.720518592164302
1.718841128579994
```

Using these 5 values, we build the table (extrapolate) to get more precision.

None	Round 1	Round 2	Round 3	Round 4
1.859140914229				
1.753931092464	1.718861151876			
1.727221904557	1.718318841921	1.718282687924		
1.720518592164	1.718284154699	1.718281842218	1.718281828794	
1.718841128579	1.718281974051	1.718281828675	1.718281828460	1.718281828459

Richardson Example

Error for Richardson Extrapolation (aka Romberg integration)

```
1/h    None         Round 1     Round 2      Round 3      Round 4
 1   1.4086e-01
 2   3.5649e-02   5.7932e-04
 4   8.9401e-03   3.7013e-05   8.5947e-07
 8   2.2368e-03   2.3262e-06   1.3759e-08   3.3549e-10
16   5.5930e-04   1.4559e-07   2.1631e-10   1.3429e-12   3.2419e-14
       O(h^2)       O(h^4)       O(h^6)       O(h^8)       O(h^10)
```

Gauss Quadrature Results

```
 n       Qn           E
 2   1.8591e+00   1.4086e-01
 3   1.7189e+00   5.7932e-04
 4   1.7183e+00   1.0995e-06
 5   1.7183e+00   1.1666e-09
 6   1.7183e+00   7.8426e-13
 7   1.7183e+00            0
```

Richardson Extrapolation

- In many problems, such as numerical integration or differentiation, approximate value for some quantity is computed based on some step size
- Ideally, we would like to obtain limiting value as step size approaches zero, but we cannot take step size arbitrarily small because of excessive cost or rounding error
- Based on values for nonzero step sizes, however, we may be able to estimate value for step size of zero
- One way to do this is called *Richardson extrapolation*

Richardson Extrapolation, continued

- Let $F(h)$ denote value obtained with step size h
- If we compute value of F for some nonzero step sizes, and if we know theoretical behavior of $F(h)$ as $h \to 0$, then we can extrapolate from known values to obtain approximate value for $F(0)$
- Suppose that

$$F(h) = a_0 + a_1 h^p + \mathcal{O}(h^r)$$

  as $h \to 0$ for some p and r, with $r > p$
- Assume we know values of p and r, but not a_0 or a_1 (indeed, $F(0) = a_0$ is what we seek)

Richardson Extrapolation, continued

- Suppose we have computed F for two step sizes, say h and h/q for some positive integer q

- Then we have

$$\begin{aligned} F(h) &= a_0 + a_1 h^p + \mathcal{O}(h^r) \\ F(h/q) &= a_0 + a_1 (h/q)^p + \mathcal{O}(h^r) = a_0 + a_1 q^{-p} h^p + \mathcal{O}(h^r) \end{aligned}$$

- This system of two linear equations in two unknowns a_0 and a_1 is easily solved to obtain

$$a_0 = F(h) + \frac{F(h) - F(h/q)}{q^{-p} - 1} + \mathcal{O}(h^r)$$

- Accuracy of improved value, a_0, is $\mathcal{O}(h^r)$

Richardson Extrapolation, continued

- Extrapolated value, though improved, is still only approximate, not exact, and its accuracy is still limited by step size and arithmetic precision used

- If $F(h)$ is known for several values of h, then extrapolation process can be repeated to produce still more accurate approximations, up to limitations imposed by finite-precision arithmetic

Example: Richardson Extrapolation

- Use Richardson extrapolation to improve accuracy of finite difference approximation to derivative of function $\sin(x)$ at $x = 1$
- Using first-order accurate forward difference approximation, we have

$$F(h) = a_0 + a_1 h + \mathcal{O}(h^2)$$

so $p = 1$ and $r = 2$ in this instance

- Using step sizes of $h = 0.5$ and $h/2 = 0.25$ (i.e., $q = 2$), we obtain

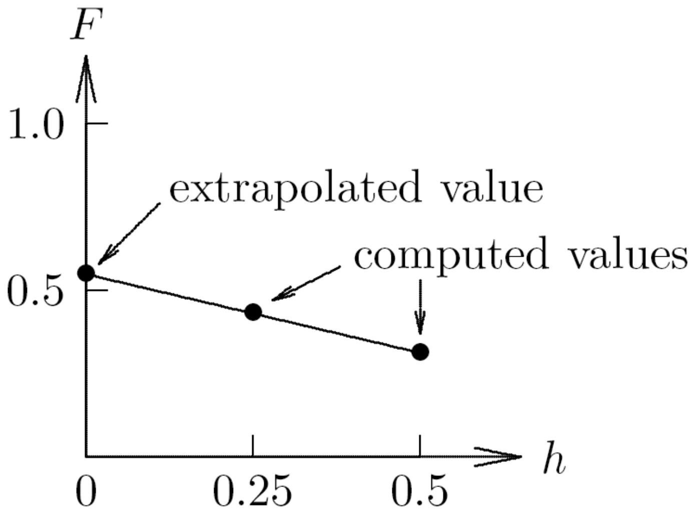
$$\begin{aligned} F(h) &= \frac{\sin(1.5) - \sin(1)}{0.5} = 0.312048 \\ F(h/2) &= \frac{\sin(1.25) - \sin(1)}{0.25} = 0.430055 \end{aligned}$$

Example, continued

- Extrapolated value is then given by

$$F(0) = a_0 = F(h) + \frac{F(h) - F(h/2)}{(1/2) - 1} = 2F(h/2) - F(h) = 0.548061$$

- For comparison, correctly rounded result is $\cos(1) = 0.540302$



< interactive example >

Example: Romberg Integration

- As another example, evaluate

$$\int_0^{\pi/2} \sin(x)\, dx$$

- Using composite trapezoid rule, we have

$$F(h) = a_0 + a_1 h^2 + \mathcal{O}(h^4)$$

  so $p = 2$ and $r = 4$ in this instance

- With $h = \pi/2$, $F(h) = F(\pi/2) = 0.785398$

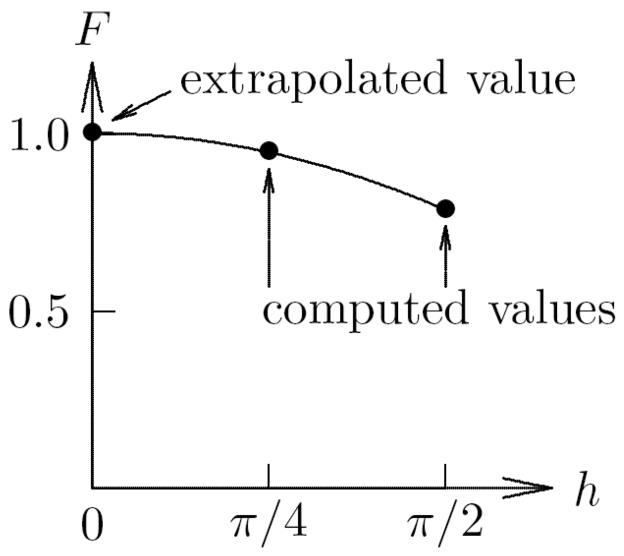
- With $q = 2$, $F(h/2) = F(\pi/4) = 0.948059$

Example, continued

Extrapolated value is then given by

$$F(0) = a_0 = F(h) + \frac{F(h) - F(h/2)}{2^{-2} - 1} = \frac{4F(h/2) - F(h)}{3} = 1.002280$$

which is substantially more accurate than values previously computed (exact answer is 1)



< interactive example >

Romberg Integration

- Continued Richardson extrapolations using composite trapezoid rule with successively halved step sizes is called *Romberg integration*
- It is capable of producing very high accuracy (up to limit imposed by arithmetic precision) for very smooth integrands
- It is often implemented in automatic (though nonadaptive) fashion, with extrapolations continuing until change in successive values falls below specified error tolerance

Double Integrals

Approaches for evaluating double integrals include

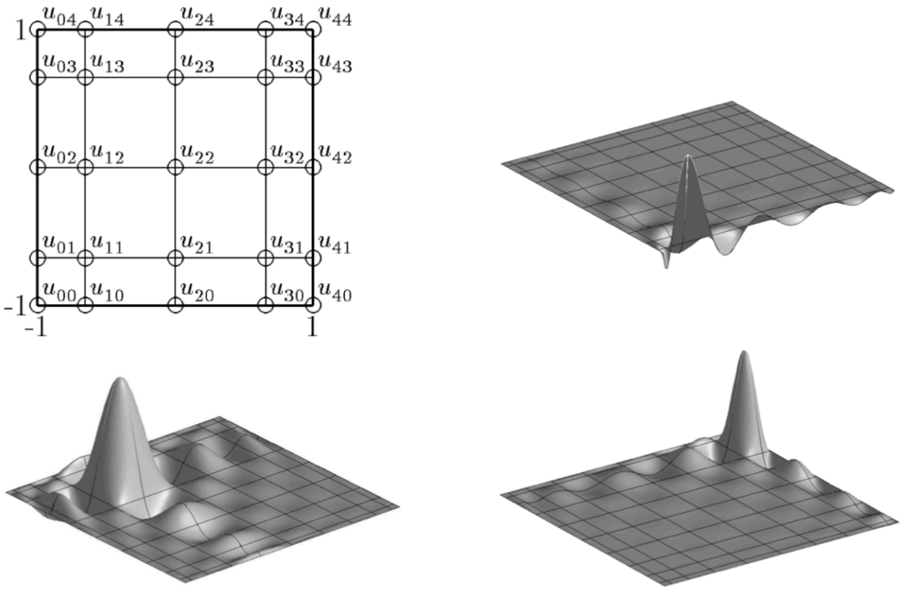
- Use automatic one-dimensional quadrature routine for each dimension, one for outer integral and another for inner integral
- Use product quadrature rule resulting from applying one-dimensional rule to successive dimensions
- Use non-product quadrature rule for regions such as triangles

Tensor-Product Integration

- As with interpolation, if domain can be expressed in rectangular form, then can use tensor-products of 1D interpolants:

$$Q_n \;=\; \sum_{j=0}^{N} \sum_{i=0}^{N} w_i \, w_j f(\xi_i, \xi_j)$$

- The weights are just the 1D quadrature weights.
- More complex domains handled by mappings of [-1,1] to more general shapes and/or by using composite multidomain integration.


u_{04} u_{14} u_{24} u_{34} u_{44}
u_{03} u_{13} u_{23} u_{33} u_{43}
u_{02} u_{12} u_{22} u_{32} u_{42}
u_{01} u_{11} u_{21} u_{31} u_{41}
u_{00} u_{10} u_{20} u_{30} u_{40}
1
-1
-1
1

Multiple Integrals

- To evaluate multiple integrals in higher dimensions, only generally viable approach is Monte Carlo method

- Function is sampled at n points distributed randomly in domain of integration, and mean of function values is multiplied by area (or volume, etc.) of domain to obtain estimate for integral

- Error in estimate goes to zero as $1/\sqrt{n}$, so to gain one additional decimal digit of accuracy requires increasing n by factor of 100

- For this reason, Monte Carlo calculations of integrals often require millions of evaluations of integrand

< interactive example >

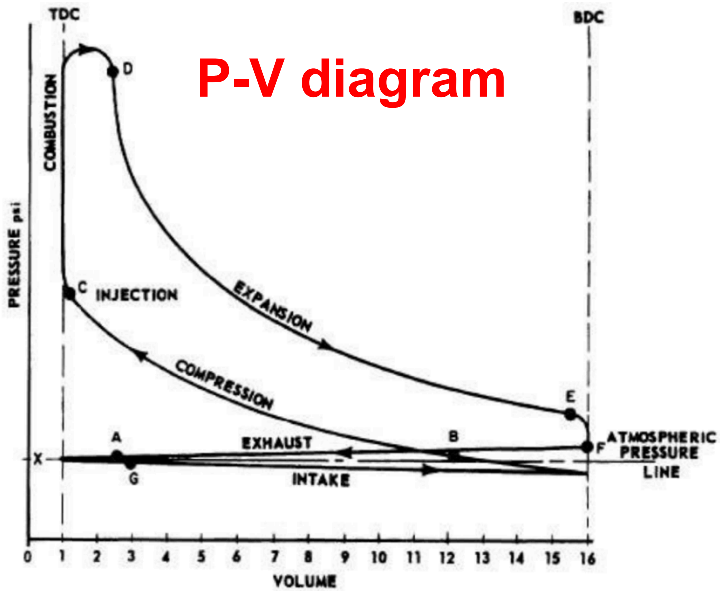
Multiple Integrals, continued

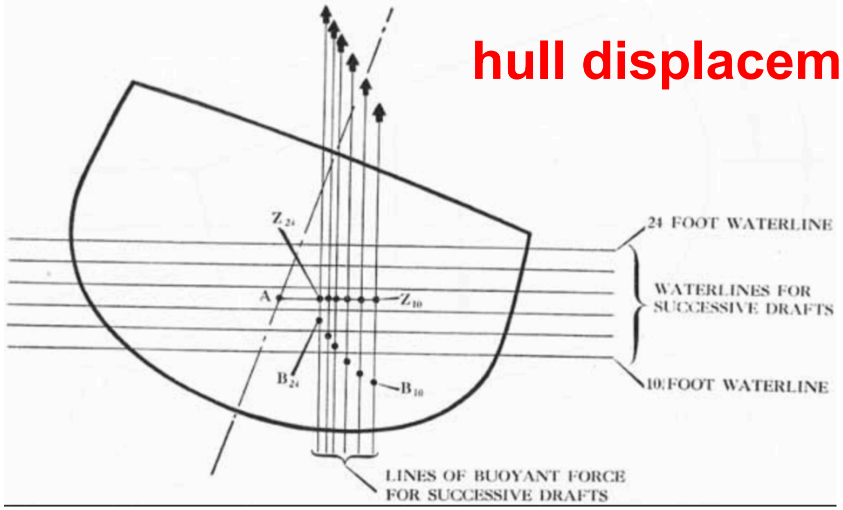
- Monte Carlo method is not competitive for dimensions one or two, but strength of method is that its convergence rate is independent of number of dimensions
- For example, one million points in six dimensions amounts to only ten points per dimension, which is much better than any type of conventional quadrature rule would require for same level of accuracy

< interactive example >

Mechanical Integration

- In the 19th Century, integration was required for many engineering disciplines, including engine and ship design.
  - For engine design, knowing the area swept out in the P-V diagram can tell how much energy is produced during a single cycle.
  - For ship design, knowing the displacement of the hull at different angles of roll tells about the buoyancy. Knowing the moments is important for stability.

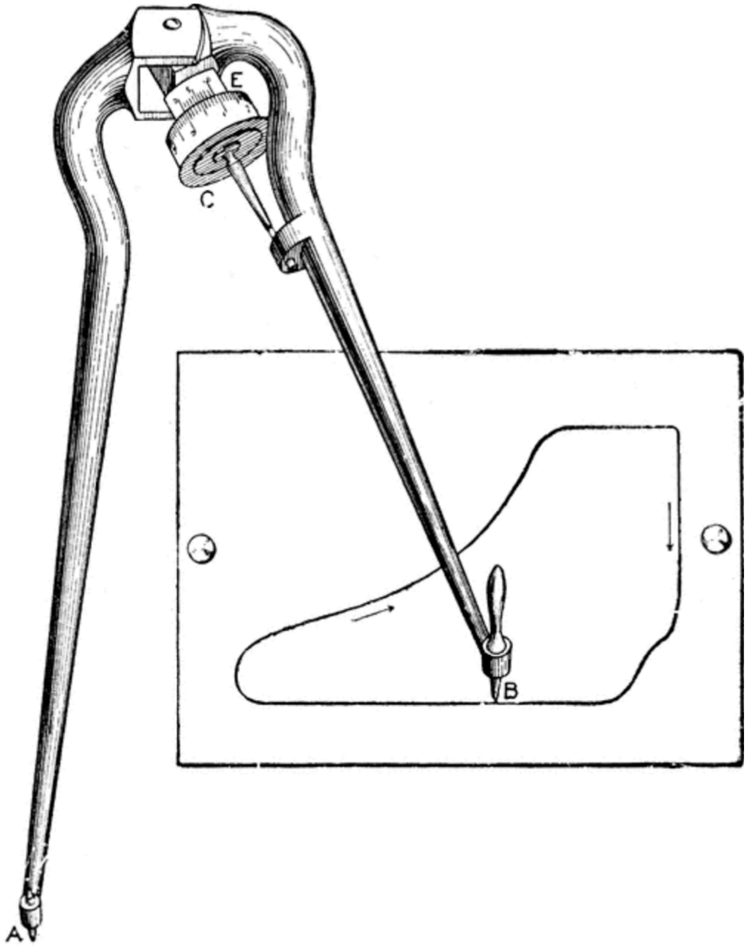

P-V diagram


hull displacement
Z24
A
Z10
B24
B10
24 FOOT WATERLINE
WATERLINES FOR SUCCESSIVE DRAFTS
10 FOOT WATERLINE
LINES OF BUOYANT FORCE FOR SUCCESSIVE DRAFTS

Mechanical Integration

- In the mid-19th century, mathematicians developed mechanical integrators that can easily find the area under curves to remarkable precision.

- They generally go by the name of planimeters or polar planimeters.

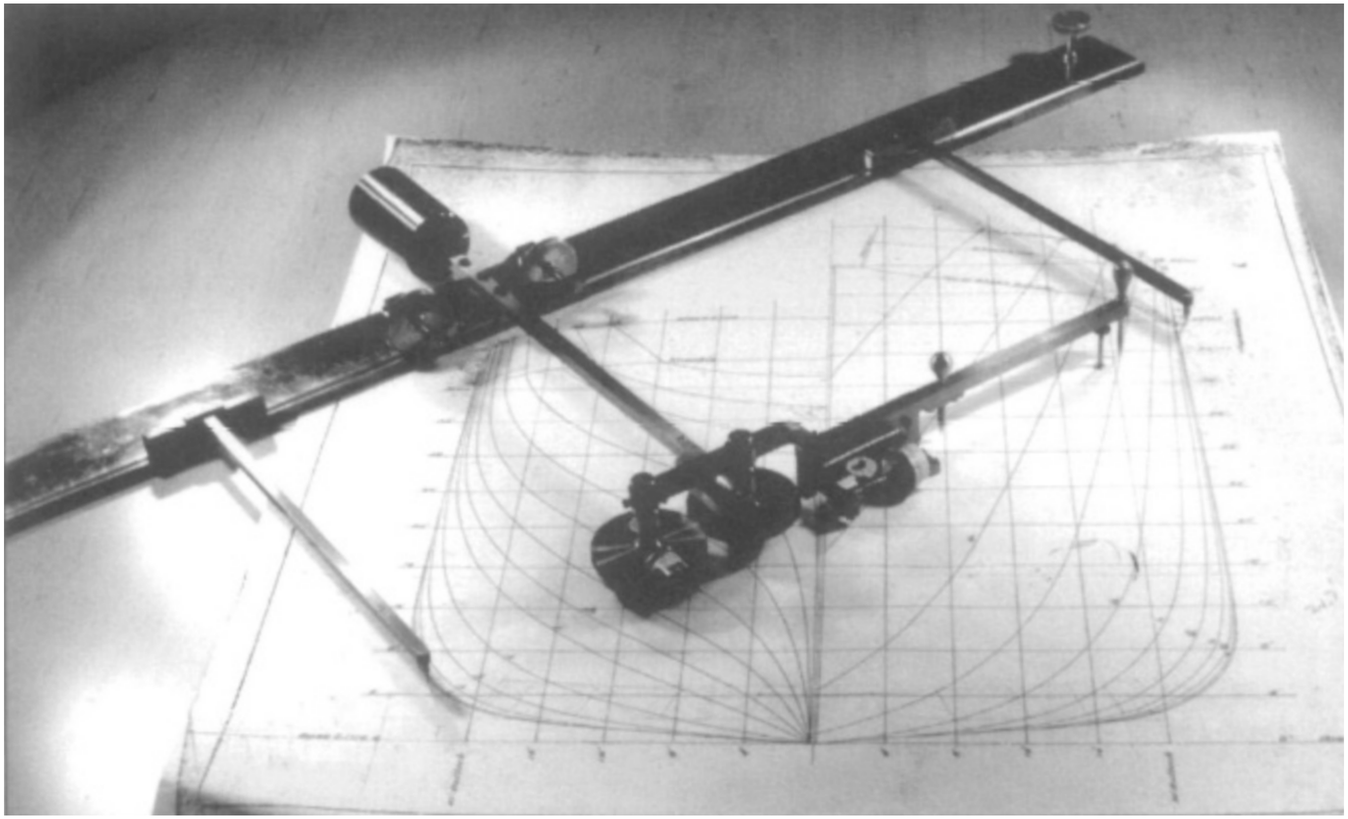


Mechanical Integration

- Even more sophisticated integrators could compute first moments – important for many design problems.

- Later, we'll see mechanical integrators for Fourier transforms that were responsible for the discovery of the Gibbs phenomenon. U of I had an important role in this discovery.

$$\int_A y\, dA$$



Basic Idea of Planimeter

- **Let**

$$\begin{aligned} \mathbf{u} &= (u,v) = \textit{vector field} \\ \nabla\cdot\mathbf{u} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = \textit{scalar field.} \end{aligned}$$

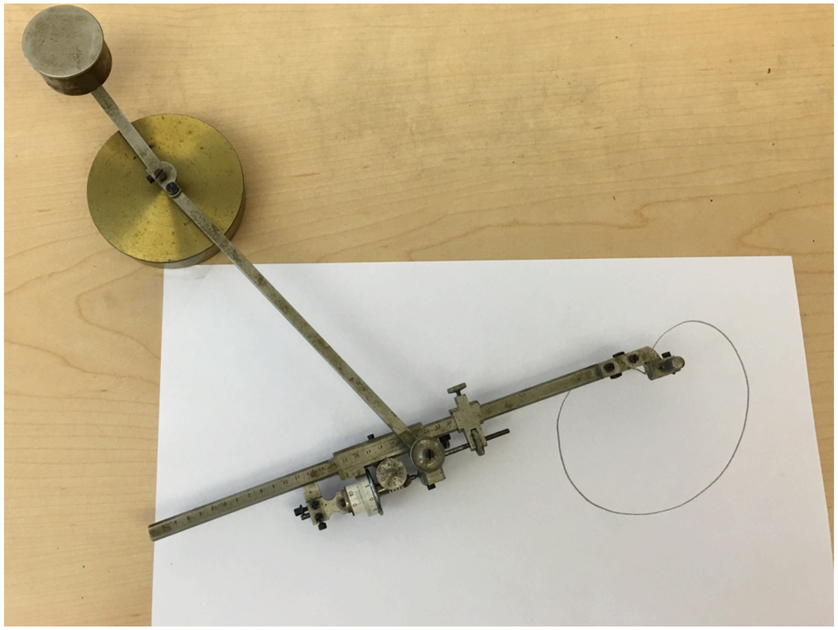
- **Divergence Theorem:**

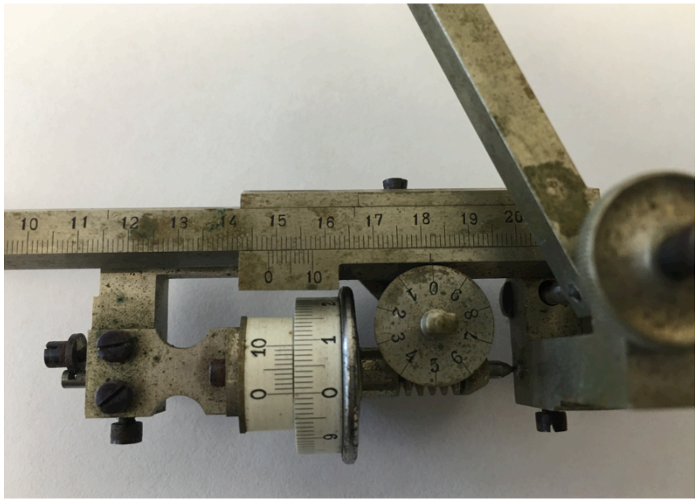
$$\int_A \nabla\cdot\mathbf{u}\,dA = \int_S \mathbf{u}\cdot\hat{\mathbf{n}}\,dS.$$

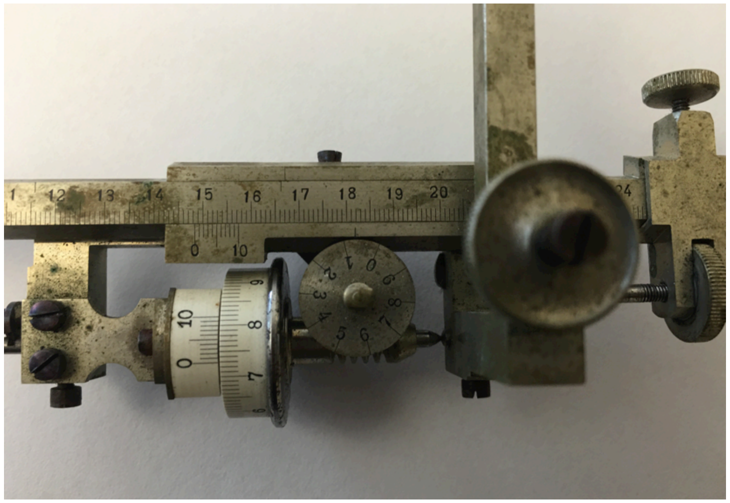
- For the planimeter, consider $\mathbf{u} = (0, y)$:

$$\begin{aligned} \nabla\cdot\mathbf{u} &= \frac{\partial}{\partial x}0 + \frac{\partial}{\partial y}y \equiv 1 \\ A &= \int_A 1\,dA = \int_S y\mathbf{e}_y\cdot\hat{\mathbf{n}}\,dS. \end{aligned}$$

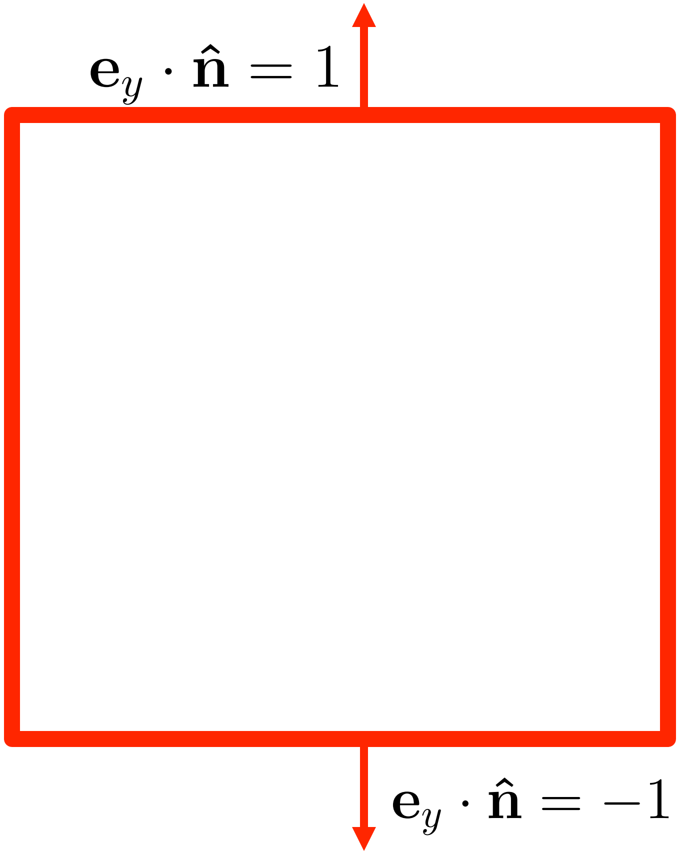
Planimeter Operation







$$A = \int_A 1\, dA = \int_S y\mathbf{e}_y \cdot \hat{\mathbf{n}}\, dS.$$



$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 1$

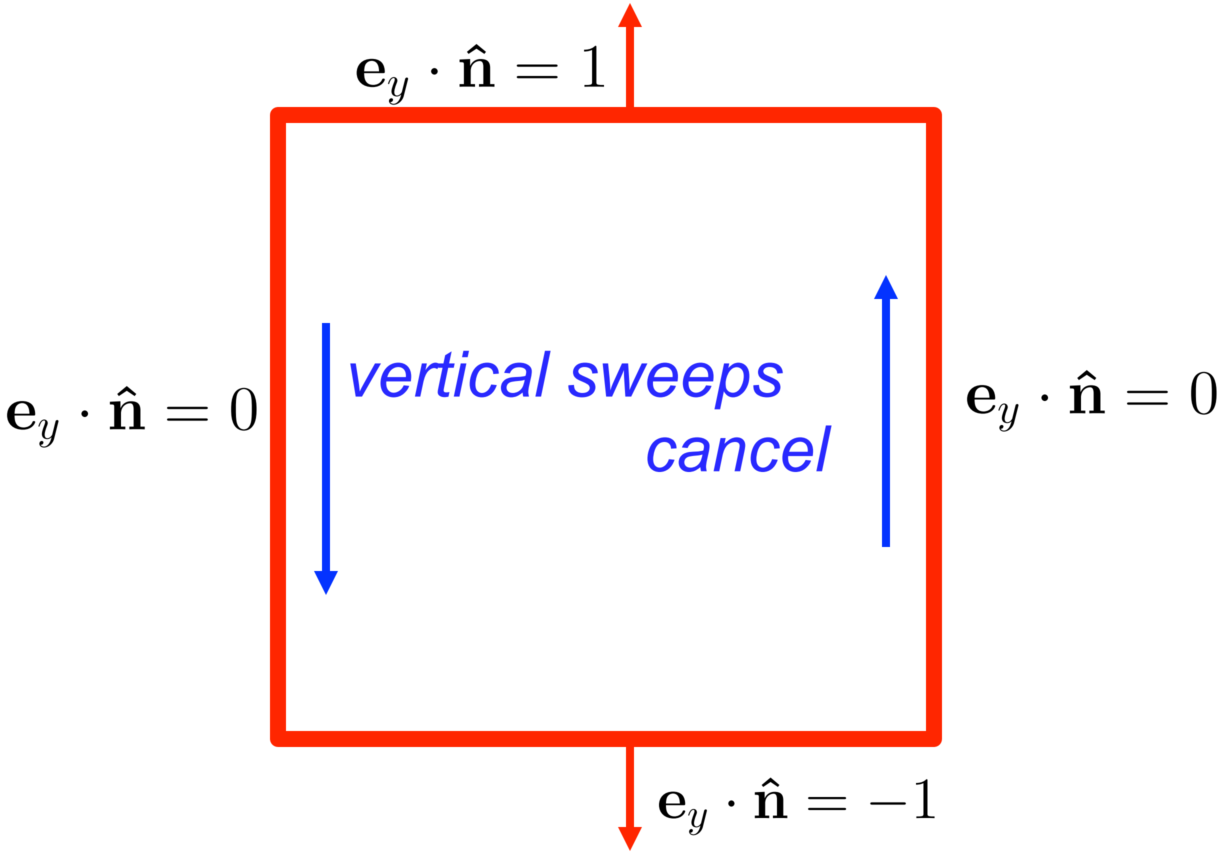
$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 0$

$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 0$

$\mathbf{e}_y \cdot \hat{\mathbf{n}} = -1$

Action of Planimeter

$$A \;=\; \int_A 1\,dA \;=\; \int_S y\mathbf{e}_y \cdot \hat{\mathbf{n}}\,dS.$$



$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 1$

$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 0$

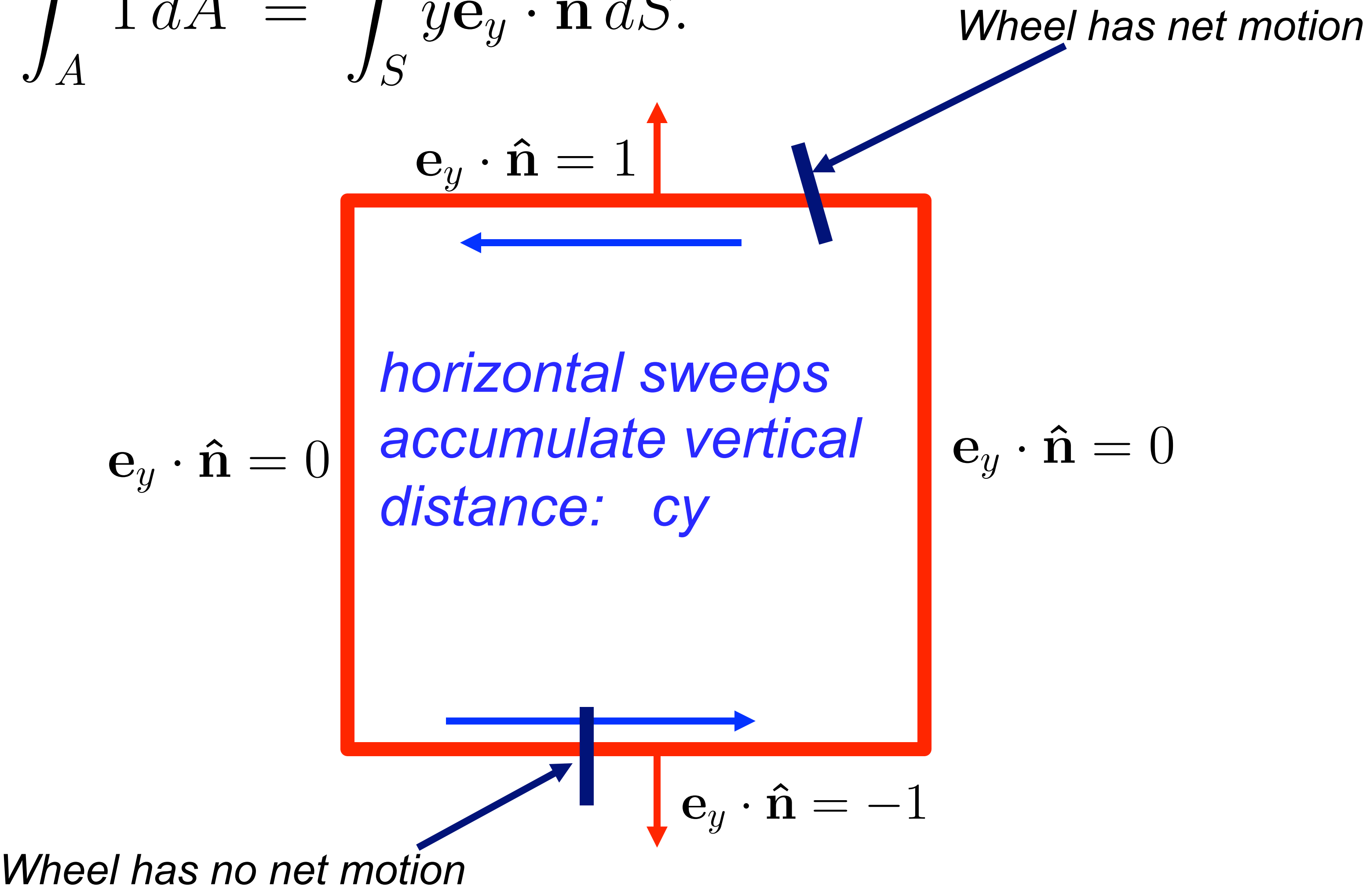
vertical sweeps cancel

$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 0$

$\mathbf{e}_y \cdot \hat{\mathbf{n}} = -1$

Action of Planimeter

$$A \;=\; \int_A 1\,dA \;=\; \int_S y\mathbf{e}_y \cdot \hat{\mathbf{n}}\,dS.$$

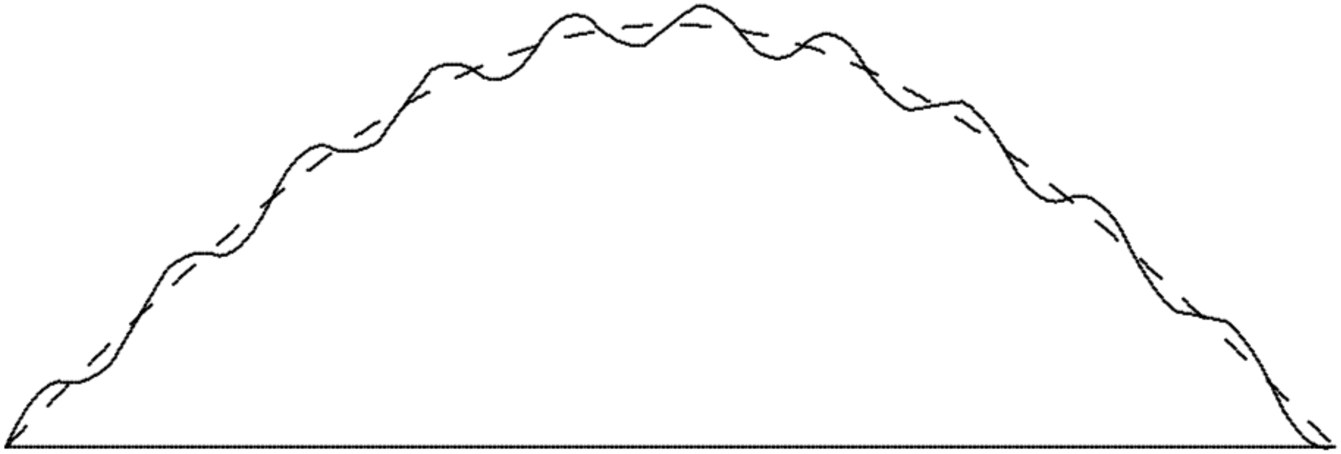


Wheel has net motion
$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 1$
$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 0$
$\mathbf{e}_y \cdot \hat{\mathbf{n}} = 0$
$\mathbf{e}_y \cdot \hat{\mathbf{n}} = -1$
Wheel has no net motion

horizontal sweeps accumulate vertical distance: cy

Numerical Differentiation

- Differentiation is inherently sensitive, as small perturbations in data can cause large changes in result
- Differentiation is inverse of integration, which is inherently stable because of its smoothing effect
- For example, two functions shown below have very similar definite integrals but very different derivatives



Numerical Differentiation, continued

- To approximate derivative of function whose values are known only at discrete set of points, good approach is to fit some smooth function to given data and then differentiate approximating function

- If given data are sufficiently smooth, then interpolation may be appropriate, but if data are noisy, then smoothing approximating function, such as least squares spline, is more appropriate

Numerical Differentiation Techniques

Three common approaches for deriving formulas

- ❑ Taylor series
- ❑ Taylor series + Richardson extrapolation
- ❑ Differentiate Lagrange interpolants
  - ❑ Readily programmed, see, e.g., Fornberg’s spectral methods text.

Using Taylor Series to Derive Difference Formulas

Taylor Series:

$$
\begin{aligned}
(1) \quad f_{j+1} &= f_j + hf_j' + \frac{h^2}{2}f_j'' + \frac{h^3}{3!}f_j''' + \frac{h^4}{4!}f^{(4)}(\xi_+) \\
(2) \quad f_j &= f_j \\
(3) \quad f_{j-1} &= f_j - hf_j' + \frac{h^2}{2}f_j'' - \frac{h^3}{3!}f_j''' + \frac{h^4}{4!}f^{(4)}(\xi_-)
\end{aligned}
$$

Approximation of $f_j' := f'(x_j)$:

$$\frac{1}{h}\,[(1)-(2)] : \quad \frac{f_{j+1}-f_j}{h} = f_j' + \frac{h}{2}f_j'' + h.o.t.$$

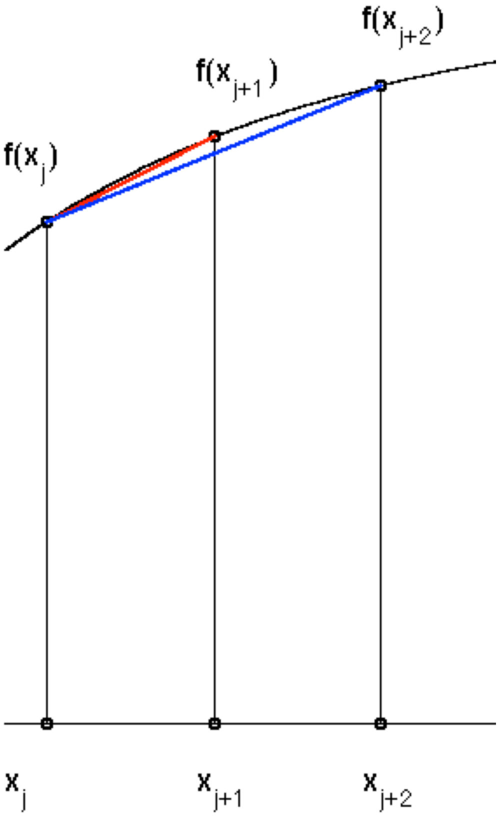
or

$$\frac{1}{2h}\,[(1)-(3)] : \quad \frac{f_{j+1}-f_{j-1}}{2h} = f_j' + \frac{h^2}{3!}f_j''' + h.o.t.$$

Richardson Extrapolation

$$\delta_h : \quad \frac{f_{j+1} - f_j}{h} = f'_j + c_1 h + c_2 h^2 + c_3 h^3 + \cdots$$

$$\delta_{2h} : \quad \frac{f_{j+2} - f_j}{2h} = f'_j + c_1 2h + c_2 4h^2 + c_3 8h^3 + \cdots$$



$$\begin{aligned} 2\delta_h - \delta_{2h} &= \frac{4f_{j+1} - 4f_j}{2h} - \frac{f_{j+2} - f_j}{2h} \\ &= \frac{-3f_j + 4f_{j+1} - f_{j+2}}{2h} \\ &= f'_j + \tilde{c}_2 h^2 + \tilde{c}_3 h^3 + \cdots \end{aligned}$$

- Formula is improved from O(h) to O(h^2)

Finite Difference Approximations

- Given smooth function $f\colon \mathbb{R} \to \mathbb{R}$, we wish to approximate its first and second derivatives at point x
- Consider Taylor series expansions

$$\begin{aligned} f(x+h) &= f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \cdots \\ f(x-h) &= f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \cdots \end{aligned}$$

- Solving for $f'(x)$ in first series, obtain *forward difference approximation*

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)}{2}h + \cdots \approx \frac{f(x+h) - f(x)}{h}$$

which is first-order accurate since dominant term in remainder of series is $\mathcal{O}(h)$

Finite Difference Approximations, continued

- Similarly, from second series derive *backward difference approximation*

$$\begin{aligned} f'(x) &= \frac{f(x) - f(x-h)}{h} + \frac{f''(x)}{2}h + \cdots \\ &\approx \frac{f(x) - f(x-h)}{h} \end{aligned}$$

  which is also first-order accurate

- Subtracting second series from first series gives *centered difference approximation*

$$\begin{aligned} f'(x) &= \frac{f(x+h) - f(x-h)}{2h} - \frac{f'''(x)}{6}h^2 + \cdots \\ &\approx \frac{f(x+h) - f(x-h)}{2h} \end{aligned}$$

  which is second-order accurate

Finite Difference Approximations, continued

- Adding both series together gives *centered difference approximation* for second derivative

$$\begin{aligned} f''(x) &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{f^{(4)}(x)}{12}h^2 + \cdots \\ &\approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \end{aligned}$$

  which is also second-order accurate

- Finite difference approximations can also be derived by polynomial interpolation, which is less cumbersome than Taylor series for higher-order accuracy or higher-order derivatives, and is more easily generalized to unequally spaced points

Example

- Use Richardson extrapolation to derive 5 point high-order approximation to f'(x_j)

Derivative Matrices

- It is often convenient, particularly with interpretive languages like Python and matlab, or in multiple space dimensions, to effect interpolation and differentiation through matrix-vector, or matrix-matrix, products.

- Consider the question of evaluating the $p(x)$ at a set of m points, $\tilde{\mathbf{x}} = [\tilde{x}_1 \; \tilde{x}_2 \; \cdots \tilde{x}_m]^T$, where $p(x_j) := f(x_j) = f_j$ is an interpolant (usually, but not necessarily, a polynomial).

- Define
$$\tilde{f}_i \;\; := \;\; \sum_{j=1}^{n} l_j(\tilde{x}_i) f_j.$$

- Here, we have introduced the *interpolation matrix*,
$$\hat{J}_{ij} \;\; := \;\; l_j(\tilde{x}_i),$$
which allows this operation to be expressed succinctly as $\tilde{\mathbf{f}} = \hat{J}\mathbf{f}$.

- Such an expression is particularly useful for Python and matlab because they support *compiled* (i.e., efficient) versions of matrix-vector products that will run hundreds of times faster than a basic *for loop*.

- It is also useful for higher-dimensional interpolation since we can (for example) express a coarse-to-fine interpolation in 2D as $\tilde{\mathbf{f}} = J\mathbf{f} = (\hat{J} \otimes \hat{J})\mathbf{f}$, where
$$\tilde{f}_{ij} \;\; = \;\; p(\tilde{x}_i, \tilde{y}_j) \;\; = \;\; \sum_{q=1}^{n} \sum_{p=1}^{n} l_p(x_i)\, l_q(y_j)\, f_{pq}$$
is the interpolant from an $n \times n$ input array to an $m \times m$ output array.

- **Recall** the *very* efficient matrix form of this expression: $\tilde{F} = \hat{J} F \hat{J}^T$.

Derivative Matrices

- We can similarly construct *differentiation* matrices.
- If $g(x) := f'(x)$ and we only have values $f(x_j)$, we take $p'(x)$ to be our estimate of $g(x)$.
- We are generally interested in evaluating $p'(x)$ at some set of output points $\tilde{\mathbf{x}} = [\tilde{x}_1 \; \tilde{x}_2 \; \cdots \tilde{x}_m]^T$.
- Assuming differentiability of the Lagrange cardinal interpolants yields

$$\tilde{g}_i \quad := \quad \sum_{j=1}^{n} \left. \frac{dl_j}{dx} \right|_{\tilde{x}_i} f_j,$$

  or $\tilde{\mathbf{g}} = \tilde{D}\mathbf{f}$, with

$$\tilde{D}_{ij} \quad := \quad \left. \frac{dl_j}{dx} \right|_{\tilde{x}_i} .$$

- (**Q:** For which common set of of Lagrange interpolants is differentiability problematic?)

Derivative Matrices

- For *polynomial* interpolation, we can cast the mapping $f(x_j) \longrightarrow p'(\tilde{x}_i)$ as differentiation followed by interpolation (or vice versa).

- Define the differentiation matrix,

$$\hat{D}_{ij} \;\; := \;\; \left. \frac{dl_j}{dx} \right|_{x_i},$$

  which maps f_j to $p'(x_i)$.

- Given that $p'(x) \in \mathbb{P}_{n-2}$, we have $\tilde{D} = \tilde{J}\hat{D}$, because $\tilde{J}$ is exact for any input in $\mathbb{P}_{n-1}$.

- The general differentiation problem is reduced to evaluating $\hat{D}_{ij}$.

Derivative Matrices

- Consider

$$\begin{aligned}
p(x) &= \sum_{j=1}^{n} l_j(x) f_j \\
p'(x) &= \sum_{j=1}^{n} \frac{dl_j}{dx} f_j \\
p'(x_i) &= \sum_{j=1}^{n} \left.\frac{dl_j}{dx}\right|_{x_i} f_j \;=\; \sum_{j=1}^{n} d_{ij} f_j \;\Longrightarrow\; \mathbf{p}' = \hat{D}\mathbf{f}.
\end{aligned}$$

- Recall Lagrange cardinal polynomial,

$$\begin{aligned}
l_j(x) &= c_j(x-x_1)(x-x_2)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n) \\
&= c_j \prod_{k \neq j} (x-x_k) \\
&= q_n(x)\frac{c_j}{x-x_j}.
\end{aligned}$$

- Derivative:

$$
\begin{array}{rccccccccccc}
\frac{dl_j(x)}{dx} = c_j & [& & (x-x_2) & (x-x_3) & \cdots & (x-x_{j-1}) & (x-x_{j+1}) & \cdots & (x-x_n) & \\
& + & (x-x_1) & & (x-x_3) & \cdots & (x-x_{j-1}) & (x-x_{j+1}) & \cdots & (x-x_n) & \\
& + & (x-x_1) & (x-x_2) & & \cdots & (x-x_{j-1}) & (x-x_{j+1}) & \cdots & (x-x_n) & \\
& & & & & \vdots & & & & & \\
& + & (x-x_1) & (x-x_2) & (x-x_3) & \cdots & (x-x_{j-2}) & (x-x_{j+1}) & \cdots & (x-x_n) & \\
& + & (x-x_1) & (x-x_2) & (x-x_3) & \cdots & (x-x_{j-1}) & (x-x_{j+2}) & \cdots & (x-x_n) & \\
& & & & & \vdots & & & & & \\
& + & (x-x_1) & (x-x_2) & (x-x_3) & \cdots & (x-x_{j-1}) & (x-x_{j+1}) & \cdots & &]
\end{array}
$$

$$
= c_j \sum_{i \neq j} \prod_{k \neq i,j} (x - x_k)
$$

$$
= l_j(x) \left[\sum_{k \neq j} \frac{1}{x - x_k} \right] = \left[\sum_{k \neq j} \frac{l_j(x)}{x - x_k} \right]
$$

- For $i \neq j$,

$$\frac{dl_j(x)}{dx} \;=\; \left[\sum_{k\neq j} \frac{l_j(x_i)}{x_i - x_k}\right] \;=\; \frac{l_j(x_i)}{x_i - x_i} \;=\; c_j \underbrace{\prod_{k\neq i,j} (x_i - x_k)}_{=:\alpha_{ij}} \;=\; c_j \alpha_{ij}$$

- Recall,

$$c_i^{-1} \;:=\; (x_i - x_1)(x_i - x_2)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n) \;=\; \prod_{k\neq i}(x_i - x_k),$$

$$\Longrightarrow \;\alpha_{ij} \;=\; \frac{c_i^{-1}}{(x_i - x_j)} \;=\; \frac{1}{c_i(x_i - x_j)}.$$

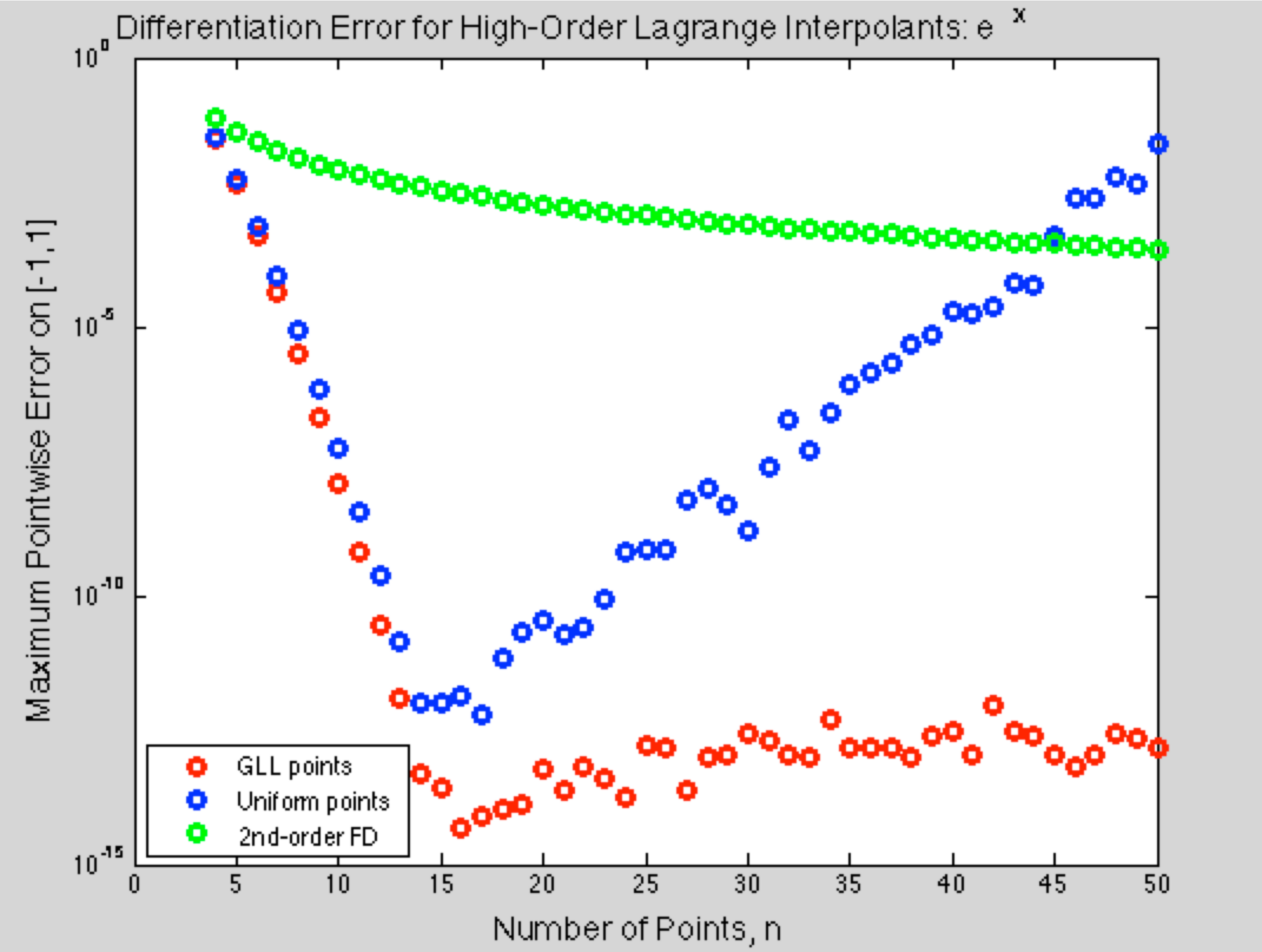
- Therefore, for $i \neq j$,

$$\boxed{d_{ij} \;=\; \frac{c_j}{c_i(x_i - x_j)}.}$$

- For $i = j$ we find simply

$$\boxed{d_{ii} \;=\; \sum_{k\neq i} \left(\frac{1}{x_i - x_k}\right).}$$

Example: testd.m, testd2.m

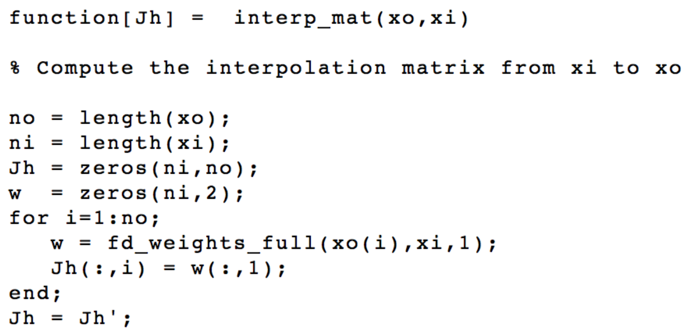


- Q: What happens if we repeatedly apply D to a vector ??

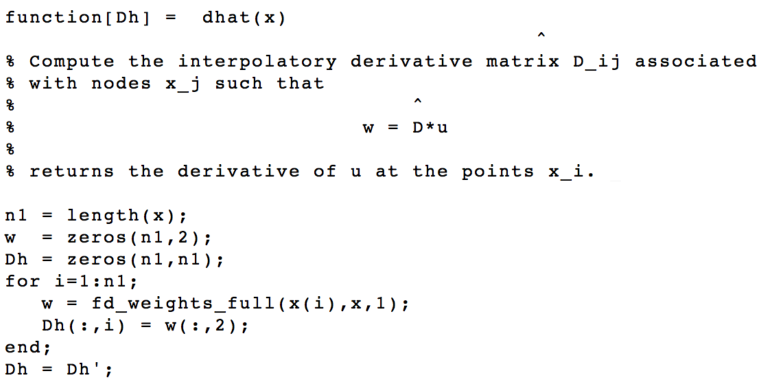
deriv_m1.m

```
function[D] = deriv_mat(x) % Interpolation matrix from x to x

n = length(x); c  = ones(n,1); D=zeros(n,n);

for i=1:n;
  for j=1:(i-1); c(i)=c(i)*(x(i)-x(j)); end;
  for j=(i+1):n; c(i)=c(i)*(x(i)-x(j)); end;
end;
c=1./c; % These are the c_i's

for j=1:n; for i=1:n; D(i,j)=x(i)-x(j); end; D(j,j)=1; end; D=1./D;
for i=1:n; D(i,i)=0; D(i,i)=sum(D(i,:)); end;

for j=1:n; for i=1:n;
   if i~=j; D(i,j) = c(j)/( c(i)*(x(i)-x(j)));end;
end;end;
```

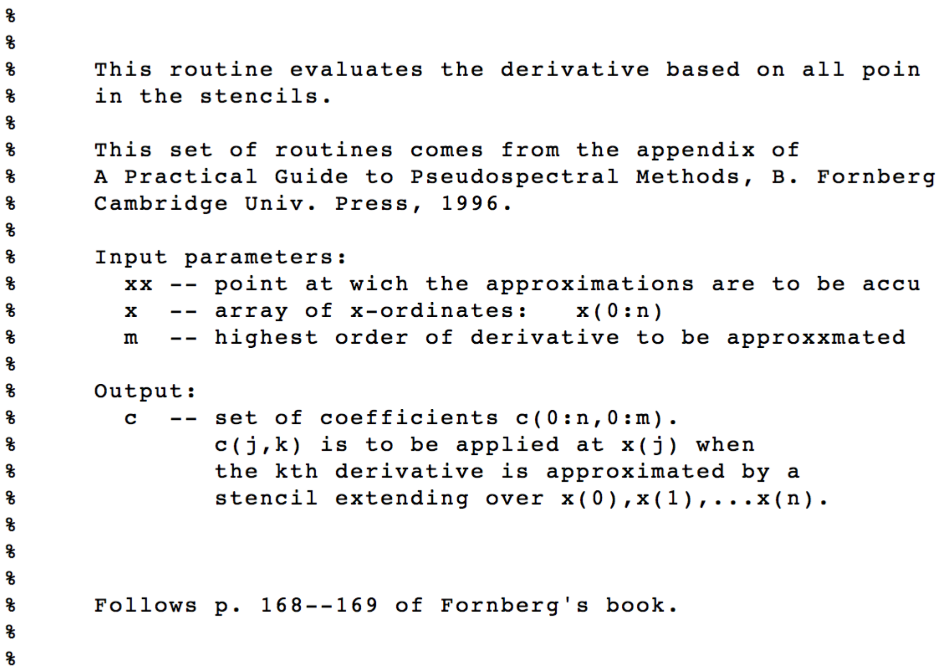
Example Matlab Code



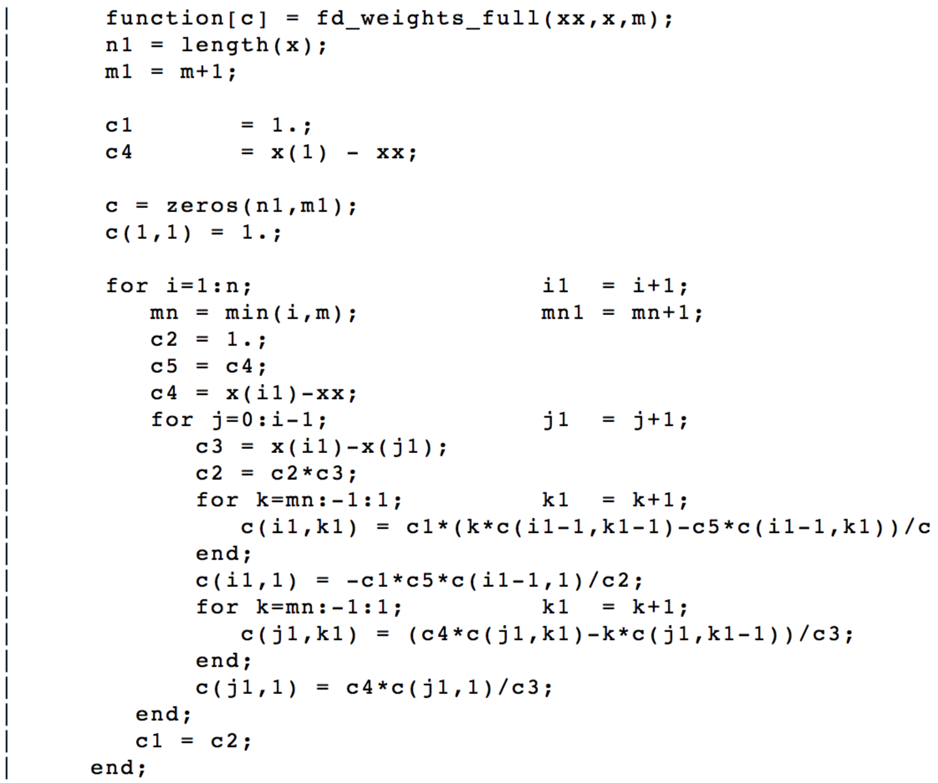
```
function[Jh] =  interp_mat(xo,xi)

% Compute the interpolation matrix from xi to xo

no = length(xo);
ni = length(xi);
Jh = zeros(ni,no);
w  = zeros(ni,2);
for i=1:no;
   w = fd_weights_full(xo(i),xi,1);
   Jh(:,i) = w(:,1);
end;
Jh = Jh';
```



```
function[Dh] =  dhat(x)
                                              ^
% Compute the interpolatory derivative matrix D_ij associated
% with nodes x_j such that
%                                         ^
%                            w = D*u
%
% returns the derivative of u at the points x_i.

n1 = length(x);
w  = zeros(n1,2);
Dh = zeros(n1,n1);
for i=1:n1;
   w = fd_weights_full(x(i),x,1);
   Dh(:,i) = w(:,2);
end;
Dh = Dh';
```



```
%
%
%     This routine evaluates the derivative based on all poin
%     in the stencils.
%
%     This set of routines comes from the appendix of
%     A Practical Guide to Pseudospectral Methods, B. Fornberg
%     Cambridge Univ. Press, 1996.
%
%     Input parameters:
%       xx -- point at wich the approximations are to be accu
%       x  -- array of x-ordinates:   x(0:n)
%       m  -- highest order of derivative to be approxxmated
%
%     Output:
%       c  -- set of coefficients c(0:n,0:m).
%             c(j,k) is to be applied at x(j) when
%             the kth derivative is approximated by a
%             stencil extending over x(0),x(1),...x(n).
%
%
%
%     Follows p. 168--169 of Fornberg's book.
%
%
```



```
|      function[c] = fd_weights_full(xx,x,m);
|      n1 = length(x);
|      m1 = m+1;
|
|      c1        = 1.;
|      c4        = x(1) - xx;
|
|      c = zeros(n1,m1);
|      c(1,1) = 1.;
|
|      for i=1:n;                     i1  = i+1;
|         mn = min(i,m);              mn1 = mn+1;
|         c2 = 1.;
|         c5 = c4;
|         c4 = x(i1)-xx;
|         for j=0:i-1;                j1  = j+1;
|            c3 = x(i1)-x(j1);
|            c2 = c2*c3;
|            for k=mn:-1:1;           k1  = k+1;
|               c(i1,k1) = c1*(k*c(i1-1,k1-1)-c5*c(i1-1,k1))/c
|            end;
|            c(i1,1) = -c1*c5*c(i1-1,1)/c2;
|            for k=mn:-1:1;           k1  = k+1;
|               c(j1,k1) = (c4*c(j1,k1)-k*c(j1,k1-1))/c3;
|            end;
|            c(j1,1) = c4*c(j1,1)/c3;
|          end;
|          c1 = c2;
|     end;
```