NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Module 6: Numerical Differentiation & Integration

Learning Objectives

- Learn numerical differentiation
  - 2-point forward-difference
  - 3-point centered-difference
  - Richardson extrapolation
- Learn numerical integration
  - Newton-Cotes Rules
  - Romberg Integration
  - Gaussian Quadrature

Sources

- Textbook (Chapter 5: Numerical Differentiation and Integration)

- Wiki: Finite difference

  https://en.wikipedia.org/wiki/Finite_difference

- Wiki: Numerical integration

  https://en.wikipedia.org/wiki/Numerical_integration

Outline

Introduction

- Basic problem (of computational calculus): to compute derivatives and integrals of functions
  - Symbolic computing:

```
>> syms x;
>> f=sin(3*x);
>> f1=diff(f)

f1=

3*cos(3*x)
```

  - Numerical computing:
    - A function may be specified as a tabulated list. E.g., a list $\{(t_1, T_1), (t_2, T_2), \ldots, (t_n, T_n)\}$ of time/temperature pairs.
    - A function may be specified as the output of an experiment or computer simulation.
    - The antiderivative has no simple expression.

Applications & Motivations

- Applications
  - Computer-aided manufacturing: precise control of motion
  - Computer animation / games / filmmaking / robotics: control speed along a path
  - Numerical computation: gradients
  - Deep learning: convolution
  - …
- Motivations:
  - In many practical applications, functions are given in discrete form. Examples: images, CT / MRI data, …

Outline

Differentiation

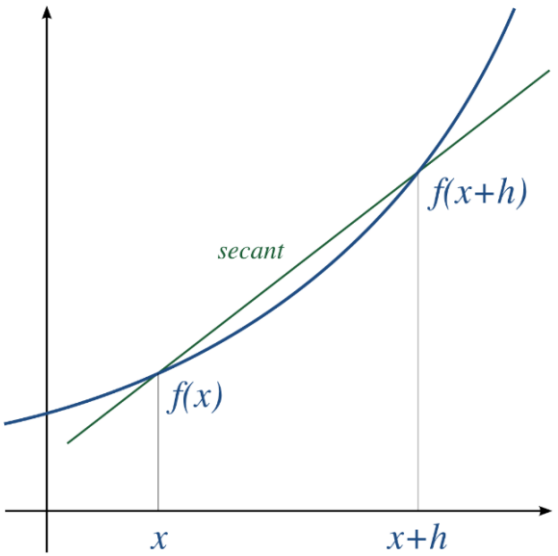
Task: Suppose we have access to the function values f(x) of some function f, but would like to know the derivative f'(x) of f at some x.

- 2-point forward-difference
- 3-point centered-difference
- Rounding error
- Richardson extrapolation

2.1. The 2-point forward-differences

- Recall the definition of the derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$



- We can expect the approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

for small values of h. This formula is called the ***2-point forward-difference*** formula.

- Geometric interpretation:
  - Fitting a linear function to f at x and x+h
  - Taking the derivative of the linear function as an approximation of f'(x).

Analysis of approximation order

- **Analysis:**

  The first order Taylor series approximation is

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(c), \quad c \in (x, x+h)$$

  from which we get

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(c).$$

  Therefore the approximation error of the 2-point forward-difference formula is $-\frac{h}{2} f''(c)$, which is on the order of $O(h) = C \cdot h$, for some constant C.

- **Conclusion:**

  If we divide h by 10, then we expect the error to get smaller by a factor of 10 as well, thus gaining one digit of accuracy. Consequently, the 2-point forward-difference formula is called a *first order* method.

2.2. The 3-point centered-difference

- Basic idea: The 2-point forward-difference can be derived by approximating the function *f* by a linear function that interpolates *f* at *x* and *x*+*h*. This suggests that we might improve the approximation by fitting a quadratic function to f, this time symmetrically at *x*-*h*, *x* and *x*+*h*.
- Let $x_0 = x-h$, $x_1 = x$, $x_2 = x+h$. Write the interpolating quadratic in Newton's form:

$$g(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

Thus $g'(x) = f[x_0, x_1] + f[x_0, x_1, x_2](x - x_0 + x - x_1)$

and $f'(x) \approx g'(x_1) = f[x_0, x_1] + f[x_0, x_1, x_2](x_1 - x_0)$ which gives

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

This is called the *3-point centered-difference formula.*

Analysis of approximation order

- **Analysis:**

  The second order Taylor series approximations give

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(c_1), \quad c_1 \in (x, x+h)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(c_2), \quad c_2 \in (x-h, x)$$

  from which we get

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f'''(c), \quad c \in (x-h, x+h).$$

  Therefore the 3-point centered-difference formula has the approximation error on the order of $O(h^2)$, and is called a *second order* method.

- **Conclusion:**

  If we divide h by 10, then we expect the error to get smaller by a factor of 100, thus gaining two digits of accuracy.

2.3. Rounding error

- Example

Approximate the derivative of $f(x) = e^x$ at $x = 0$ (the correct value is 1).

2-point formula: $f'(x) \approx f_2'(x) = \dfrac{e^{x+h} - e^x}{h}$

3-point formula: $f'(x) \approx f_3'(x) = \dfrac{e^{x+h} - e^{x-h}}{2h}$

h	2-point formula	error	3-point formula	error
10^{-1}	1.05170918075648	−0.05170918075648	1.00166750019844	−0.00166750019844
10^{-2}	1.00501670841679	−0.00501670841679	1.00001666674999	−0.00001666674999
10^{-3}	1.00050016670838	−0.00050016670838	1.00000016666668	−0.00000016666668
10^{-4}	1.00005000166714	−0.00005000166714	1.00000000166689	−0.00000000166689
10^{-5}	1.00000500000696	−0.00000500000696	1.00000000001210	−0.00000000001210
10^{-6}	1.00000049996218	−0.00000049996218	0.99999999997324	0.00000000002676
10^{-7}	1.00000004943368	−0.00000004943368	0.99999999947364	0.00000000052636
10^{-8}	0.99999999392253	0.00000000607747	0.99999999392253	0.00000000607747
10^{-9}	1.00000008274037	−0.00000008274037	1.00000002722922	−0.00000002722922

Observations

- Numerical behavior of 2-point forward-difference method
  - We roughly get k correct digits for $h = 10^{-k}$.
  - However, for $k > 8$, the approximation can deteriorate, due to the rounding error of floating-point arithmetic.

- Numerical behavior of 3-point centered-difference method
  - We roughly get 2k correct digits for $h = 10^{-k}$.
  - However, for $k > 5$, the approximation can deteriorate, due to the rounding error of floating-point arithmetic.

Rounding error analysis

Analyze the 3-point centered-difference formula.

- Denote the floating point version of $f(x+h), f(x-h)$ by $\hat{f}(x+h), \hat{f}(x-h)$, respectively. Then

$$\hat{f}(x+h) = f(x+h) + \epsilon_1, \qquad \hat{f}(x-h) = f(x-h) + \epsilon_2$$

  where $|\epsilon_1|, |\epsilon_2|$ are on the order of machine precision $\epsilon_{\text{machine}}$.

- The difference between the correct $f'(x)$ and the machine version of the 3-point formula is

$$\begin{aligned} f'(x)_{\text{correct}} - f'(x)_{\text{machine}} &= f'(x) - \frac{\hat{f}(x+h) - \hat{f}(x-h)}{2h} \\ &= f'(x) - \frac{f(x+h) + \epsilon_1 - f(x-h) - \epsilon_2}{2h} \\ &= \left(f'(x) - \frac{f(x+h) - f(x-h)}{2h} \right) + \frac{\epsilon_2 - \epsilon_1}{2h} \\ &= (f'(x)_{\text{correct}} - f'(x)_{\text{formula}}) + \text{error}_{\text{rounding}} \end{aligned}$$

Rounding error analysis (cont)

- Note that $\left|\dfrac{\epsilon_2 - \epsilon_1}{2h}\right| \leq \dfrac{2\epsilon_{\text{machine}}}{2h} = \dfrac{\epsilon_{\text{machine}}}{h}$.
- The absolute value of the error of the machine approximation of $f'(x)$ is bounded by

$$E(h) = \frac{h^2}{6} f'''(c) + \frac{\epsilon_{\text{machine}}}{h}$$

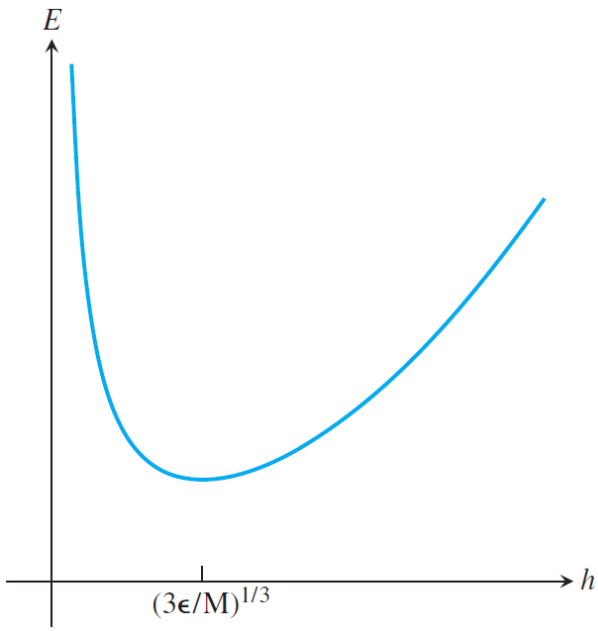
- The minimum of $E(h)$ occurs at

$$h = \sqrt[3]{\frac{3\epsilon_{\text{machine}}}{M}}$$

which is the solution of

$$0 = E'(h) = \frac{M}{3} h - \frac{\epsilon_{\text{machine}}}{h^2}$$

where we approximate $|f'''(c)| \approx |f'''(x)|$ by M.



- In double precision, h is approximately $\epsilon_{\text{machine}}^{1/3} \approx 10^{-5}$, consistent with the table.

2.4. Richardson extrapolation

- Suppose we have an order-n approximation of some quantity Q:

$$Q \approx F(h) + Ch^n$$

for some constant C.

- Halving h will improve the approximation roughly by a factor of $1/2^n$:

$$Q - F(h/2) \approx Ch^n/2^n \approx \frac{1}{2^n}(Q - F(h))$$

- We can re-arrange this to give

$$Q \approx \frac{2^n F(h/2) - F(h)}{2^n - 1}$$

which is called *Richardson extrapolation*.

5-point centered-difference

- 3-point centered-difference formula:

$$\underbrace{f'(x)}_{Q} = \underbrace{\frac{f(x+h)-f(x-h)}{2h}}_{F(h)} \underbrace{-\frac{f'''(c)}{6}}_{C} \underbrace{h^2}_{h^n}$$

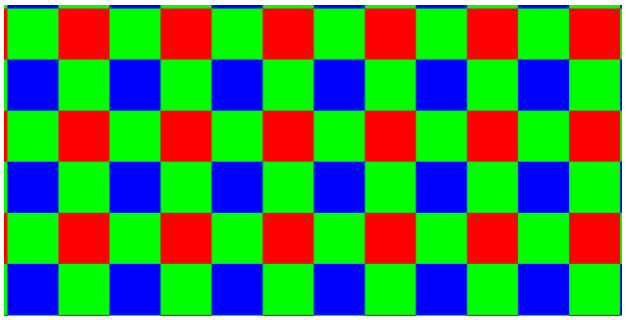
- Applying Richardson extrapolation leads to an improved approximation:

$$f'(x) \approx \frac{-f(x+h)+8f(x+h/2)-8f(x-h/2)+f(x-h)}{6h}$$

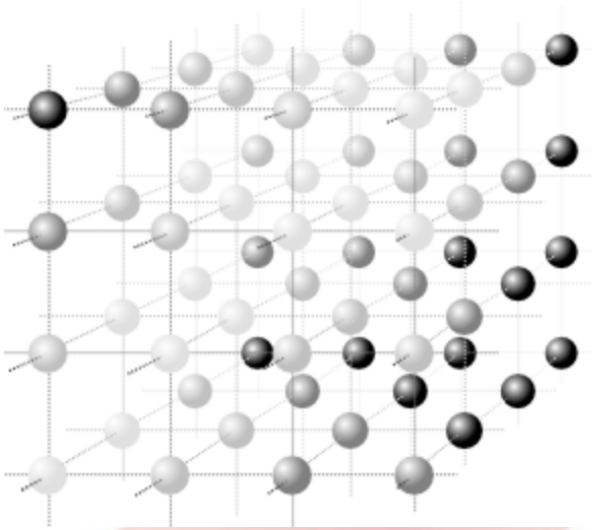
- This approximation is of 4th order. We get about $4k$ correct digits for $h = 10^{-k}$, but the safe h is roughly $\sqrt[5]{\epsilon_{\text{machine}}}$. That is, $k = 3$ in double precision.

2.5. Questions for you to think about

- How can we extend these difference formulae to a 2D function f(x,y)?
  - direct application: image processing where the function is given by intensity at each pixel

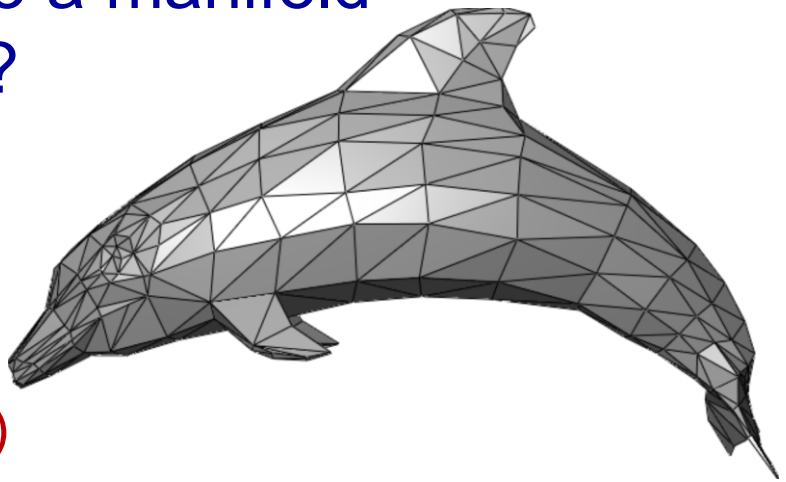


- How can we extend these difference formulae to a 3D function f(x,y,z)?
  - direct application: volume data such as MRI where the function is given by intensity at each voxel



Questions for you to think about

- **More challenging question:** How can we extend these difference formulae to a manifold surface such as a triangular mesh?



  - Parameterization:

    introduce parameters (u,v)

    construct vector-valued function P(u,v)

    Ref: https://www.inf.usi.ch/hormann/parameterization/CourseNotes.pdf

  - Direct definition based on discrete geometry

    Ref: Mark Meyer, Mathieu Desbrun, Peter Schröder and Alan H. Barr, "*Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*", VisMath 2002. Online: http://multires.caltech.edu/pubs/diffGeoOps.pdf

Outline

Integration

Task: Suppose we have access to the function values f(x) of some function f, but would like to know the definite integral $\int_a^b f(x)dx$ of f.

Main idea: As for differentiation, we fit a local polynomial to *f* and integrate it instead.

- Newton-Cotes Rules
- Romberg Integration
- Gaussian Quadrature

3.1. Closed Newton-Cotes rules

- Simple situation

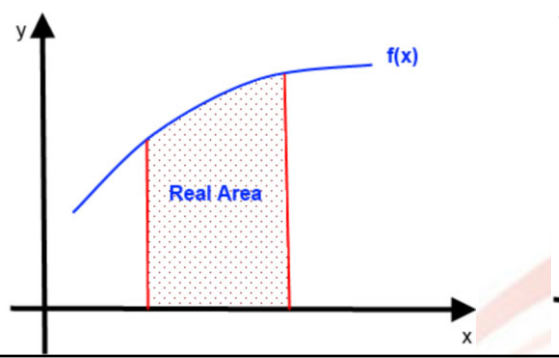
We use the linear interpolant to $f(x)$ at $x = a$ and $x = b$:

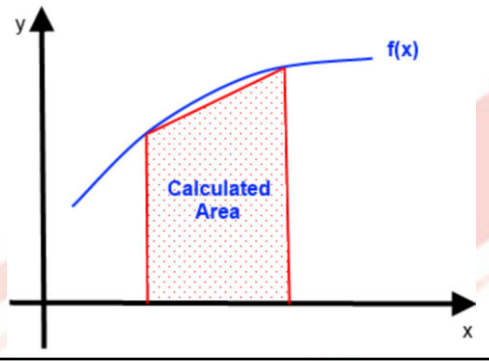
$$p_1(x) = \frac{b-x}{b-a} f(a) + \frac{x-a}{b-a} f(b).$$

Then

$$\int_a^b f(x)dx \approx \int_a^b p_1(x)dx = (b-a)\frac{f(a)+f(b)}{2}$$

which is known as the *Trapezoid Rule*.
Actually it computes the area of the trapezoid with corners $(a, 0), (a, f(a)), (b, f(b)), (b, 0)$.





Closed Newton-Cotes rules

- General situation

We consider the degree n polynomial that interpolates $f(x)$ at $x_i = a + ih$ with $h = (b-a)/n$ for $i = 0, \cdots, n$:

$$p_n(x) = \sum_{i=0}^{n} f(x_i) L_i^n(x)$$

where $L_i^n(x) = \prod_{k=0, k\neq i}^{n} \frac{x - x_k}{x_i - x_k}$ are the Lagrange polynomials with respect to $x_0, \cdots, x_n$. This leads to the **Newton-Cotes Rule of degree n**:

$$\int_a^b f(x)dx \approx \int_a^b p_n(x)dx = h \sum_{i=0}^{n} f(x_i)\alpha_i^n$$

with the weights

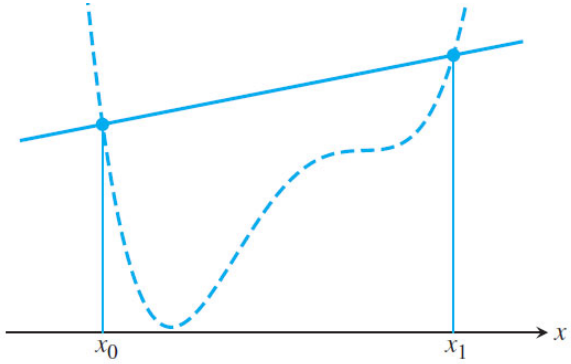
$$\alpha_i^n = \frac{1}{h}\int_a^b L_i^n(x)dx = \int_0^n L_i^n(a+ht)dt = \int_0^n \prod_{k=0, k\neq i}^{n} \frac{t-k}{i-k} dt$$

are independent of f, a and b. They just depend on n and can be precomputed; see the Table in next slide.

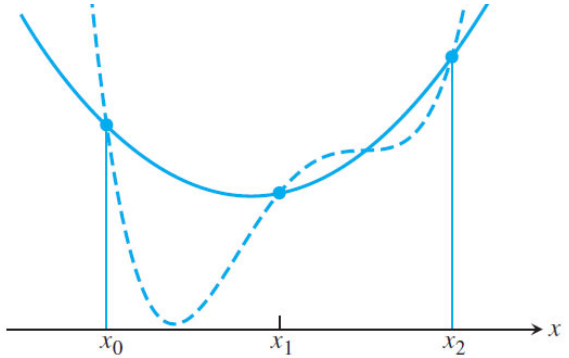
Closed Newton-Cotes rules

n	α_i^n	Name
1	$\frac{1}{2}, \frac{1}{2}$	Trapezoidal Rule
2	$\frac{1}{3}, \frac{4}{3}, \frac{1}{3}$	Simpson's Rule
3	$\frac{3}{8}, \frac{9}{8}, \frac{9}{8}, \frac{3}{8}$	3/8 Rule
4	$\frac{15}{45}, \frac{64}{45}, \frac{24}{45}, \frac{64}{45}, \frac{15}{45}$	Boole's Rule

Table 1: Weights for the Newton–Cotes Rules of Degree n.



Trapezoid rule



Simpson's rule

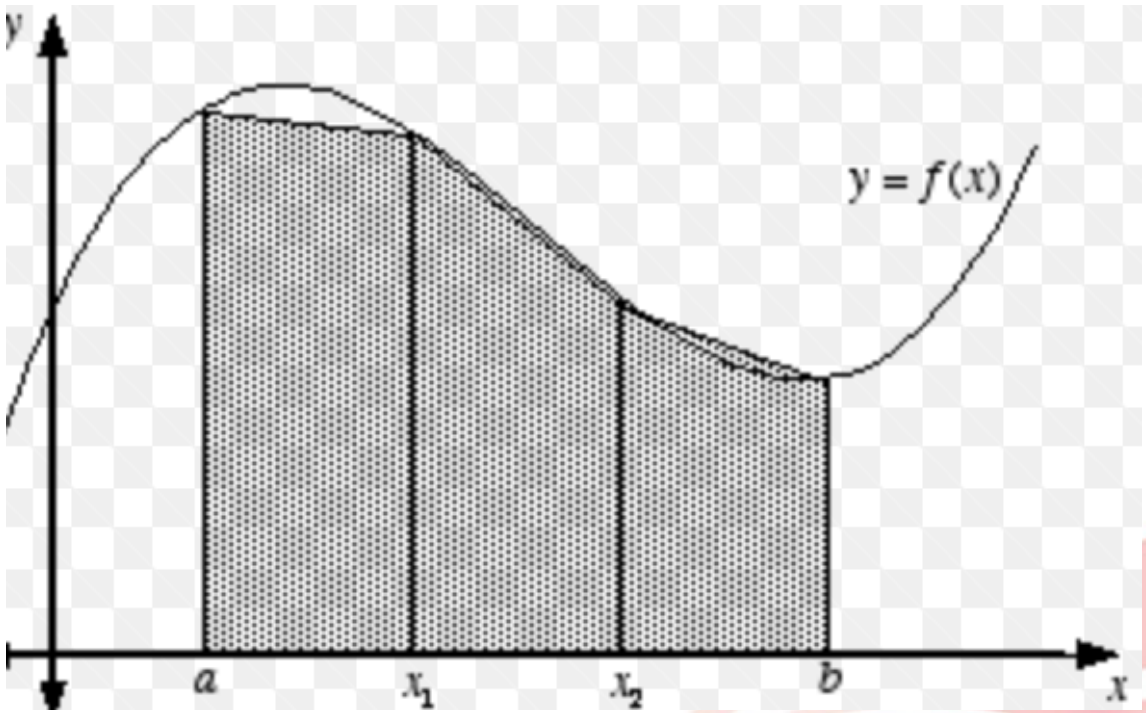
In particular,

- Trapezoidal Rule: $\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$
- Simpson's Rule: $\int_a^b f(x)dx \approx (b-a)\frac{f(a)+4f\left(\frac{a+b}{2}\right)+f(b)}{6}$

Closed Newton-Cotes rules

- The error of the Newton-Cotes Rule of degree *n* is on the order of $O(h^{p+1})$, where $p = n+1$ for odd n and $p = n+2$ for even n.
- Composite Newton-Cotes rules

In practice, we split the integration interval into m pieces of equal length. Let $x_0 = a, x_i = a + ih, x_m = b$ with $h = (b-a)/m$. Then $\int_a^b f(x)dx = \sum_{i=0}^{m-1} \int_{x_i}^{x_{i+1}} f(x)dx$.



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

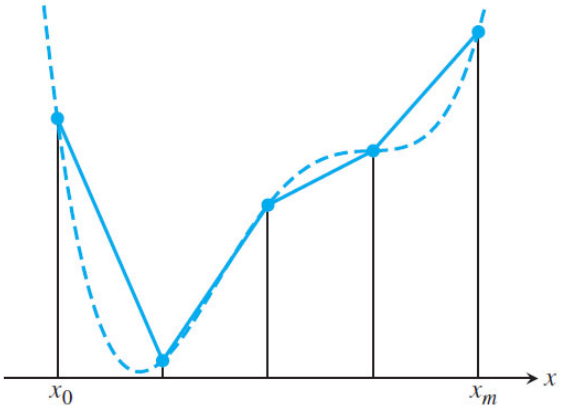
Closed Newton-Cotes rules

- Compostie Trapezoid rule: Consider an evenly spaced grid $a = x_0 < x_1 < \cdots < x_m = b$, with $x_i = a + ih$ and $h = \frac{b-a}{m}$.

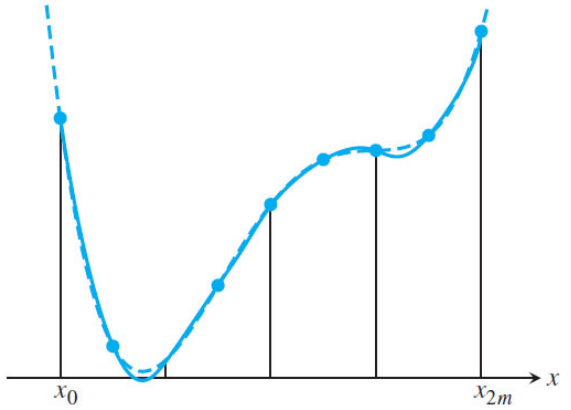
$$\int_a^b f(x) \approx \frac{h}{2}\left(f(a) + f(b) + 2\sum_{i=1}^{m-1} f(x_i)\right)$$

- Composite Simpson's rule: Consider an evenly spaced grid $a = x_0 < x_1 < \cdots < x_{2m} = b$, with $x_i = a + ih$ and $h = \frac{b-a}{2m}$.

$$\int_a^b f(x) \approx \frac{h}{3}\left(f(a) + f(b) + +4\sum_{i=1}^{m} f(x_{2i-1}) + 2\sum_{i=1}^{m-1} f(x_{2i})\right)$$



Composite Trapezoid



Composite Simpson's

3.2. Open Newton-Cotes rules

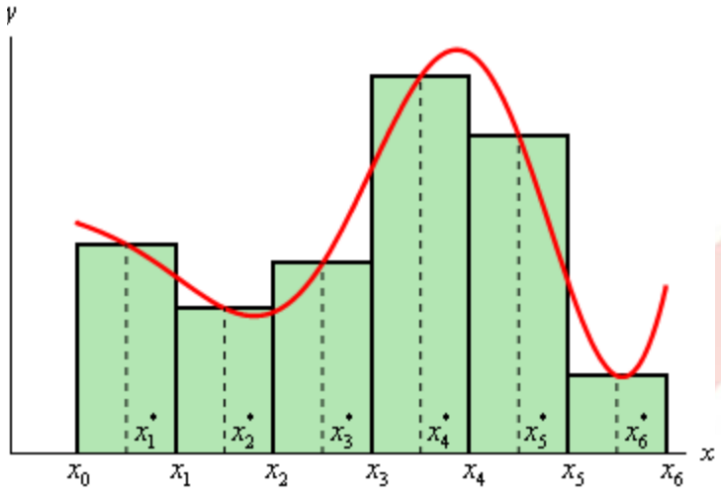
- **Midpoint Rule:**

  If the function f is not well defined at the boundary of the integration interval $[a, b]$, then we can consider an approximation on fitting local polynomial to data inside the interval. The simplest case is to fit a constant to f at the midpoint of the interval $x = (a + b)/2$ to get the *Midpoint Rule*:

$$\int_a^b f(x)dx \approx (b-a) f\left(\frac{a+b}{2}\right)$$

- **Composite Midpoint Rule:** Similarly, consider an evenly spaced grid $a = x_0 < x_1 < \cdots < x_m = b$, with $x_i = a + ih$ and $h = \frac{b-a}{m}$.

$$\int_a^b f(x)dx \approx h \sum_{i=0}^{m-1} f\left(\frac{x_i + x_{i+1}}{2}\right)$$



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

3.3. Romberg integration

- **Romberg integration** aims to improve the approximation results using the idea of **Richardson extrapolation**.

We take the Composite Trapezoid Rule as example.

– We first compute iteratively the approximation $R_{j,1}$ using the Composite Trapezoid Rule for $m = 2^{j-1}$ and $h_j = (b-a)/2^{j-1}$:

$$h_1 = b - a \quad \longrightarrow \quad R_{1,1} = \frac{h_1}{2}\big(f(a) + f(b)\big),$$

$$h_2 = \frac{b-a}{2} \quad \longrightarrow \quad R_{2,1} = \frac{h_2}{2}\big(f(a) + 2f(\tfrac{a+b}{2}) + f(b)\big),$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$h_j = \frac{b-a}{2^{j-1}} \quad \longrightarrow \quad R_{j,1} = \frac{h_j}{2}\big(f(a) + 2f(a+h_j) + 2f(a+2h_j) + \cdots + 2f(a+(2^{j-1}-1)h_j) + f(b)\big).$$

Actually, the computation of $R_{j,1}$ can be done recursively:

$$R_{j,1} = \frac{1}{2}R_{j-1,1} + h_j \sum_{i=1}^{2^{j-2}} f(a + (2i-1)h_j).$$

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Romberg integration (cont)

– Second, we apply Richardson extrapolation. Note that the Composite Trapezoid Rule has an approximation error of order $O(h^2)$. That is,

$$\underbrace{\int_a^b f(x)\,dx}_{Q} = \underbrace{R_{j-1,1}}_{F(h)} + \underbrace{O(h_{j-1}^2)}_{C\cdot h^n}, \qquad \text{with } h = h_{j-1} \text{ and } n = 2$$

We can get the improved approximation rule:

$$\int_a^b f(x)\,dx \approx \frac{2^n F(h/2) - F(h)}{2^n - 1} = \frac{4R_{j,1} - R_{j-1,1}}{3} = R_{j,2}$$

which turns out to have an approximation order of order $O(h^4)$. In fact, $R_{j,2}$ is just the Composite Simpson's Rule for $h = h_j$.

Romberg integration (cont)

– Third, we generalize the process to extrapolate the approximation $R_{j,k}$ of the integral from the values $R_{j,k-1}$ and $R_{j-1,k-1}$ using the formula

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

Then the *Romberg Integration* scheme can compute the values $R_{j,k}$ in the following way:

$$\begin{array}{ccccccccc}
R_{1,1} & & & & & & & & \\
 & \ddots & & & & & & & \\
R_{2,1} & \dots & R_{2,2} & & & & & & \\
 & \ddots & & \ddots & & & & & \\
R_{3,1} & \dots & R_{3,2} & \dots & R_{3,3} & & & & \\
 & \ddots & & \ddots & & \ddots & & & \\
R_{4,1} & \dots & R_{4,2} & \dots & R_{4,3} & \dots & R_{4,4} & & \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots &
\end{array}$$

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Romberg integration (cont)

The evaluation goes row by row, and in each row it starts to compute the $R_{j,1}$ and then from left to right to get the next $R_{j,k}$, until the difference $|R_{j,j} - R_{j,j-1}|$ is smaller than the threshold. Note that the values $R_{j,k}$ in the k-th column of the diagram have an error of order $O(h_j^{2k})$, so that $R_{j,j}$ converges very quickly to the correct value of the integral as j increases.

- The algorithm basically involves two computational formulae. The outline of the algorithm is given in the figure.

Romberg Integration

$$R_{11} = (b-a)\frac{f(a)+f(b)}{2}$$

for $j = 2, 3, \ldots$

$$h_j = \frac{b-a}{2^{j-1}}$$

$$R_{j1} = \frac{1}{2}R_{j-1,1} + h_j \sum_{i=1}^{2^{j-2}} f(a + (2i-1)h_j)$$

for $k = 2, \ldots, j$

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

end

end

3.4. Gaussian quadrature

- Note that $\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{u(b-a)+(a+b)}{2}\right)\frac{b-a}{2}du.$

  So here we focus on computing the integral $\int_{-1}^{1} f(x)dx$.

- Introduce the*Legendre polynomial*:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n.$$

  - $P_n(x)$ is a degree n polynomial.
  - The set of $P_i(x)$ for $0 \le i \le n$ is orthogonal on $[-1,1]$.
  - $P_n(x)$ has n roots $x_1, \cdots, x_n$ in $[-1,1]$.

Gaussian quadrature (cont)

- A higher degree of precision (with respect to the number of function evaluations) can be achieved by giving up equidistant spacing. In fact, *Gaussian quadrature* uses the roots of the n-th Legendre polynomial as the evaluation nodes and approximating the integral as

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

where the weights $c_i = \int_{-1}^{1} L_i(x)dx$ with $L_i(x)$ denoting the i-th Lagrange polynomial with respect to the nodes $x_1, \cdots, x_n$. x_i and c_i can be precomputed.

n	roots x_i	coefficients c_i
2	$-\sqrt{1/3} = -0.57735026918963$	$1 = 1.00000000000000$
	$\sqrt{1/3} = 0.57735026918963$	$1 = 1.00000000000000$
3	$-\sqrt{3/5} = -0.77459666924148$	$5/9 = 0.55555555555555$
	$0 = 0.00000000000000$	$8/9 = 0.88888888888888$
	$\sqrt{3/5} = 0.77459666924148$	$5/9 = 0.55555555555555$

- It can be proved (see [Chapter 5.5]) that the Gaussian Quadrature has degree of precision $2n - 1$. So if one has the freedom to evaluate f at the non-equidistant nodes $x_1, \cdots, x_n$, then this is a preferred method for approximating the integral of f.

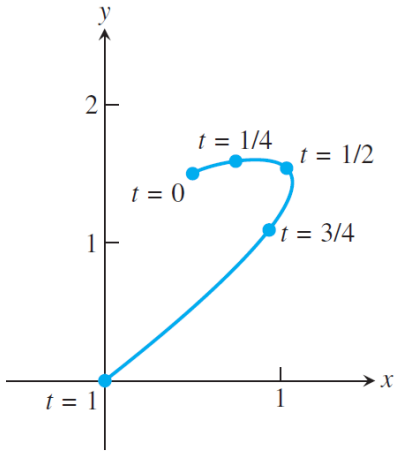
Outline

Motion control

- Suppose we are given a parametric path $P(t), t \in [0,1]$ and would like to move some device (e.g., a robot, milling machine, or plotter) along the path. It may happen that the given parameterization of the path leads to a non-uniform motion of the device, with drastic variation in speed. However, some applications such as numerical machining prefer to maintain constant speed along the path. That is, during each second, progress should be made along an equal length of the machine-material interface.

- For example, a path is given by the cubic Bezier curve

$$P(t) = \sum_{i=0}^{3} C_i B_i^3(t), \quad t \in [0,1]$$



with $C_0 = \begin{pmatrix} 0.5 \\ 1.5 \end{pmatrix}, C_1 = \begin{pmatrix} 0.6 \\ 1.6 \end{pmatrix}, C_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, C_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

- This model has a slow speed $\|P'(0)\| = 0.42$ at the start, compared to $\|P'(1)\| = 8.49$ at the end. If we want to traverse the path with near-constant speed, numerical integration comes to our rescue.

Motion control (cont)

- The length of the path from point $P(t_1)$ to point $P(t_2)$ is given by the integral

$$L(t_1, t_2) = \int_{t_1}^{t_2} \|P'(t)\| dt$$

which is usually not solvable analytically. Numerical integraion is needed.

- To find the midpoint $P(T)$ of the path P, one may apply Newton's method to find the root of the function

$$f(T) = L(0, T) - L(T, 1) = \int_0^T \|P'(t)\| dt - \int_T^1 \|P'(t)\| dt.$$

Note that $f'(T) = 2\|P'(t)\|$. Using the initial guess $T_0 = 0.5$, we can find $T \approx 0.8$ with just 6 iterations.

Motion control (cont)

- To further find the parameters s_1 and s_3 that give the points at $1/4$ and $3/4$ along P, apply Newton's method to the functions

$$f(s_1) = L(0, s_1) - L(s_1, T), \quad f(s_3) = L(T, s_3) - L(s_3, 1).$$

  Overall, we find that for the parameter values

$$s_0 = 0.0, \quad s_1 = 0.55, \quad s_2 = 0.80, \quad s_3 = 0.92, \quad s_4 = 1.0,$$

  the corresponding points $P(s_i)$ are equdistant along the path P, i.e., $L(s_i, s_{i+1}) \approx \frac{1}{4} L(0, 1) = 0.623812$ for $i = 0, 1, 2, 3$.

- Similarly, one can find any number of equidistanct points along path P and then traverse the path approximately with constant speed by moving linearly from one of these equidistant points to the next.

  If a smoother C^1 or C^2 traversal is needed, then these equidistant points can be interpolated with a cubic Bezier spline or a cubic B-spline.

Outline

Homework

Q6.1: Work out a third order method for approximating the first derivative of a function, based on a 4-point one-sided difference formula. That is, your approximation of $f'(x)$ should depend on $f(x), f(x+h), f(x+2h)$ and $f(x+3h)$.

Describe how you derive our formula and show that it really leads to a third order method.

Q6.2: Let f be a cubic polynomial and g be a quadratic polynomial that matches f at $a, a+h$ and $a+2h$ for some a and $h > 0$. That is,

$$g(a+ih) = f(a+ih), \quad i = 0, 1, 2.$$

Show that $\int_a^{a+2h} f(x)dx = \int_a^{a+2h} g(x)dx.$

Outline

Summary

- Main idea: fit a local polynomial, and then differentiate / integrate it instead.
- Differentiation
  - 2-point forward-difference
  - 3-point centered-difference
  - Richardson extrapolation
- Integration
  - Closed Newton-Cotes Rules (Trapezoid, Simpson's)
  - Open Newton-Cotes Rules
  - Romberg Integration
  - Gaussian Quadrature

End