MATH 4513 Numerical Analysis

Chapter 4. Numerical Differentiation and Integration

Xu Zhang

Department of Mathematics
Oklahoma State University

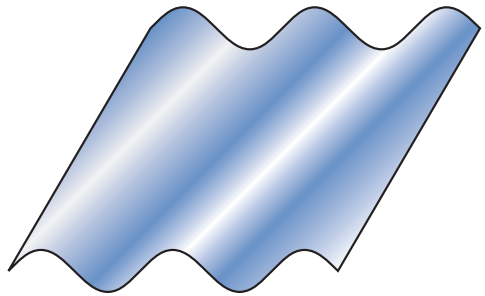
Text Book: Numerical Analysis (10th edition)
R. L. Burden, D. J. Faires, A. M. Burden

Table of Contents

Motivation

- Numerical integration and differentiation are generally key steps in many numerical computations.
- For example, finding the length of a sheet of corrugated roofing which is approximated by a sine wave

$$f(x) = \sin(x), \quad x \in [0, 48].$$



- From Calculus, we know the length of the curve is

$$L = \int_0^{48} \sqrt{1 + [f'(x)]^2}dx = \int_0^{48} \sqrt{1 + \cos^2(x)}dx = ?$$

- The analytical integral is not available. We need to approximate this integral numerically.

4.1 Numerical Differentiation

- **Goal**: Develop numerical approximation of derivatives $f'(x)$, $f''(x)$, etc, using only the values of the function $f(x)$.
- Recall that the derivative of $f(x)$ at x_0 is defined as

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}.$$

- Intuitively, we can choose small h and use

$$\frac{f(x_0 + h) - f(x_0)}{h}$$

  to approximation $f'(x_0)$.
- **Question:** How "small" is small?

- We use the formula above to approximate the derivative of $f(x) = \sin(x)$ at $x = 1$. Note that the true value is $f'(1) = \cos(1)$.
- We test with $h = \frac{1}{10}, \frac{1}{100}, \cdots, \frac{1}{10^{15}}$, and find the relative error.

A MATLAB Script

```
% Test the derivative approximation
% f(x) = sin(x), f'(x) = cos(x).
% This example we will approximation f'(1) = cos(1)
clear
format long
h = zeros(15,1);
err = zeros(15,1);
for i = 1:15
    h(i) = 10^(-i);
    app = (sin(1+h(i))-sin(1))/h(i);
    err(i) = abs(app - cos(1))/cos(1);
end
disp('   h                    relative error')
disp([h,err])
```

Numerical Result

```
>> ex4_1_0
   h                    relative error
   0.100000000000000    0.079471349402736
   0.010000000000000    0.007803640314835
   0.001000000000000    0.000778870464261
   0.000100000000000    0.000077872052482
   0.000010000000000    0.000007787052229
   0.000001000000000    0.000000778724808
   0.000000100000000    0.000000077415348
   0.000000010000000    0.000000005496710
   0.000000001000000    0.000000097244201
   0.000000000100000    0.000000108237621
   0.000000000010000    0.000002163055846
   0.000000000001000    0.000080029673119
   0.000000000000100    0.001358343083757
   0.000000000000010    0.006860929812673
   0.000000000000001    0.027409112053748
```

- At the beginning, the relative error gets smaller as h decreases.
- After the eighth iteration, $h = 10^{-8}$, the error starts to grows as we further reduce the size of h. **Why?**
- We will answer this question at the end of this section.

- **Idea:** Approximate $f(x)$ by an interpolating polynomial $P(x)$, then use $P'(x)$ to approximate the derivative $f'(x)$.
- Consider the linear Lagrange polynomials at x_0 and $x_1 = x_0 + h$.

$$L_{1,0}(x) = \frac{x - x_1}{x_0 - x_1}, \qquad L_{1,1}(x) = \frac{x - x_0}{x_1 - x_0}.$$

- Then the linear Lagrange interpolation $P_1(x)$ of $f(x)$ is

$$\begin{aligned} P_1(x) &= f(x_0)L_{1,0}(x) + f(x_1)L_{1,1}(x) \\ &= f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0} \\ &= f(x_0)\frac{x - x_1}{-h} + f(x_0 + h)\frac{x - x_0}{h}. \end{aligned}$$

- Taking the derivative, we get

$$P_1'(x) = \frac{f(x_0 + h) - f(x_0)}{h} \approx f'(x_0).$$

- This formula is the same as the definition of derivative of $f'(x_0)$.

- Recall the error bound of linear Lagrange interpolation (ref. Sec 3.1) is

$$f(x) - P_1(x) = \frac{f''(\xi(x))}{2!}(x - x_0)(x - x_1).$$

- So, the error bound of the derivative is

$$\begin{aligned} f'(x) - P_1'(x) &= D_x \left(\frac{f''(\xi(x))}{2!}(x - x_0)(x - x_1) \right). \\ &= \frac{1}{2}(2x - x_0 - x_1) f''(\xi) + \frac{(x - x_0)(x - x_1)}{2} D_x \left(f''(\xi) \right) \end{aligned}$$

  where ξ is between x_0 and x.
- When $x = x_0$, the second term on the right hand side is zero. We have

$$f'(x_0) - P_1'(x_0) = -\frac{h}{2} f''(\xi).$$

Forward/Backward Difference Formula

$$f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}. \tag{4.1}$$

- If $h > 0$, it is called **forward-difference formula**.
- If $h < 0$, it is called **backward-difference formula**.

Error Bound for Forward/Backward Difference Formula

For small value of h, we have

$$\left| f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h} \right| \leq \frac{M|h|}{2}, \tag{4.2}$$

where M is the bound for $|f''(x)|$ for x between x_0 and x_0+h. We say that the **truncation error** is $\mathcal{O}(h)$.

- In general, let $P_n(x)$ be the Lagrange interpolation of $f(x)$ at $n+1$ distinct points $x_0, x_1, \cdots, x_n$. That is

$$P_n(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}(x)$$

where

$$L_{n,k}(x) = \prod_{\substack{j \neq k \\ j=0}}^{n} \frac{x - x_j}{x_k - x_j}.$$

- The derivative of $f(x)$ can be approximated by

$$\boxed{f'(x) \approx P_n'(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}'(x).} \tag{4.3}$$

- The error when $x = x_j$ is

$$f'(x_j) - P_n'(x_j) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k).$$

- (4.3) is called an $(n+1)$-**point formula** to approximate $f'(x_j)$.

- To approximate $f'(x)$, 3-point and 5-point formulas are commonly used.
- For **three-point formulas**, we have for each $j = 0, 1, 2$,

$$\begin{aligned} f'(x_j) &= P_2'(x_j) + \frac{f'''(\xi)}{3!} \prod_{\substack{j=0 \\ j \neq k}}^{2} (x_j - x_k). \\ &= f(x_0) \frac{2x_j - x_1 - x_2}{(x_0 - x_1)(x_0 - x_2)} + f(x_1) \frac{2x_j - x_0 - x_2}{(x_1 - x_0)(x_1 - x_2)} \\ &\quad + f(x_2) \frac{2x_j - x_0 - x_1}{(x_2 - x_0)(x_2 - x_1)} + \frac{f'''(\xi)}{3!} \prod_{\substack{j=0 \\ j \neq k}}^{2} (x_j - x_k). \end{aligned}$$

- If the nodes are equally spaced, i.e., $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, then

$$\begin{aligned} f'(x_0) &= -\frac{3}{2h} f(x_0) + \frac{1}{h} f(x_1) - \frac{1}{2h} f(x_2) + \frac{h^2}{3} f'''(\xi_0), \\ f'(x_1) &= -\frac{1}{2h} f(x_0) + \frac{1}{2h} f(x_2) - \frac{h^2}{6} f'''(\xi_1), \\ f'(x_2) &= \frac{1}{2h} f(x_0) - \frac{2}{h} f(x_1) - \frac{3}{2h} f(x_2) + \frac{h^2}{3} f'''(\xi_2). \end{aligned}$$

Three-Point Endpoint Formula for f'

$$f'(x_0) = \frac{1}{2h}\left[-3f(x_0) + 4(x_0 + h) - f(x_0 + 2h)\right] + \frac{h^2}{3}f'''(\xi) \tag{4.4}$$

where ξ lies between x_0 and $x_0 + 2h$.

Remark The other end-point formula can be obtained by replacing h with $-h$.

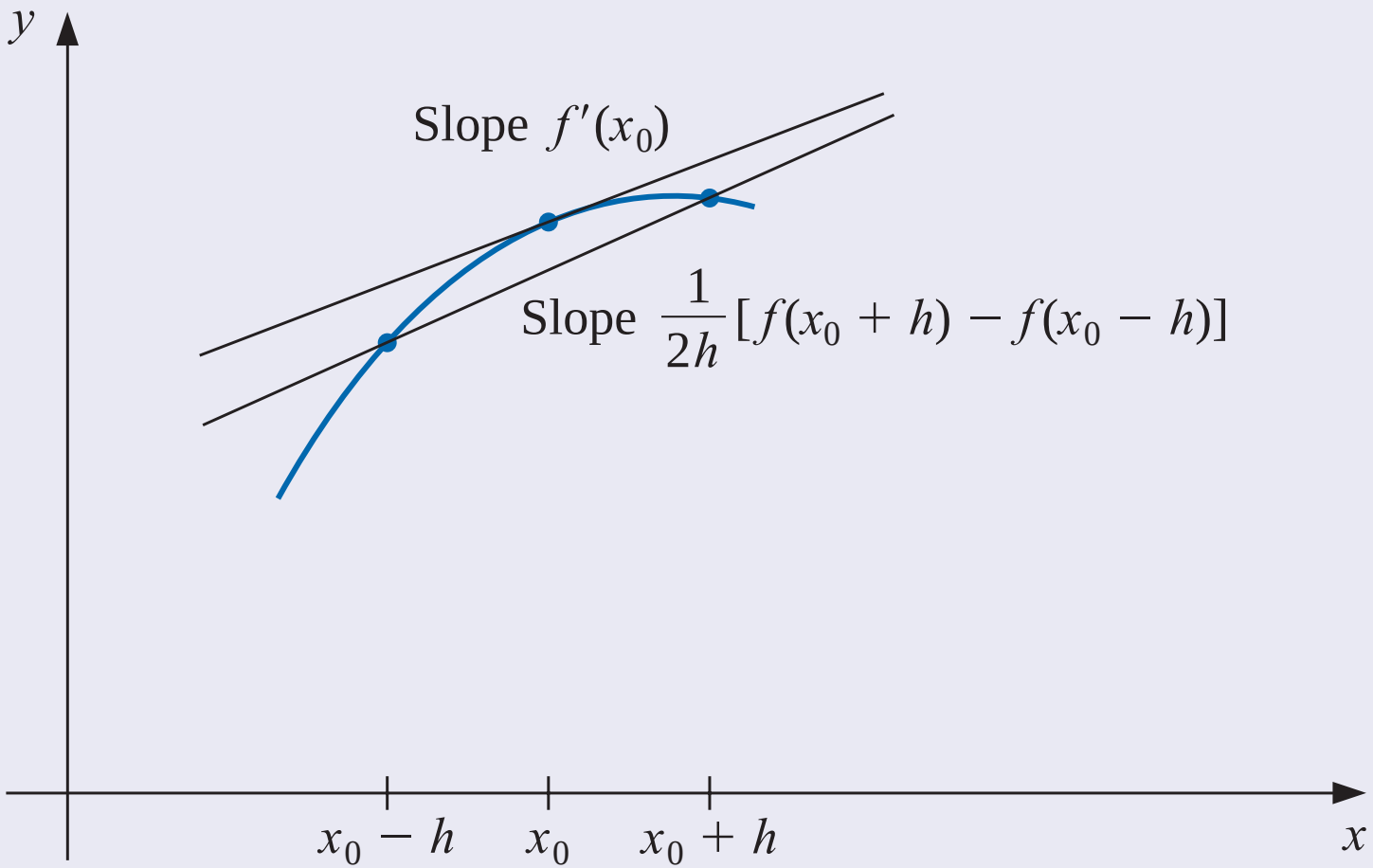
Three-Point Midpoint Formula for f'

$$f'(x_0) = \frac{1}{2h}\left[f(x_0 + h) - f(x_0 - h)\right] - \frac{h^2}{6}f'''(\xi) \tag{4.5}$$

where ξ lies between $x_0 - h$ and $x_0 + h$.

- **Remark** Although the truncation error for midpoint and endpoint formulas are both $\mathcal{O}(h^2)$, the error in (4.5) is approximately half the error in (4.4). This is because (4.5) uses data on both sides of x_0 while (4.4) only uses data from one side.
- **Exercise** Derive the three-point formulas for f' if the points are not equally-spaced. That is $x_1 = x_0 + h_1$, and $x_2 = x_1 + h_2$.

An illustration of the three-point midpoint approximation



Five-Point Midpoint Formula for f'

$$f'(x_0) = \frac{1}{12h}\Big[f(x_0-2h)-8(x_0-h)+8f(x_0+h)-f(x_0+2h)\Big]+\frac{h^4}{30}f^{(5)}(\xi). \quad (4.6)$$

where ξ lies between $x_0 - 2h$ and $x_0 + 2h$.

Five-Point Endpoint Formula for f'

$$\begin{aligned} f'(x_0) = \frac{1}{12h}\Big[&-25f(x_0)+48f(x_0+h)-36(x_0+2h) \\ &+16f(x_0+3h)-3f(x_0+4h)\Big]+\frac{h^4}{5}f^{(5)}(\xi) \end{aligned} \quad (4.7)$$

where ξ lies between x_0 and $x_0 + 4h$.

Remark

- The truncation errors of 5-point formulas (4.6) and (4.7) are $\mathcal{O}(h^4)$.
- The formula (4.7) is particularly useful for the clamped spline interpolation for approximating the derivative of the boundary points.

Example 1

Given the following data for $f(x) = xe^x$.

x	1.8	1.9	2.0	2.1	2.2
$f(x)$	10.889365	12.703199	14.778112	17.148957	19.855030

Use three-point and five-point formulas to approximate $f'(2.0)$.

Solution (1/3)

- We first use 3-point endpoint formula (4.4) with $h = 0.1$,

$$\begin{aligned} f'(2.0) &\approx \frac{1}{0.2}[-3f(2.0) + 4(2.1) - f(2.2)] \\ &= 5[-3(14.778112) + 4(17.148957) - 19.855030] \\ &= 22.032310. \end{aligned}$$

Solution (2/3)

- We can also use 3-point endpoint formula (4.4) with $h = -0.1$,

$$\begin{aligned} f'(2.0) &\approx \frac{1}{-0.2}[-3f(2.0) + 4(1.9) - f(1.8)] \\ &= -5[-3(14.778112) + 4(12.703199) - 10.889365] \\ &= 22.054525. \end{aligned}$$

- Using 3-point midpoint formula (4.5) with $h = 0.1$ we have

$$f'(2.0) \approx \frac{1}{0.2}[f(2.1) - f(1.9)] = 5(17.148957 - 12.7703199) = 22.228790.$$

- Using 3-point midpoint formula (4.5) with $h = 0.2$ we have

$$f'(2.0) \approx \frac{1}{0.4}[f(2.2) - f(1.8)] = 2.5(19.855030 - 10.889365) = 22.41416.$$

Solution (3/3)

- The only 5-point formula applicable is the 5-point midpoint formula (4.6) with $h = 0.1$

$$\begin{aligned} f'(2.0) &\approx \frac{1}{1.2}\left[f(1.8) - 8f(1.9) + 8f(2.1) - f(2.2)\right] \\ &= \frac{1}{1.2}[10.889365 - 8(12.703199) + 8(17.148957) - 19.855030] \\ &= 22.166999. \end{aligned}$$

- The true derivative is $f'(2.0) = (2+1)e^2 = 22.167168$.
- The relative error of these approximations are
  - 3-pt endpoint with $h = 0.1$: 0.00608
  - 3-pt endpoint with $h = -0.1$: 0.00509
  - 3-pt midpoint with $h = 0.1$: 0.00277
  - 3-pt midpoint with $h = 0.2$: 0.01114
  - 5-pt midpoint with $h = 0.1$: 0.0000076

Approximation of Higher Derivatives

- Methods can also be derived to find approximations to higher derivatives of a function using only tabulated values of the function at various points.
- We will use Taylor expansion to derive the 2nd-order derivative approximation. Note that

$$\begin{aligned} f(x_0+h) &= f(x_0)+hf'(x_0)+\frac{h^2}{2}f''(x_0)+\frac{h^3}{6}f'''(x_0)+\frac{h^4}{24}f^{(4)}(\xi_1), \\ f(x_0-h) &= f(x_0)-hf'(x_0)+\frac{h^2}{2}f''(x_0)-\frac{h^3}{6}f'''(x_0)+\frac{h^4}{24}f^{(4)}(\xi_{-1}), \end{aligned}$$

where $\xi_1 \in (x_0, x_0+h)$ and $\xi_{-1} \in (x_0-h, x_0)$.

- Adding these two equations we obtain

$$f(x_0+h)+f(x_0-h) = 2f(x_0)+h^2f''(x_0)+\frac{h^4}{24}\Big(f^{(4)}(\xi_1)+f^{(4)}(\xi_{-1})\Big).$$

- Subtracting $2f(x_0)$ on both side and dividing by h^2, we obtain

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - \frac{h^2}{24}\Big(f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\Big).$$

- Using the Intermediate Value Theorem, if $f^{(4)}$ is continuous, then there exists $\xi \in (x_0-h, x_0+h)$ such that

$$f^{(4)}(\xi) = \frac{1}{2}\big(f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\big).$$

Midpoint Formula for f''

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2} - \frac{h^2}{12}f^{(4)}(\xi). \tag{4.8}$$

for some $\xi \in (x_0-h, x_0+h)$.

Remark: The truncation error of the midpoint formula (4.8) is $\mathcal{O}(h^2)$.

Example 2

Given the following data for $f(x) = xe^x$.

x	1.8	1.9	2.0	2.1	2.2
$f(x)$	10.889365	12.703199	14.778112	17.148957	19.855030

Use the second derivative midpoint formula to approximate $f''(2.0)$.

Solution

- Using the midpoint formula (4.8) with $h = 0.1$ we have

$$f''(2.0) \approx \frac{1}{0.1^2}\Big[f(1.9) - 2f(2.0) + f(2.1)\Big] = 29.593200.$$

- Using the midpoint formula with $h = 0.2$ we have

$$f''(2.0) \approx \frac{1}{0.2^2}\Big[f(1.8) - 2f(2.0) + f(2.2)\Big] = 29.704275.$$

- Because $f''(x) = (x+2)e^x$, the exact value is $f''(2.0) = 29.556224$.
- The relative errors for $h = 0.1$ and 0.2 are 0.00125 and 0.00501, respectively.

Round-Off Error Stability

Example 3

Use three-point midpoint formula to approximate $f'(0.9)$ where $f(x) = \sin(x)$ with $h = 10^{-i}$, $i = 1, 2, \cdots, 12$.

Solution. (The screenshot of MATLAB command window)

Approximation of derivative of sin(x) at 0.9 using three-point midpoint

i	h	Approx	cos(0.9)	error
1	0.100000000000	0.620574469542	0.621609968271	0.001035498729
2	0.010000000000	0.621599608156	0.621609968271	0.000010360114
3	0.001000000000	0.621609864669	0.621609968271	0.000000103602
4	0.000100000000	0.621609967235	0.621609968271	0.000000001036
5	0.000010000000	0.621609968254	0.621609968271	0.000000000016
6	0.000001000000	0.621609968277	0.621609968271	-0.000000000006
7	0.000000100000	0.621609967943	0.621609968271	0.000000000327
8	0.000000010000	0.621609969054	0.621609968271	-0.000000000783
9	0.000000001000	0.621609985707	0.621609968271	-0.000000017436
10	0.000000000100	0.621609985707	0.621609968271	-0.000000017436
11	0.000000000010	0.621608320373	0.621609968271	0.000001647898
12	0.000000000001	0.621613871488	0.621609968271	-0.000003903217

The smallest error is around $i = 5, 6$, i.e., h is between 10^{-6} and 10^{-5}. **Why?**

- Let us examine the 3-pt midpoint formula:

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - \frac{h^2}{6} f'''(\xi).$$

- When evaluating $f(x_0+h)$ and $f(x_0-h)$ with computer, there are round-off errors, denoted by $e(x_0+h)$ and $e(x_0-h)$. Then our computational actually use the values $\tilde{f}(x_0 \pm h)$ given by

$$f(x_0+h) = \tilde{f}(x_0+h) + e(x_0+h), \quad f(x_0-h) = \tilde{f}(x_0-h) + e(x_0-h).$$

- The total error in the approximation is

$$\begin{aligned} & f'(x_0) - \frac{\tilde{f}(x_0+h) - \tilde{f}(x_0-h)}{2h} \\ = \; & f'(x_0) - \frac{f(x_0+h) - f(x_0-h)}{2h} + \frac{e(x_0+h) - e(x_0-h)}{2h} \\ = \; & -\frac{h^2}{6} f'''(\xi) + \frac{e(x_0+h) - e(x_0-h)}{2h}. \end{aligned}$$

- The first term is the approximation error, and the second term is round-off error.

- If we assume that the round-off errors and the third derivative f''' are bounded as follows

$$|e(x_0 \pm h)| \leq \varepsilon, \qquad |f'''(x)| \leq M,$$

then the total computational error

$$\begin{aligned} & f'(x_0) - \frac{\tilde{f}(x_0+h) - \tilde{f}(x_0-h)}{2h} \\ \leq & \frac{h^2}{6}|f'''(\xi)| + \frac{|e(x_0+h) - e(x_0-h)|}{2h} \\ \leq & \frac{h^2}{6}|f'''(\xi)| + \frac{|e(x_0+h)| + |e(x_0-h)|}{2h} \\ \leq & \frac{Mh^2}{6} + \frac{\varepsilon}{h}. \end{aligned}$$

- Note that as h is reduced, the approximation error $Mh^2/6$ decreases. However, the round-off error ε/h grows. If h is too small (e.g. $h < 10^{-6}$ as in previous example), the round-off error dominates the calculation.

- Now we analyze why h around 10^{-6} and 10^{-5} yields the most accurate approximation in Example 3. Note $f(x) = \sin(x)$, so $f'''(x) = -\cos(x)$.
- Around $x_0 = 0.9$, we consider the interval $[0.8, 1]$,

$$M = \max_{x\in[0.8,1]} |f'''(x)| = \max_{x\in[0.8,1]} |\cos(x)| = \cos(0.8) \approx 0.69671.$$

- In a 64-bit computer, the machine precision $\varepsilon \approx 2.22 \times 10^{-16}$. Hence, the total error

$$e(h) = \frac{Mh^2}{6} + \frac{\varepsilon}{h}$$

will reach its minimum where $e'(h) = \dfrac{Mh}{3} - \dfrac{\varepsilon}{h^2} = 0$. That is

$$h = \left(\frac{3\varepsilon}{M}\right)^{1/3} = \left(\frac{3 \times 2.22^{-16}}{0.69671}\right)^{1/3} \approx 0.0000098515.$$

- The value of h is **optimal** for the 3pt-midpoint formula on a 64-bit computer. This is consistent with the numerical experiment results, i.e., h is between 10^{-6} and 10^{-5}.

- Similar argument can be used to analyze why the forward difference formula reaches the best accuracy around $h = 10^{-8}$. (the first example of this section). (**Exercise**)

- In practice, we cannot compute an optimal h to use in approximating the derivative, since we have no knowledge of the derivatives of the function. But we must remain aware that reducing the step size h will not always improve the approximation.

- Numerical differentiation, as an approximation method, is **unstable**, since the small values of h needed to reduce truncation error also cause the round-off error to grow.

4.2 Elements of Numerical Integration

- In practical applications, we often need to evaluate the definite integral of a function

$$\int_a^b f(x)dx. \tag{4.9}$$

- However, the antiderivative is often hard to obtain, sometimes there is no explicit antiderivative.
- In this section, we introduce how to numerically approximate of the integral (4.9). The method is called **numerical quadrature**.
- The quadrature in this section is based on interpolation.

- Let $\{x_0, x_1, \cdots, x_n\}$ be a set of nodes from $[a, b]$. The nth-degree Lagrange interpolating polynomial is

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

- The idea is to use the integral of P_n to approximate the integral of f:

$$\begin{aligned} \int_a^b f(x)dx &\approx \int_a^b P_n(x)dx = \int_a^b \sum_{i=0}^{n} f(x_i) L_i(x)dx \\ &= \sum_{i=0}^{n} f(x_i) \int_a^b L_i(x)dx \triangleq \sum_{i=0}^{n} a_i f(x_i). \end{aligned}$$

- The quadrature formula is

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i), \quad \text{where} \quad a_i = \int_a^b L_i(x)dx, \quad i = 0, 1, \cdots, n, \tag{4.10}$$

- The error of the numerical quadrature (4.10) is given by

$$E(f) = \frac{1}{(n+1)!} \int_a^b f^{(n+1)}(\xi(x)) \prod_{i=0}^{n} (x - x_i)dx.$$

Quadrature formula using first Lagrange polynomials

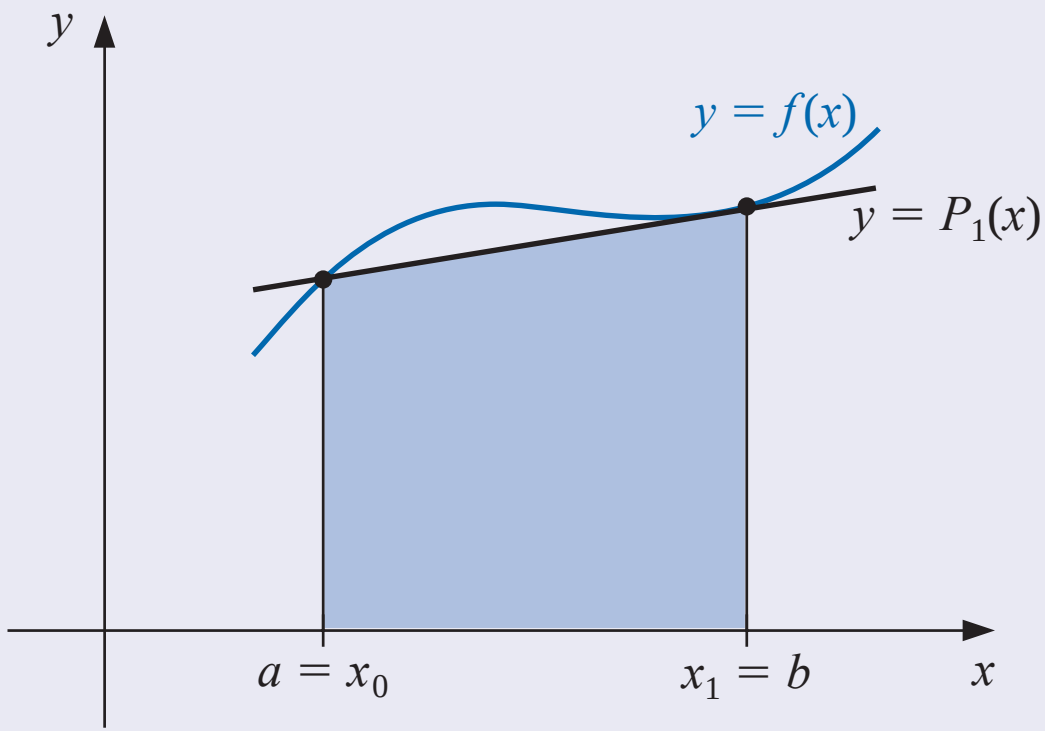
- Let $x_0 = a$, $x_1 = b$, and $h = b - a$. Then

$$P_1(x) = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0}.$$

- Then, the integral can be approximated by

$$\begin{aligned}
\int_a^b f(x)dx &\approx \int_{x_0}^{x_1} \left[f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0} \right] dx \\
&= f(x_0)\left. \frac{(x - x_1)^2}{2(x_0 - x_1)} \right|_{x_0}^{x_1} + f(x_1)\left. \frac{(x - x_0)^2}{2(x_1 - x_0)} \right|_{x_0}^{x_1} \\
&= -f(x_0)\frac{x_0 - x_1}{2} + f(x_1)\frac{x_1 - x_0}{2} \\
&= \frac{h}{2}\Big[f(x_0) + f(x_1) \Big].
\end{aligned}$$

Illustration of Numerical Quadrature



- The integral $\int_a^b f(x)dx$, which is the area of the curved rectangle, is approximated by the area of a trapezoid.
- This approximation gives no error if the function f is a linear polynomial. We will analyze this error in the next slide.

- To estimate the error of the quadrature $E(f)$, we use the Weighted Mean Value Theorem

$$\begin{aligned}
E(f) &\triangleq \int_a^b f(x) - P_1(x)dx \\
\text{by Error of Interpolation} &= \frac{1}{2}\int_{x_0}^{x_1} f''(\xi(x))(x-x_0)(x-x_1)dx \\
\text{by Weighted MVT} &= \frac{1}{2}f''(\xi_1)\int_{x_0}^{x_1}(x-x_0)(x-x_1)dx \\
&= \frac{1}{2}f''(\xi_1)\left[\frac{x^3}{3} - \frac{x_0+x_1}{2}x^2 + x_0x_1x\right]_{x_0}^{x_1} \\
&= -\frac{h^3}{12}f''(\xi_1).
\end{aligned}$$

Trapezoidal Rule

$$\int_a^b f(x)dx = \frac{h}{2}\Big[f(x_0)+f(x_1)\Big] - \frac{h^3}{12}f''(\xi). \tag{4.11}$$

Here $h = b-a$, $x_0 = a$, $x_1 = b$, and ξ is between x_0 and x_1.

Quadrature using Second Lagrange polynomials

- Let $h = \dfrac{b-a}{2}$, $x_0 = a$, $x_1 = a + h$, and $x_2 = b$. Then

$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_0}^{x_2} \left[f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \right. \\ &\qquad \left. + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \right] dx + \int_{x_0}^{x_2} E_2(x)dx \\ &= \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] - \frac{h^5}{90} f^{(4)}(\xi).\end{aligned}$$

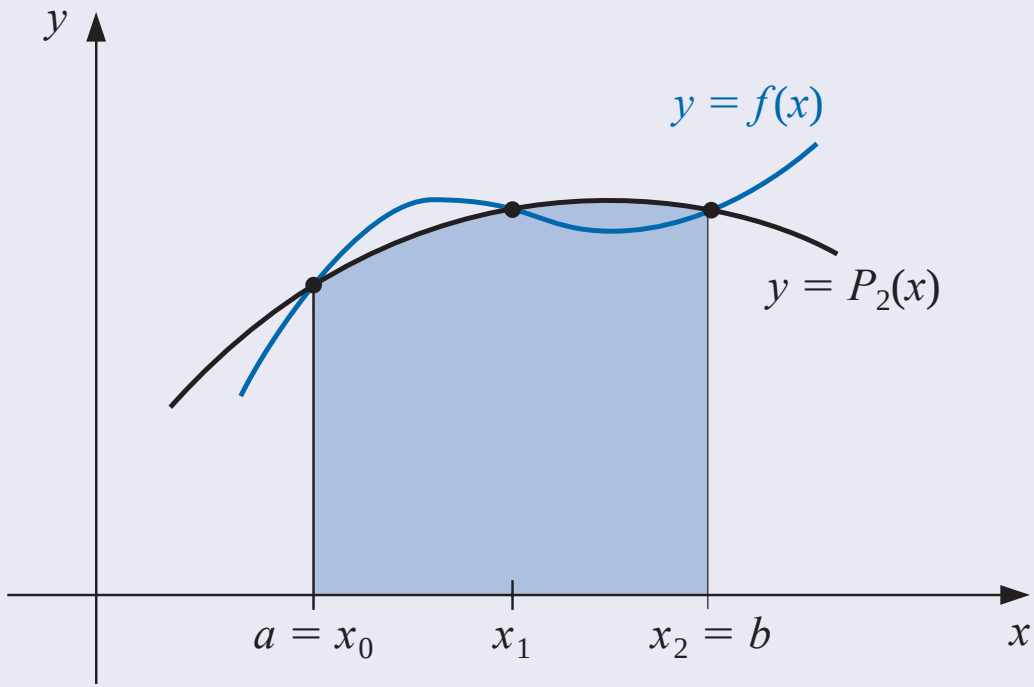
- It is good exercise to go through the above calculation. We obtain

Simpson's Rule

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] - \frac{h^5}{90} f^{(4)}(\xi), \tag{4.12}$$

where $h = (b-a)/2$, $x_0 = a$, $x_1 = a + h$, and $x_2 = b$.

Illustration of Simpson's Rule



- The error term for (4.12) has a factor $f^{(4)}$, so Simpson's rule is exact for any polynomial of degree up to three.

Example 4

Compare Trapezoidal rule and Simpson's rule in approximating $\int_0^2 f(x)dx$ when $f(x)$ is

(a) x, (b) x^2, (c) x^4, (d) $(1+x)^{-1}$, (e) $\sin(x)$

Solution

- For Trapezoidal rule, we have $h = 2$, $x_0 = 0$, $x_1 = 2$, then

$$\int_0^2 f(x)dx \approx f(0) + f(2).$$

- For Simpson's rule, we have $h = 1$, $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, then

$$\int_0^2 f(x)dx \approx \frac{1}{3}\left[f(0) + 4f(1) + f(2)\right].$$

$f(x)$	x	x^2	x^4	$(1+x)^{-1}$	$\sin(x)$
Trapezoidal	2.0000	4.0000	16.0000	1.3333	0.9093
Simpson	2.0000	2.6667	6.6667	1.1111	1.4251
Exact	2.0000	2.6667	6.4000	1.0986	1.4161

Measuring Precision

How to measure the precision of a numerical quadrature formula?

Definition 5 (degree of precision)

The **degree of precision** of a quadrature formula is the largest positive integer n such that the formula is exact for x^k, for each $k = 0, 1, \cdots, n$.

Remark

- The DoP of the Trapezoidal rule is one.
- The DoP of the Simpson's rule is three.
- The DoP of a quadrature formula is n if and only if the error is zero for all polynomials of degree up to n, but not zero for some polynomial of degree $n + 1$.

Example 6

Find the degree of the precision of the quadrature formula

$$\int_{-1}^{1} f(x)dx = f\Big(\frac{-\sqrt{3}}{3}\Big) + f\Big(\frac{\sqrt{3}}{3}\Big).$$

Solution (1/2)

- By definition of DoP, we will test $f(x) = 1, x, x^2, \cdots$, until we find a number k such that the quadrature formula does not hold for x^k.

$$f(x) = 1, \quad \int_{-1}^{1} 1dx = 2, \quad f\left(-\sqrt{3}/3\right) + f\left(\sqrt{3}/3\right) = 1 + 1 = 2.$$

$$f(x) = x, \quad \int_{-1}^{1} xdx = 0, \quad f\left(-\sqrt{3}/3\right) + f\left(\sqrt{3}/3\right) = -\frac{\sqrt{3}}{3} + \frac{\sqrt{3}}{3} = 0.$$

$$f(x) = x^2, \quad \int_{-1}^{1} x^2dx = \frac{2}{3}, \quad f\left(-\sqrt{3}/3\right) + f\left(\sqrt{3}/3\right) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}.$$

Solution (2/2)

$$f(x) = x^3, \quad \int_{-1}^{1} x^3 dx = 0, \quad f\left(-\sqrt{3}/3\right) + f\left(\sqrt{3}/3\right) = -\frac{1}{3\sqrt{3}} + \frac{1}{3\sqrt{3}} = 0.$$

$$f(x) = x^4, \quad \int_{-1}^{1} x^4 dx = \frac{2}{5}, \quad f\left(-\sqrt{3}/3\right) + f\left(\sqrt{3}/3\right) = \frac{1}{9} + \frac{1}{9} = \frac{2}{9}.$$

The formula is exact for polynomial of degree up to 3, but not accurate for degree 4. The degree of precision is 3.

Newton-Cotes Formulas

- Trapezoidal rule and Simpson's rule are examples of a larger class of methods called **Newton-Cotes Formulas**.
- There are two types of Newton-Cotes formulas: **open and closed Newton-Cotes Formulas**, depending on whether or not to include the endpoints in the quadrature formulas.

Closed Newton-Cotes Formula

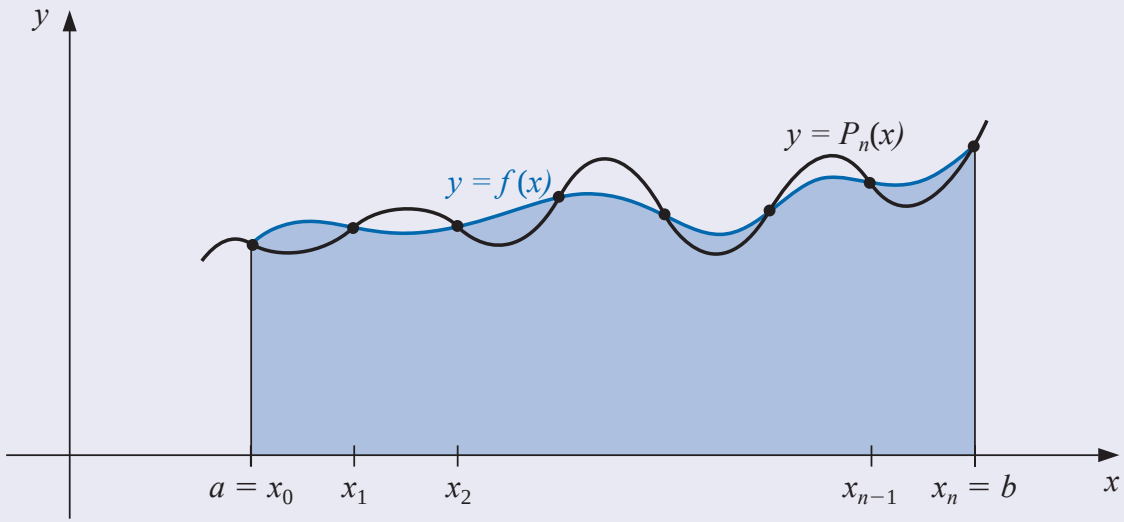
The $(n+1)$-point closed Newton-Cotes quadrature formula is

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i), \tag{4.13}$$

where

- $h = (b-a)/n$, $x_0 = a$, and $x_i = x_0 + ih$.
- $a_i = \int_{x_0}^{x_n} L_i(x)dx = \int_{x_0}^{x_n} \prod_{j=0, j\neq i} \frac{(x-x_j)}{(x_i-x_j)}dx.$

Illustration of Closed Newton-Cotes Formula



Theorem 7 (Error Bound for Closed Newton-Cotes Formulas)

The $(n+1)$-point closed Newton-Cotes formula obeys the following error estimates

$$\int_a^b f(x)dx = \sum_{i=0}^{n} a_i f(x_i) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_0^n t^2(t-1)\cdots(t-n)dt$$

if n is even and $f \in C^{n+2}(a,b)$, and

$$\int_a^b f(x)dx = \sum_{i=0}^{n} a_i f(x_i) + \frac{h^{n+2} f^{(n+1)}(\xi)}{(n+1)!} \int_0^n t(t-1)\cdots(t-n)dt$$

if n is odd and $f \in C^{n+1}(a,b)$.

Several Closed Newton-Cotes Formulas

For $0 \le i \le n$, $x_i = a + ih$ and $h = \dfrac{b-a}{n}$.

n	h	Formula	Error	Name
1	$b-a$	$\frac{h}{2}(f_0 + f_1)$	$-\frac{h^3}{12} f''(\xi)$	Trapezoidal Rule
2	$\frac{b-a}{2}$	$\frac{h}{3}(f_0 + 4f_1 + f_2)$	$-\frac{h^5}{90} f^{(4)}(\xi)$	Simpson's Rule
3	$\frac{b-a}{3}$	$\frac{3h}{8}(f_0 + 3f_1 + 3f_2 + f_3)$	$-\frac{3h^5}{80} f^{(4)}(\xi)$	Simpson's 3/8 Rule
4	$\frac{b-a}{4}$	$\frac{2h}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4)$	$-\frac{8h^7}{945} f^{(6)}(\xi)$	Boole's Rule

Here, we use the simplified notation $f_i = f(x_i)$.

- Another class of Newton-Cotes quadrature formulas do not include the endpoints of $[a, b]$ as nodes for interpolation. To unify the notation, we label these endpoints $x_{-1} = a$ and $x_{n+1} = b$.

Open Newton-Cotes Formula

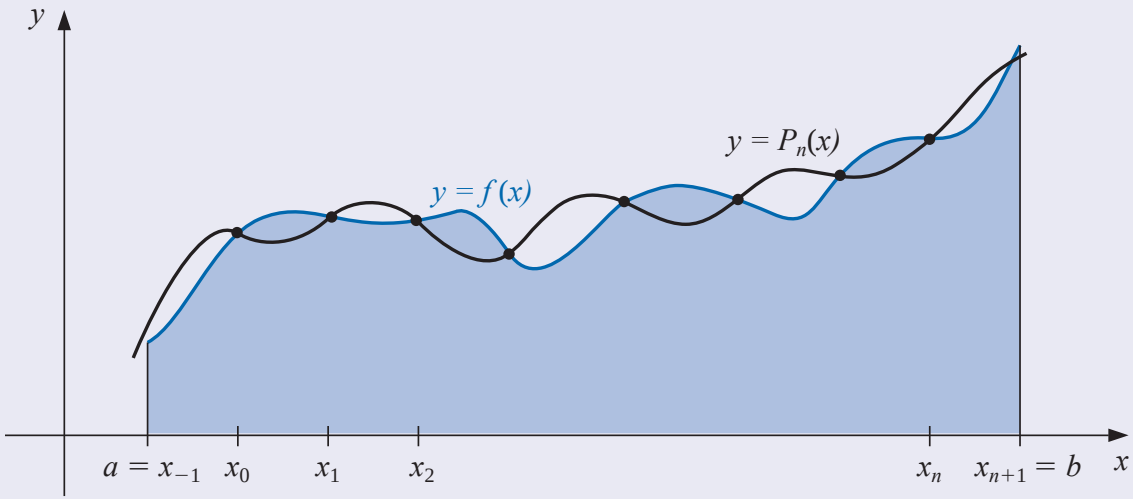
The $(n+1)$-point open Newton-Cotes quadrature formula is

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i), \tag{4.14}$$

where

- $h = \dfrac{b-a}{n+2}$, $a = x_{-1} < x_0 < x_1 < \cdots < x_n < x_{n+1} = b$, and $x_i = x_0 + ih$
- $a_i = \displaystyle\int_{x_{-1}}^{x_{n+1}} L_i(x)dx = \int_{x_{-1}}^{x_{n+1}} \prod_{j=0, j\neq i} \frac{(x - x_j)}{(x_i - x_j)} dx.$

Illustration of Open Newton-Cotes Formula



Theorem 8 (Error Bound for Open Newton-Cotes Formulas)

The $(n+1)$-point closed Newton-Cotes formula obeys the following error estimates

$$\int_a^b f(x)dx = \sum_{i=0}^{n} a_i f(x_i) + \frac{h^{n+3} f^{(n+2)}(\xi)}{(n+2)!} \int_{-1}^{n+1} t^2(t-1)\cdots(t-n)dt$$

if n is even and $f \in C^{n+2}(a,b)$, and

$$\int_a^b f(x)dx = \sum_{i=0}^{n} a_i f(x_i) + \frac{h^{n+2} f^{(n+1)}(\xi)}{(n+1)!} \int_{-1}^{n+1} t(t-1)\cdots(t-n)dt$$

if n is odd and $f \in C^{n+1}(a,b)$.

Several Open Newton-Cotes Formulas

Let $h = \dfrac{b-a}{n+2}$ and $x_0 = a + h$. For $0 \le i \le n$, let $x_i = x_0 + ih$.

n	h	Formula	Error	Name
0	$\dfrac{b-a}{2}$	$2hf_0$	$\dfrac{h^3}{3}f''(\xi)$	Midpoint rule
1	$\dfrac{b-a}{3}$	$\dfrac{3h}{2}(f_0+f_1)$	$\dfrac{3h^3}{4}f''(\xi)$	N/A
2	$\dfrac{b-a}{4}$	$\dfrac{4h}{3}(2f_0-f_1+2f_2)$	$\dfrac{28h^5}{90}f^{(4)}(\xi)$	N/A
3	$\dfrac{b-a}{5}$	$\dfrac{5h}{24}(11f_0+f_1+f_2+11f_3)$	$\dfrac{95h^5}{144}f^{(4)}(\xi)$	N/A

Here, $f_i = f(x_i)$.

Example 9

Compare the results of the closed and open Newton-Cotes formulas when approximating

$$\int_0^{\pi/4} \sin(x)dx \approx 0.29289322.$$

Solution (1/5)

- We write two MATLAB subroutines for closed and open Newton-Cotes formulas.
- The subroutines should contain the following inputs:
  1. function f, 2. interval $[a, b]$, 3, number of points

Solution (2/5): Subroutine for Closed NC formulas

```
function y = NewtonCotesCl(fun,a,b,n)

h = (b-a)/n;
if n == 1 % Trapezoidal rule
    x0 = a; x1 = x0+h;
    f0 = fun(x0); f1 = fun(x1);
    y = (h/2)*(f0+f1);
elseif n == 2 % Simpson's rule
    x0 = a; x1 = x0+h; x2 = x0+2*h;
    f0 = fun(x0); f1 = fun(x1); f2 = fun(x2);
    y = (h/3)*(f0+4*f1+f2);
elseif n == 3 % Simpson's 3/8 rule
    x0 = a; x1 = x0+h; x2 = x0+2*h; x3 = x0+3*h;
    f0 = fun(x0); f1 = fun(x1); f2 = fun(x2); f3 = fun(x3);
    y = (3*h/8)*(f0+3*f1+3*f2+f3);
elseif n == 4 % Boole's rule
    x0 = a; x1 = x0+h; x2 = x0+2*h; x3 = x0+3*h; x4 = x0+4*h;
    f0 = fun(x0); f1=fun(x1); f2=fun(x2); f3=fun(x3); f4=fun(x4);
    y = (2*h/45)*(7*f0+32*f1+12*f2+32*f3+7*f4);
end
```

Solution (3/5): Subroutine for Open NC formulas

```
function y = NewtonCotesOp(fun,a,b,n)

h = (b-a)/(n+2);
if n == 0 % Midpoint rule
    x0 = a+h;
    f0 = fun(x0);
    y = (2*h)*f0;
elseif n == 1
    x0 = a+h; x1 = x0+h;
    f0 = fun(x0); f1 = fun(x1);
    y = (3*h/2)*(f0+f1);
elseif n == 2
    x0 = a+h; x1 = x0+h; x2 = x0+2*h;
    f0 = fun(x0); f1 = fun(x1); f2 = fun(x2);
    y = (4*h/3)*(2*f0-f1+2*f2);
elseif n == 3
    x0 = a+h; x1 = x0+h; x2 = x0+2*h; x3 = x0+3*h;
    f0 = fun(x0); f1=fun(x1); f2=fun(x2); f3=fun(x3);
    y = (5*h/24)*(11*f0+f1+f2+11*f3);
end
```

Solution (4/5): A MATLAB Driver File

```
% ex_4_2_1
clear
fun = @(x) sin(x);
a = 0;
b = pi/4;

y1c = NewtonCotesCl(fun,a,b,1);
y2c = NewtonCotesCl(fun,a,b,2);
y3c = NewtonCotesCl(fun,a,b,3);
y4c = NewtonCotesCl(fun,a,b,4);

y0o = NewtonCotesOp(fun,a,b,0);
y1o = NewtonCotesOp(fun,a,b,1);
y2o = NewtonCotesOp(fun,a,b,2);
y3o = NewtonCotesOp(fun,a,b,3);

y = -cos(pi/4)+cos(0);

formatSpec = '%2i    % .8f    % .8f      % .8f   \n';
disp('Closed Newton Cotes Formula')
disp(' n     True           Close N-C        Error')
disp('-------------------------------------------------')
fprintf(formatSpec,1,y,y1c,abs(y-y1c))
fprintf(formatSpec,2,y,y2c,abs(y-y2c))
fprintf(formatSpec,3,y,y3c,abs(y-y3c))
fprintf(formatSpec,4,y,y4c,abs(y-y4c))
disp(' ')
disp('Open Newton Cotes Formula')
disp(' n     True           Open N-C         Error')
disp('-------------------------------------------------')
fprintf(formatSpec,0,y,y0o,abs(y-y0o))
fprintf(formatSpec,1,y,y1o,abs(y-y1o))
fprintf(formatSpec,2,y,y2o,abs(y-y2o))
fprintf(formatSpec,3,y,y3o,abs(y-y3o))
```

Solution (5/5): Output from MATLAB Command Window

```
>> ex4_2_1
Closed Newton Cotes Formula
 n     True            Close N-C         Error
-------------------------------------------------
 1     0.29289322      0.27768018        0.01521304
 2     0.29289322      0.29293264        0.00003942
 3     0.29289322      0.29291070        0.00001748
 4     0.29289322      0.29289318        0.00000004

Open Newton Cotes Formula
 n     True            Open N-C          Error
-------------------------------------------------
 0     0.29289322      0.30055886        0.00766565
 1     0.29289322      0.29798754        0.00509432
 2     0.29289322      0.29285866        0.00003456
 3     0.29289322      0.29286923        0.00002399
```

4.3 Composite Numerical Integration

- The Newton-Cotes formula is generally not suitable for large integrating intervals.
- High-degree formulas would be required, and the values of the coefficients in these formulas are difficult to obtain.
- Newton-Cotes formulas are based on interpolating polynomials that use equally-spaced nodes, a procedure that is inaccurate over large intervals because of the oscillatory nature of high-degree polynomials.
- In this section, we discuss a piecewise approach to numerical integration that uses the low-order Newton-Cotes formulas. These are the techniques most often applied.

Example 10

In this example, we apply the Simpson's rule to approximate $\int_0^4 e^x dx$, whose true value is 53.59815. We will do it in the three ways.

1. approximate $\int_0^4 e^x dx$ directly.
2. approximate separately on $\int_0^2 e^x dx$ and $\int_2^4 e^x dx$
3. approximate separately on $\int_0^1 e^x dx$, $\int_1^2 e^x dx$, $\int_2^3 e^x dx$, and $\int_3^4 e^x dx$

Solution (1/3)

- Directly applying the Simpson's rule on $[0, 4]$, we have $h = \dfrac{4-0}{2} = 2$, then,

$$\int_0^4 e^x dx \approx \frac{2}{3}[e^0 + 4e^2 + e^4] = 56.76958.$$

- The relative error is 0.0592.

Solution (2/3)

- Now we use Simpson's rule separately on $[0, 2]$ and $[2, 4]$.
- Note that on each subinterval $h = \dfrac{2-0}{2} = \dfrac{4-2}{2} = 1$, then,

$$\begin{aligned}
\int_0^4 e^x dx &= \int_0^2 e^x dx + \int_2^4 e^x dx \\
&\approx \frac{1}{3}[e^0 + 4e^1 + e^2] + \frac{1}{3}[e^2 + 4e^3 + e^4] \\
&= 53.86385.
\end{aligned}$$

- The relative error is 0.00496.

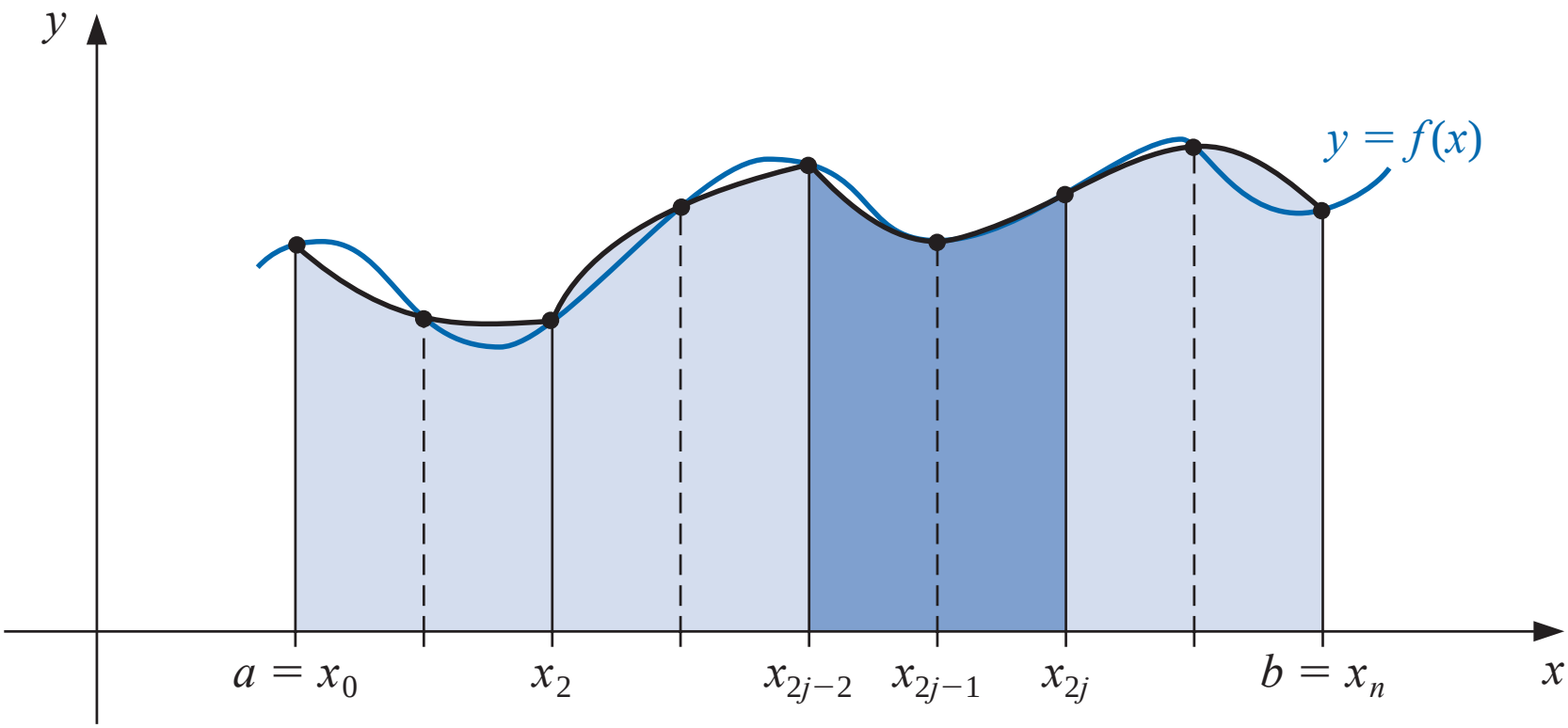
Solution (3/3)

- Next we use Simpson's rule separately on $[0,1]$, $[1,2]$, $[2,3]$ and $[3,4]$.
- Note that on each subinterval $h = \dfrac{1-0}{2} = \dfrac{1}{2}$, then,

$$\begin{aligned}
\int_0^4 e^x dx &= \int_0^1 e^x dx + \int_1^2 e^x dx + \int_2^3 e^x dx + \int_3^4 e^x dx \\
&\approx \frac{1}{6}[e^0 + 4e^{1/2} + e^1] + \frac{1}{6}[e^1 + 4e^{3/2} + e^2] \\
&\quad + \frac{1}{6}[e^2 + 4e^{5/2} + e^3] + \frac{1}{6}[e^3 + 4e^{7/2} + e^4] \\
&= \frac{1}{6}[e^0 + 4e^{1/2} + 2e^1 + 4e^{3/2} + 2e^2 + 4e^{5/2} + 2e^3 + 4e^{7/2} + e^4] \\
&= 53.61622.
\end{aligned}$$

- The relative error is 0.000337.

In general, to approximate $\int_a^b f(x)dx$, choose an even number n, subdivide $[a, b]$ to n subintervals, and apply Simpson's rule on each consecutive pair of subintervals.



- Let n be an even number. Let $h = \dfrac{b-a}{n}$, and $x_j = a + jh$, $j = 0, 1, \cdots, n$.

$$\begin{aligned}
\int_a^b f(x)dx &= \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)dx \\
&\approx \frac{h}{3}\Big[f(x_0) + 4f(x_1) + f(x_2)\Big] + \frac{h}{3}\Big[f(x_2) + 4f(x_3) + f(x_4)\Big] \\
&\qquad + \cdots + \frac{h}{3}\Big[f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\Big] \\
&= \frac{h}{3}\Big[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots \\
&\qquad + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\Big] \\
&= \frac{h}{3}\Big[f(x_0) + 4\big(f(x_1) + f(x_3) + \cdots + f(x_{n-1})\big) \\
&\qquad + 2\big(f(x_2) + f(x_4) + \cdots + f(x_{n-2})\big) + f(x_n)\Big].
\end{aligned}$$

Composite Simpson's Rule

The composite Simpson's rule on the interval $[a, b]$ is given by

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{(n/2)-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b)\right]. \quad (4.15)$$

where n is even, $h = \dfrac{b-a}{n}$, and $x_j = a + jh$ for each $j = 0, 1, \cdots, n$.

Theorem 11 (Error of Composite Simpson's rule)

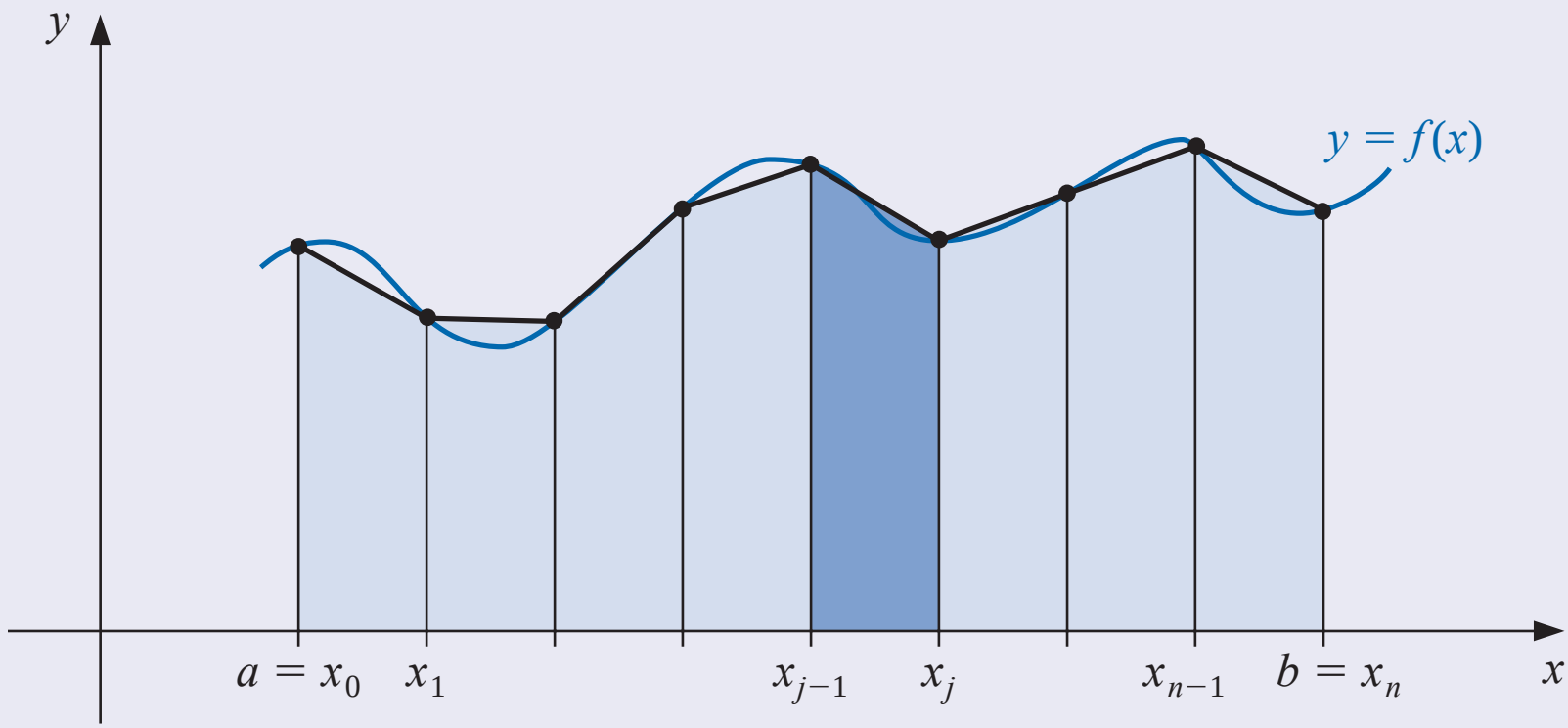
Assume $f \in C^4[a, b]$, then the error of composite Simpson's rule is

$$E(f) = -\frac{b-a}{180}h^4 f^{(4)}(\mu) \quad (4.16)$$

for some $\mu \in (a, b)$.

The subdivision approach can be applied to any Newton-Cotes formula. For example, the composite Trapezoidal rule requires only one interval for each application, so the integer n can be either odd or even.

Illustration of Composite Trapezoidal Rule



Composite Trapezoidal Rule

Suppose that the interval $[a, b]$ is split up into n sub-intervals. The composite trapezoidal rule is given by

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] \tag{4.17}$$

where $x_j = a + jh$ for each $j = 0, 1, \cdots, n$, with $h = (b - a)/n$.

Theorem 12 (Error of Composite Trapezoidal Rule)

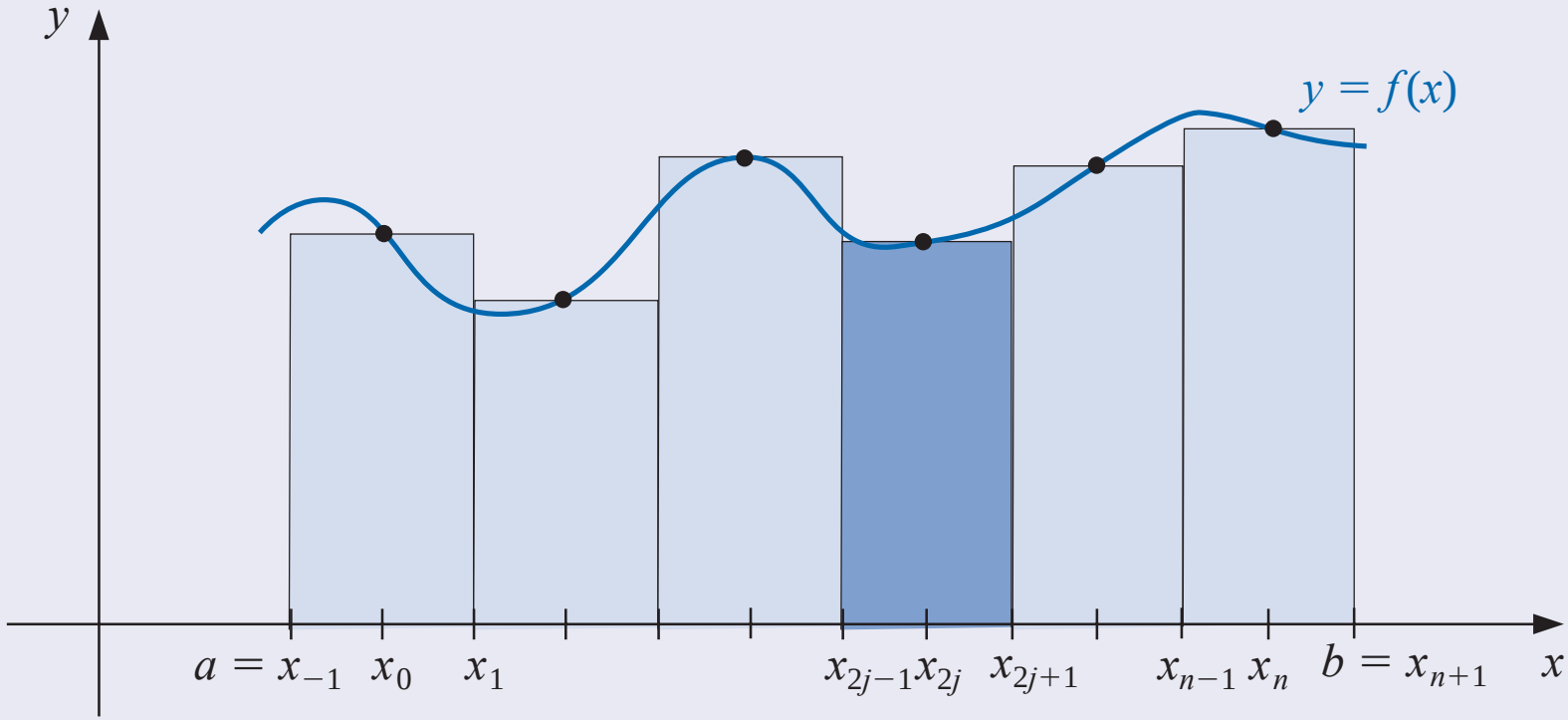
Assume $f \in C^2[a, b]$, then the error of composite trapezoidal rule is

$$E(f) = -\frac{b-a}{12}h^2 f''(\mu) \tag{4.18}$$

for some $\mu \in (a, b)$.

Similarly, we can also apply the subdivision idea to the midpoint rule.

Illustration of Composite Midpoint Rule



Composite Midpoint Rule

Suppose that the interval $[a, b]$ is split up into $n + 1$ sub-intervals. The composite midpoint rule is given by

$$\int_a^b f(x)dx = 2h \sum_{j=0}^{n/2} f(x_{2j}) \tag{4.19}$$

where $x_j = a + (j + 1)h$ for each $-1 \leq j \leq n + 1$, with $h = \dfrac{b - a}{n + 2}$.

Theorem 13 (Error of Composite Midpoint Rule)

Assume $f \in C^2[a, b]$, then the error of composite midpoint rule is

$$E(f) = -\frac{b - a}{6} h^2 f''(\mu) \tag{4.20}$$

for some $\mu \in (a, b)$.

Example 14

Determine values of h that will ensure an approximation error of less than 0.00002 when approximating

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

by (a) Composite Trapezoidal rule and (b) Composite Simpson's rule.

Solution. (1/3)

- Note that the error bound of composite trapezoidal rule is

$$E(f) = -\frac{b-a}{12}h^2 f''(\mu).$$

- On the interval $[0, \pi]$, we have

$$E(f) \leq \left|\frac{\pi h^2}{12} f''(\mu)\right| = \frac{\pi h^2}{12}|-\sin(\mu)| \leq \frac{\pi h^2}{12} < 0.00002 \implies h < 0.008740.$$

Solution. (2/3)

- Since $h = \pi/n$, then

$$\frac{\pi}{n} < 0.008740 \quad \Longrightarrow \quad n > 359.4$$

We need at least 360 subintervals to ensure the desired accuracy.

- The error bound of composite Simpson's rule is

$$\left|\frac{\pi h^4}{180} f^{(4)}(\mu)\right| = \frac{\pi h^4}{180}|\sin(\mu)| \leq \frac{\pi h^4}{180} < 0.00002 \quad \Longrightarrow \quad h < 0.1840.$$

- Since $h = \pi/n$, then

$$\frac{\pi}{n} < 0.1840 \quad \Longrightarrow \quad n > 17.08$$

We need at least 18 subintervals to ensure the desired accuracy.

Solution. (3/3)

- We write a MATLAB subroutine for composite Simpson's rule, and a driver file for testing this example.

```
function Q = simpsonComp(fun,a,b,n)

if mod(n,2) ~= 0
    warning('n must be an even number. Replaced n by n+1');
    n = n + 1;
end

h = (b-a)/n;
Q = h/3*(fun(a)+fun(b));

for j = 1:(n/2)-1
    Q = Q + (h/3)*2*fun(a+2*j*h) + (h/3)*4*fun(a+(2*j-1)*h);
end
Q = Q + (h/3)*4*fun(a+(n-1)*h);

end
```

```
% ex4_4_2

fun = @(x) sin(x);
a = 0;
b = pi;
n = 18;
Q = simpsonComp(fun,a,b,n)
```

- Using composite Simpson's rule with 18 subintervals, we obtain

$$\int_0^{\pi} \sin(x)dx \approx 2.0000103.$$

Example 15

A car laps a race track in 84 seconds. The speed of the car at each 6-second interval is determined by using a radar gun and is given from the beginning of the lap, in feet/second, by the entries in the following table.

Time	0	6	12	18	24	30	36	42	48	54	60	66	72	78	84
Speed	124	134	148	156	147	133	121	109	99	85	78	89	104	116	123

How long is the track?

Solution. (1/2)

- Note that the length l of the track is $l = \int_0^{84} s(t)dt$.
- We don't know the formula of $s(t)$. So, the integral can only be approximated by numerical quadrature using the above 15 data points:

$$\int_0^{84} s(t)dt \approx \sum_{i=1}^{15} a_i s(t_i).$$

Solution. (2/2)

- Using composite trapezoidal rule with $h = 6$,

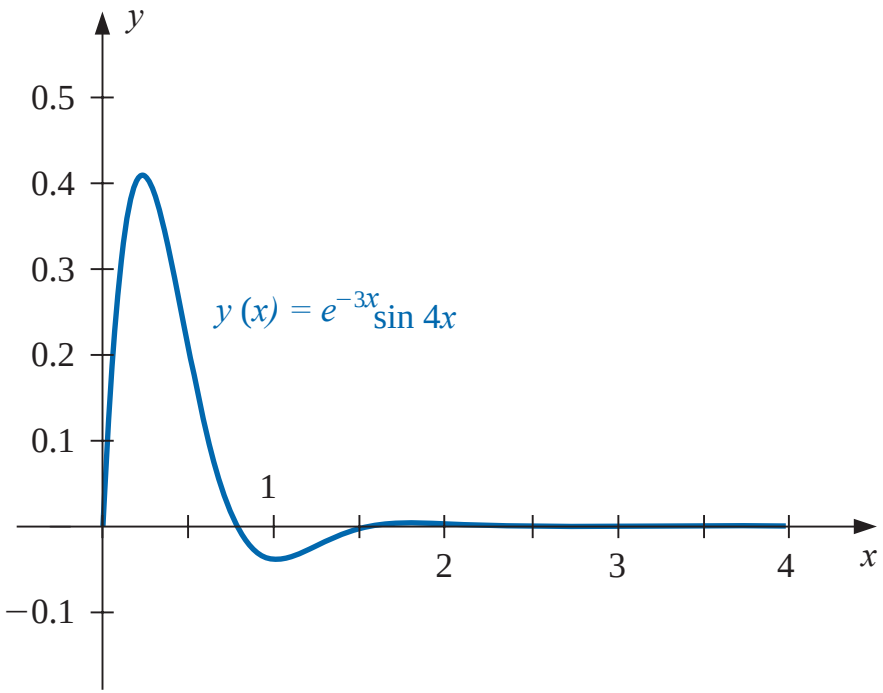
$$\begin{aligned} l = \int_0^{84} s(t)dt &\approx \frac{6}{2}\Big[124 + 2(134 + 148 + \cdots + 116) + 123\Big] \\ &= 9855 \end{aligned}$$

- Using composite Simpson's rule with $h = 6$,

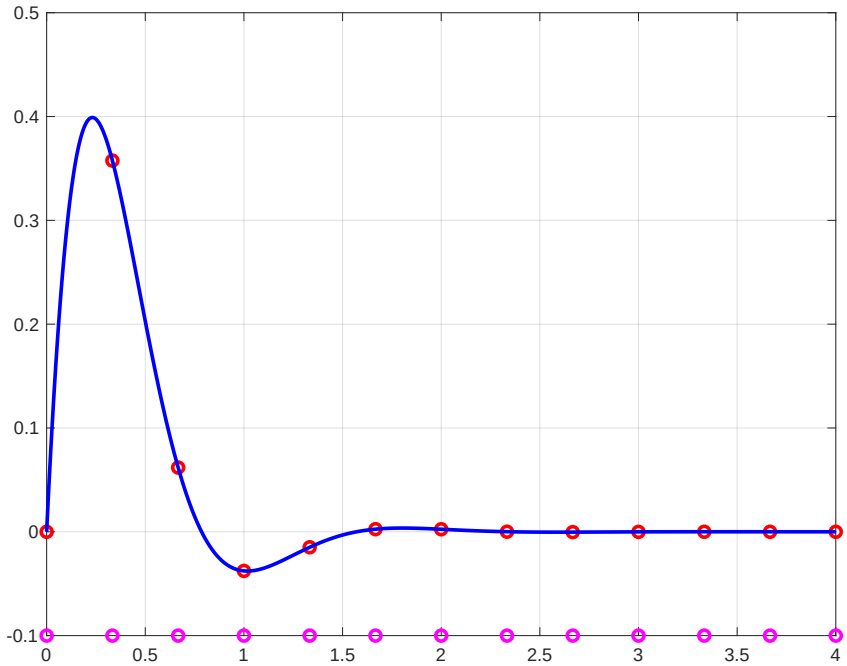
$$\begin{aligned} l = \int_0^{84} s(t)dt &\approx \frac{6}{3}\Big[124 + 4(134 + 156 + 133 + 109 + 85 + 89 + 116) \\ &\qquad +2(148 + 147 + 121 + 99 + 78 + 104) + 123\Big] \\ &= 9858 \end{aligned}$$

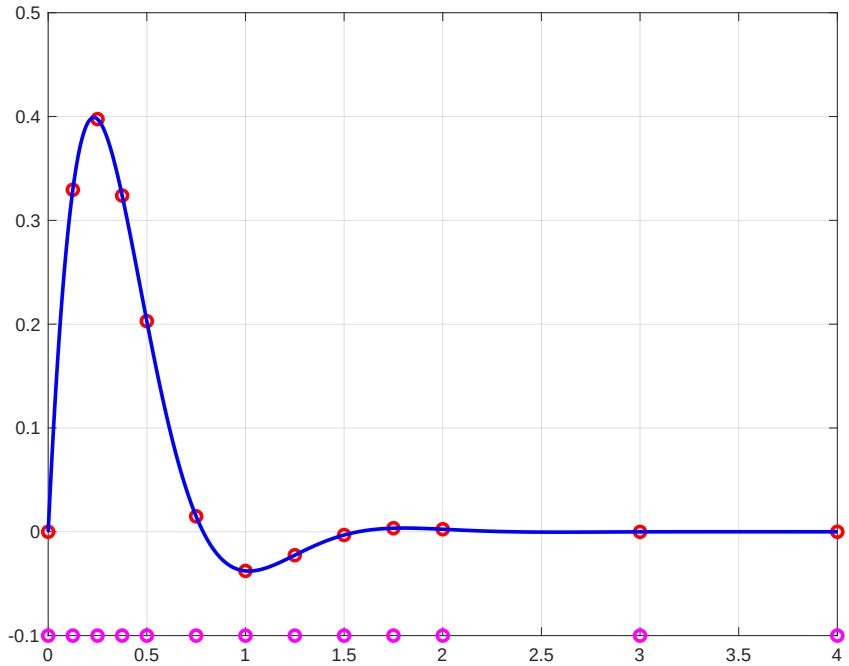
4.4 Adaptive Quadrature Method

- The composite formulas, such as composite Simpson' rule, are very effective in most situations, but they suffer occasionally because they require the use of equally-spaced nodes.
- This is inappropriate when integrating a function on an interval that contains both regions with large functional variation and regions with small functional variation. For example, consider the integral of the function $y = e^{-3x}\sin(4x)$ over the interval $[0, 4]$.



- The true integral value is $\int_0^4 e^{-3x}\sin(4x)dx = 0.160001$.
- Using Composite Simpson's rule with 12 equal-spaced points, we obtain the quadrature $Q_1 = 0.154225$. The relative error is 3.61%.





- If these 12 points are distributed more "smartly", i.e., more nodes in the region with large functional variation. Using the same quadrature rule, we obtain $Q_2 = 0.160101$. The relative error is 0.0626%.

- How to determine what technique should be applied on various portions of the interval of integration, and how accurate can we expect the final approximation to be?
- If the approximation error for an integral on a given interval is to be evenly distributed, a smaller step size is needed for the large-variation regions than for those with less variation.
- An efficient technique should predict the amount of functional variation and adapt the step size as necessary. These methods are called **Adaptive quadrature methods**.
- Adaptive methods are particularly popular for inclusion in professional software packages because, in addition to being efficient, they generally provide approximations that are within a given specified tolerance.

- In this section, we consider the adaptive Simpson's rule, but the technique can be modified to use other composite methods.
- Suppose we want to approximate $\int_a^b f(x)dx$ by the Simpson's rule

$$S(a,b) = \frac{h}{3}[f(a) + 4f(a+h) + f(b)],$$

where the step size $h = (b-a)/2$.

- The error is

$$E(f) = \int_a^b f(x)dx - S(a,b) = -\frac{h^5}{90}f^{(4)}(\xi), \qquad \text{for some } \xi \in (a,b).$$

- The error term contains $f^{(4)}(\xi)$, which is not known a priori. We will derive some alternative formula to estimate the error that does not require $f^{(4)}(\xi)$.

- Consider the composite Simpson's rule with $n = 4$ and the step size $(b-a)/4 = h/2$

$$\begin{aligned}
\int_a^b f(x)dx &= \frac{h/2}{3}\Big[f(a) + 4f(a+\frac{1}{2}h) + f(a+h)\Big] - \frac{(h/2)^5}{90}f^{(4)}(\xi_1) \\
&\quad + \frac{h/2}{3}\Big[f(a+h) + 4f(a+\frac{3}{2}h) + f(b)\Big] - \frac{(h/2)^5}{90}f^{(4)}(\xi_2) \\
&= S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b) - \frac{1}{16}\left(\frac{h^5}{180}f^{(4)}(\xi_1) + \frac{h^5}{180}f^{(4)}(\xi_2)\right) \\
&= S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b) - \left(\frac{1}{16}\right)\frac{h^5}{90}f^{(4)}(\tilde{\xi}),
\end{aligned}$$

where we use the intermediate value theorem in the last step and the value $\tilde{\xi}$ is in (a, b).

- Now we have two error bounds:

$$\int_a^b f(x)dx - \left(S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b) \right) = -\left(\frac{1}{16}\right) \frac{h^5}{90} f^{(4)}(\tilde{\xi}) \quad (4.21)$$

$$\int_a^b f(x)dx - S(a,b) = -\frac{h^5}{90} f^{(4)}(\xi) \quad (4.22)$$

  where ξ and $\tilde{\xi}$ are both in (a, b).
- The error estimation is derived by assuming $f^{(4)}(\xi) \approx f^{(4)}(\tilde{\xi})$. Thus,

$$S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b) - S(a,b) \approx -\frac{15}{16} \cdot \frac{h^5}{90} f^{(4)}(\xi)$$

  or

$$-\frac{h^5}{90} f^{(4)}(\xi) \approx \frac{16}{15} \left(S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b) - S(a,b) \right) \quad (4.23)$$

- That means the error term $f^{(4)}(\xi)$ can be approximated by the summation of some computable terms on the right hand side of (4.23).

- Substituting (4.23) in (4.21) and (4.22), we have

$$\left|\int_a^b f(x)dx - \left(S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b)\right)\right| \approx \frac{1}{15}\left|S(a,b) - \left(S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b)\right)\right|. \tag{4.24}$$

- If we assign the tolerance

$$\left|S(a,b) - \left(S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b)\right)\right| < 15\epsilon, \tag{4.25}$$

  then we expect to have

$$\left|\int_a^b f(x)dx - \left(S(a, \frac{a+b}{2}) + S(\frac{a+b}{2}, b)\right)\right| < \epsilon, \tag{4.26}$$

- We may use the computable quantity (4.25) as an error indicator to approximate the error.

- When the approximations in (4.25) differ by more than 15ϵ, we can apply the Simpson's rule individually to the subintervals $[a, (a+b)/2]$ and $[(a+b)/2, b]$, and let the tolerance on each integral within $\epsilon/2$.
- If the approximation on one of the subintervals fails to be within the tolerance $\epsilon/2$, then that subinterval is itself subdivided, and the procedure is reapplied to the two subintervals to determine if the approximation on each subinterval is accurate to within $\epsilon/4$.
- This halving procedure is continued until each portion is within the required tolerance.

MATLAB Driver: Adaptive Simpson's Method

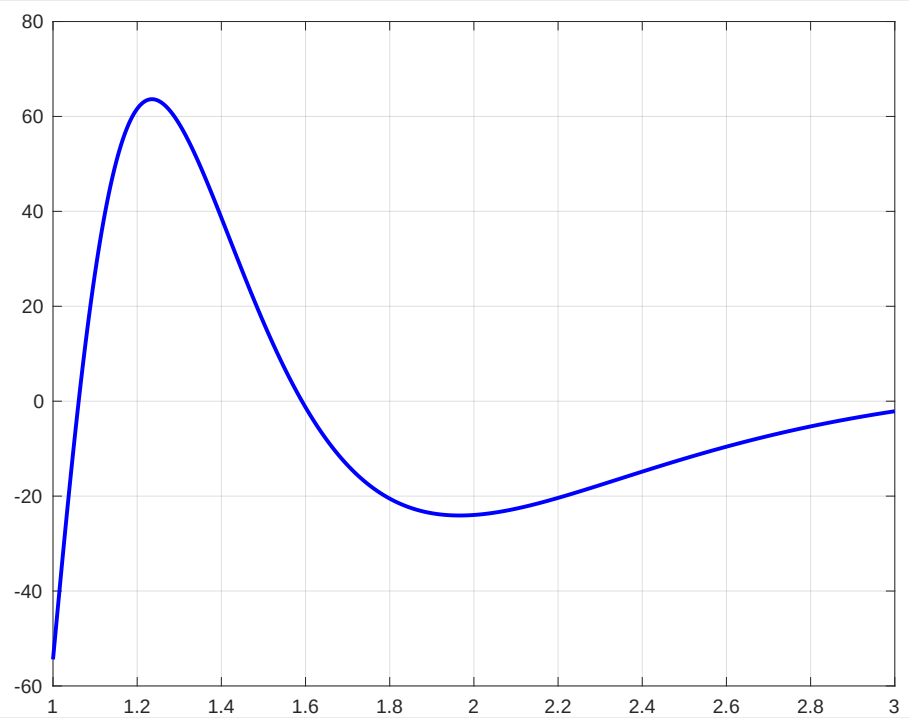
```
function [Q,X] = simpsonAdpt(fun,a,b,tol)
% Adaptive Simpson's Rule
% Inputs
%   fun --- function handle
%   a,b --- integrating interval [a,b]
%   tol --- tolerance
% Outputs
%   Q --- quadrature value
%   X --- nodes generated by adaptive strategy

%%
m = (a+b)/2;
Q = simpson(fun,a,b);
Q1 = simpson(fun,a,m);
Q2 = simpson(fun,m,b);
X = [a,m,b];

if abs(Q1+Q2-Q) > 15*tol
    tol = tol/2;
    [Q1,X1] = simpsonAdpt(fun,a,m,tol);
    [Q2,X2] = simpsonAdpt(fun,m,b,tol);
    Q = Q1+Q2;
    % remove the repeated nodes and sort these nodes
    X = sort(unique([X1,X2]));
end
```

Example 16

Use adaptive Simpson's method with the tolerance 10^{-3} to approximate the integral

$$\int_1^3 \frac{100}{x^2}\sin(\frac{10}{x})dx.$$

Then use the Composite Simpson's rule with same number of nodes to approximate this integral. Compare their results.



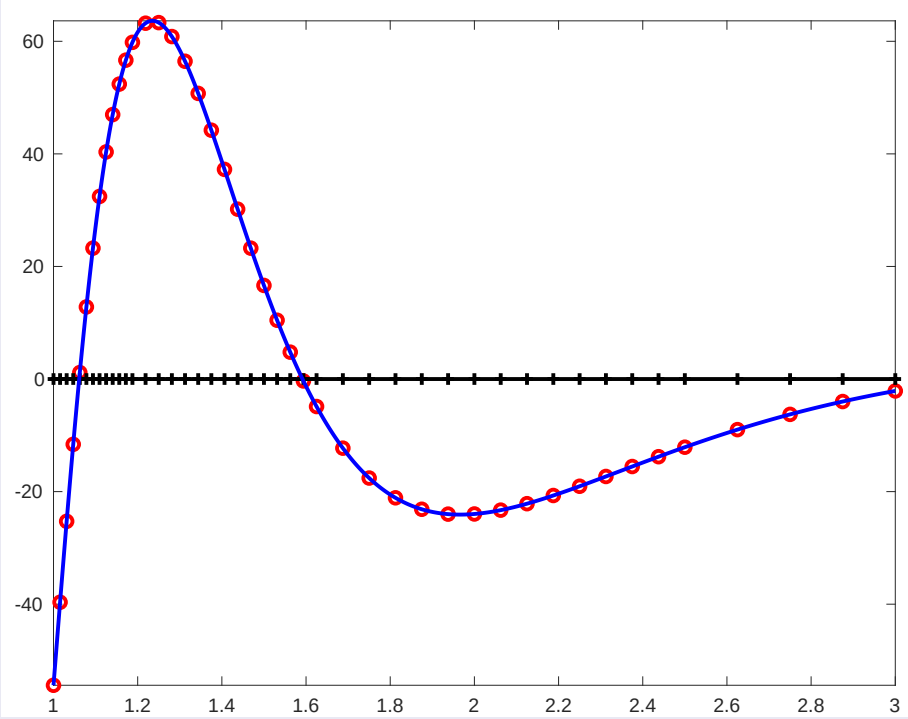
Solution (1/2): MATLAB Driver File

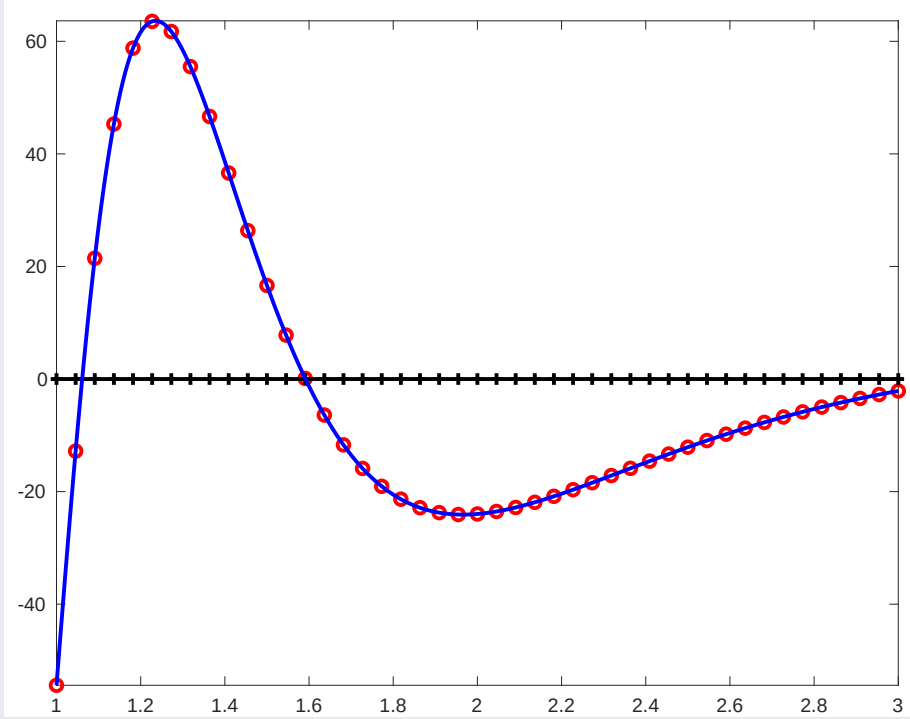
```
clear; clc;
fun = @(x) (100./x.^2).*sin(10./x);
a = 1;
b = 3;
tQ = -1.426024756346266;
%% Adaptive Simpson
tol = 1e-4;
[Q,X] = simpsonAdpt(fun,a,b,tol);
hmin = min(X(2:end)-X(1:end-1));
%% Composite Simpson
n = length(X)-1; % using same number of points as in adaptive method
if mod(n,2) == 1
    n = n+1;
end
Q2 = simpsonComp(fun,a,b,n);

%% Output
disp('-------------------------------------------------------------------')
disp(' pt      hmin       True Integral    Adaptive Simpson    Abs Error')
disp('-------------------------------------------------------------------')
formatSpec = '% 2i   %9.5f    %12.8f     %12.8f       %12.8f\n';
fprintf(formatSpec,[length(X)-1,hmin,tQ, Q,abs(tQ-Q)])

disp(' ')
disp('-------------------------------------------------------------------')
disp(' pt      hmin       True Integral    Composite Simpson   Abs Error')
disp('-------------------------------------------------------------------')
formatSpec = '% 2i   %9.5f    %12.8f     %12.8f       %12.8f\n';
fprintf(formatSpec,[n,(b-a)/(n-1),tQ, Q2,abs(tQ-Q2)])

figure(1); clf
qx = a:0.001:b;
plot(X,fun(X),'ro','lineWidth',2)
hold on
plot(X,zeros(size(X)),'k+-','lineWidth',2)
plot(qx,fun(qx),'b-','lineWidth',2)
hold off; axis tight

figure(2); clf
X2 = a:(b-a)/n:b;
plot(X2,fun(X2),'ro','lineWidth',2)
hold on
plot(X2,zeros(size(X2)),'k+-','lineWidth',2)
plot(qx,fun(qx),'b-','lineWidth',2)
hold off; axis tight
```

Solution (2/2): MATLAB Driver File

pt	hmin	True Integral	Adaptive Simpson	Abs Error
44	0.01562	-1.42602476	-1.42593843	0.00008633

pt	hmin	True Integral	Composite Simpson	Abs Error
44	0.04651	-1.42602476	-1.42327343	0.00275133



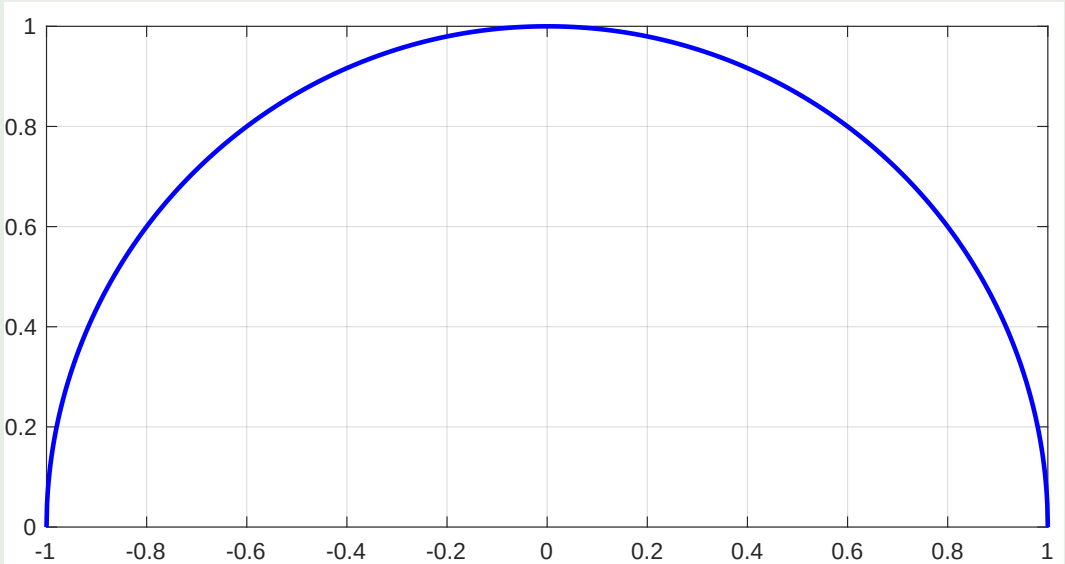


Example 17

In this problem, we use the area of a unit sphere to approximate the value of π. Note that the area of the upper semi-sphere

$$\int_{-1}^{1} \sqrt{1-x^2}dx = \frac{\pi}{2}$$

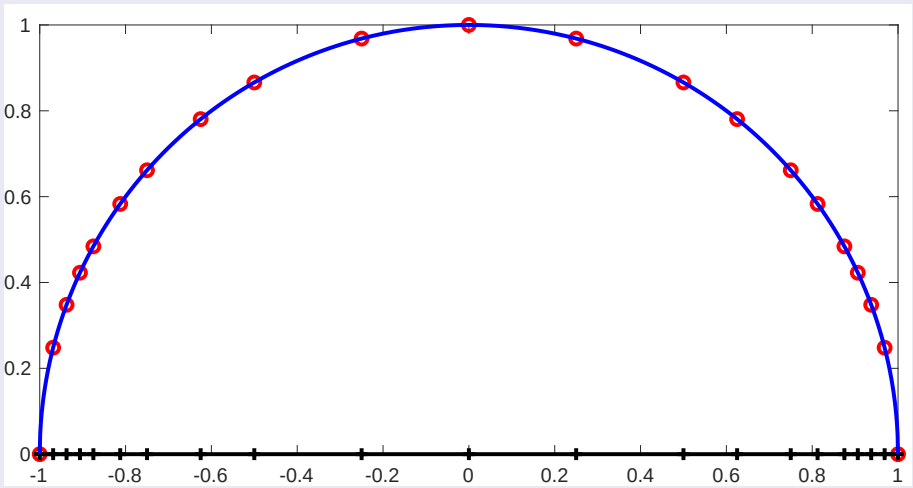
Use the adaptive Simpson's method and the Composite Simpson's method with similar number of nodes to approximate this integral.

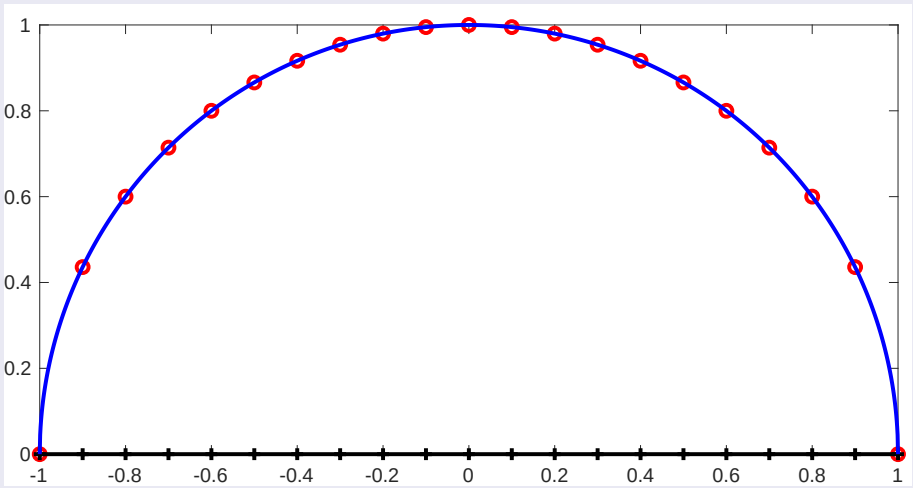


Solution

Set the tolerance to be $tol = 1e-3$. We write a similar driver file as the previous example.

```
-------------------------------------------------------------------
 pt      hmin        True Integral     Adaptive Simpson     Abs Error
-------------------------------------------------------------------
 20     0.03125      1.57079633         1.56937393          0.00142239

-------------------------------------------------------------------
 pt      hmin        True Integral     Composite Simpson    Abs Error
-------------------------------------------------------------------
 20     0.10526      1.57079633         1.56350408          0.00729225
```





In the region the function is steeper, there are more points.

4.5 Gaussian Quadrature

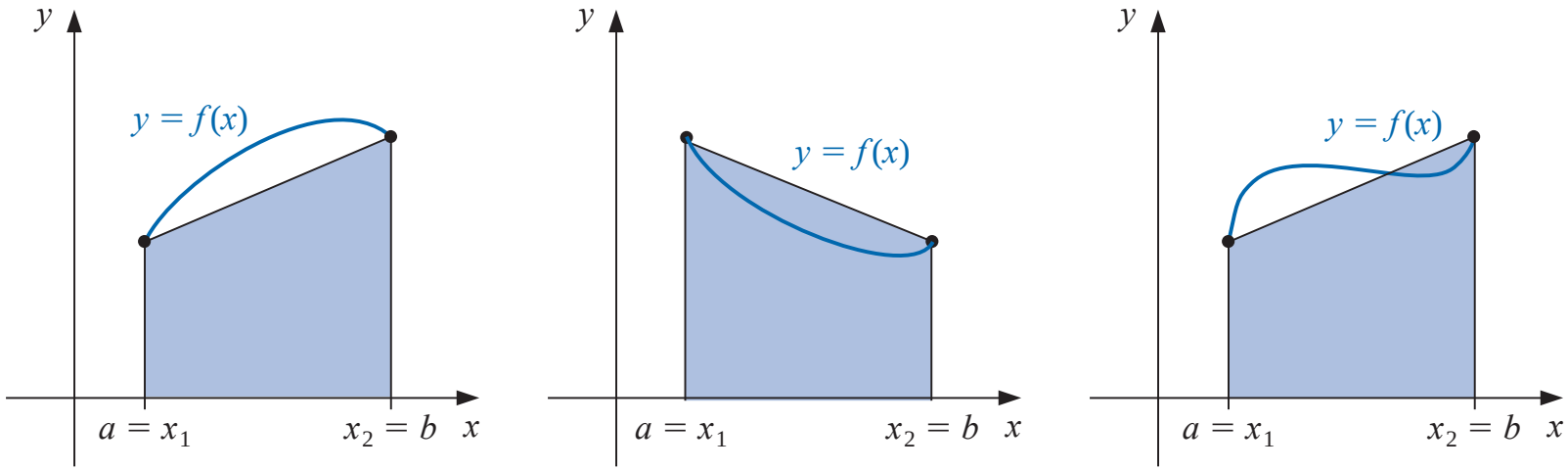
Recall Newton-Cotes Formulas (e.g. Trapezoidal, Simpson's rule):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} a_i f(x_i), \qquad a_i = \int_a^b L_i(x)dx.$$

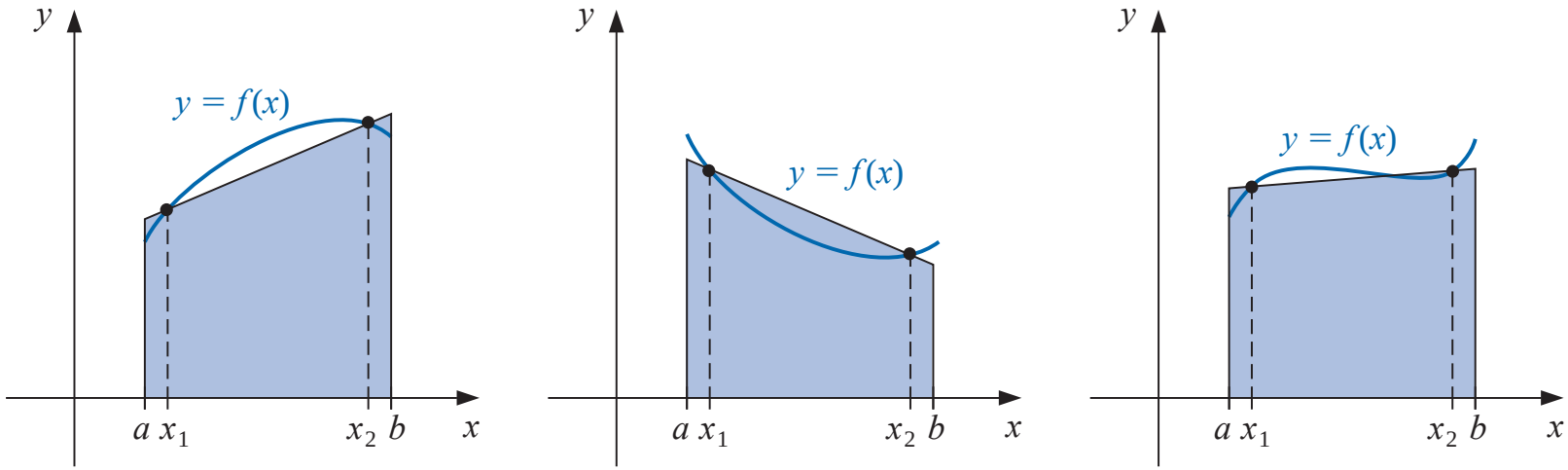
- They use equally-spaced points: $x_i = a + ih$.
- The degree of precision is at most n. i.e., it is exact for polynomial of degree no more than n.

Question: Can we choose the evaluating points in an optimal, rather than an equally-spaced way in order to get higher-degree precision?

- The Trapezoidal rule approximates the integral function by integrating the linear function that joins the endpoints.



- This may not be the best line approximation for the integral. The following choice of the evaluation points is likely to give better approximation:



- In this section, we introduce a numerical quadrature that gives the best approximation of the integral given the number of the quadrature points. It is called the **Gaussian Quadrature**.
- We need to determine a set of nodes $x_1, x_2, \cdots, x_n$ in the interval $[a, b]$, and a set of coefficients (or weights) $c_1, c_2, \cdots, c_n$ such that

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i) \tag{4.27}$$

gives the highest possible degree of precision.

Remark

- There are $2n$ parameters to determine, and the polynomial of degree $2n-1$ also contains $2n$ parameters.
- Thus, we expect the n-point Gaussian quadrature (4.27) to be exact for polynomials of degree $2n-1$.

Example 18

Determine the two-point Gaussian quadrature formula on the interval $[-1, 1]$.

Solution (1/2)

- We need to determine two weights c_1, c_2 and two points x_1, x_2 so that

$$\int_{-1}^{1} f(x)dx \approx c_1 f(x_1) + c_2 f(x_2)$$

  is accurate whenever $f(x)$ is a polynomial of degree up to 3.
- Due to the linearity of integral operator, that is

$$\int_{-1}^{1} (a_0 + a_1 x + a_2 x^2 + a_3 x^3)dx$$
$$= a_0 \int_{-1}^{1} 1dx + a_1 \int_{-1}^{1} xdx + a_2 \int_{-1}^{1} x^2 dx + a_3 \int_{-1}^{1} x^3 dx.$$

  It suffices to test for monomials $f(x) = 1$, x, x^2, and x^3.

Solution (2/2)

$$f(x)=1: \quad \int_{-1}^{1} 1dx = 2, \quad c_1 f(x_1) + c_2 f(x_2) = c_1 + c_2 = 2.$$

$$f(x)=x: \quad \int_{-1}^{1} x dx = 0, \quad c_1 f(x_1) + c_2 f(x_2) = c_1 x_1 + c_2 x_2 = 0.$$

$$f(x)=x^2: \quad \int_{-1}^{1} x^2 dx = \frac{2}{3}, \quad c_1 f(x_1) + c_2 f(x_2) = c_1 x_1^2 + c_2 x_2^2 = \frac{2}{3}.$$

$$f(x)=x^3: \quad \int_{-1}^{1} x^3 dx = 0, \quad c_1 f(x_1) + c_2 f(x_2) = c_1 x_1^3 + c_2 x_2^3 = 0.$$

That is

$$\begin{cases} c_1 + c_2 = 2 \\ c_1 x_1 + c_2 x_2 = 0 \\ c_1 x_1^2 + c_2 x_2^2 = 2/3 \\ c_1 x_1^3 + c_2 x_2^3 = 0 \end{cases} \Longrightarrow \begin{cases} c_1 = 1, \\ c_2 = 1, \\ x_1 = -\sqrt{3}/3, \\ x_2 = \sqrt{3}/3 \end{cases}$$

The 2-pt Gaussian Quadrature is

$$\int_{-1}^{1} f(x)dx \approx f(-\sqrt{3}/3) + f(-\sqrt{3}/3).$$

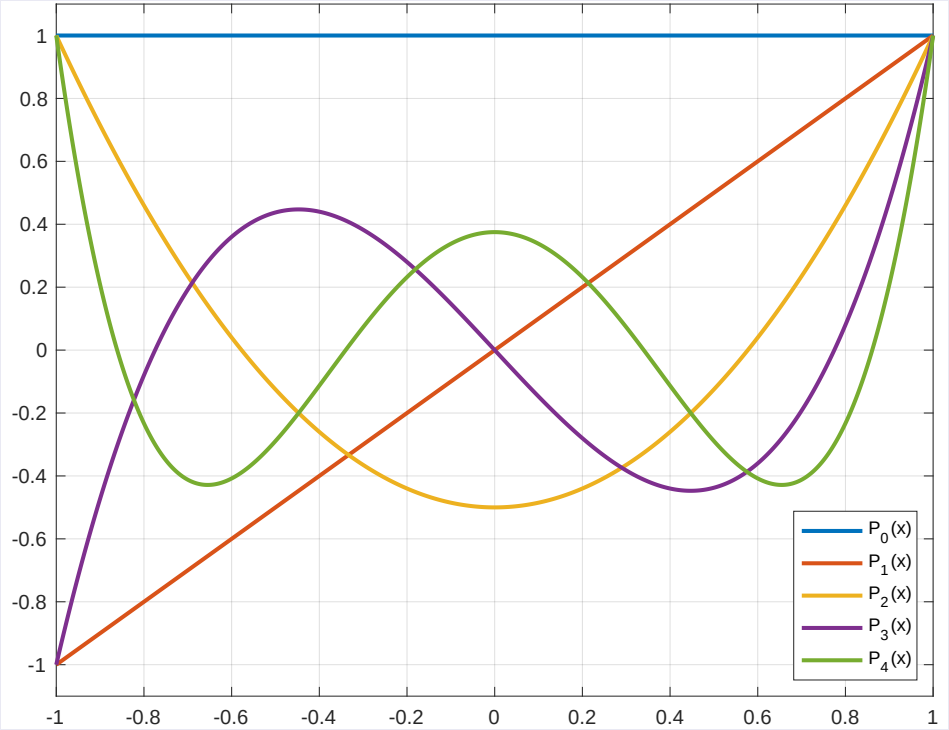
- In general, to determine the Gaussian nodes and weights for higher degrees, we introduce a family of **orthogonal polynomials**, called **Legendre polynomials** $\{P_n\}$.
- The first few Legendre polynomials are

$$
\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \frac{1}{2}(3x^2 - 1) \\
P_3(x) &= \frac{1}{2}(5x^3 - 3x) \\
P_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3)
\end{aligned}
$$

- In general, the Legendre polynomial can be obtained using the **recursive formula**

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \qquad n \geq 1.$$

Illustration of Legendre Polynomials



- $P_n(x)$ has n simple roots $x_1, x_2, \cdots, x_n$, and they all lie in $(-1, 1)$. **These roots are the nodes of the n-point Gaussian quadrature formula.**
- **Orthogonality**

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \frac{2}{2n+1}\delta_{mn}.$$

Theorem 19 (Gaussian Quadrature Weights)

Suppose that $x_1, x_2, \cdots, x_n$ *are the roots of the* n*-th Legendre polynomial* $P_n(x)$*, and that for each* $i = 1, 2, \cdots, n$*, the number* w_i *is defined by*

$$w_i = \int_{-1}^{1} \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} dx. \tag{4.28}$$

If $P(x)$ *is any polynomial of degree less than* $2n$*, then*

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

- The proof of this theorem is on Page 231-232 on the text book.
- While the weight w_i for the quadrature rule can be generated from (4.28), but both the weights w_i and the roots x_i of the Legendre polynomials are extensively tabulated.

Nodes and Weights of Gaussian Quadratures on $[-1, 1]$

Number of points, n	Points, x_i	Approximately, x_i	Weights, w_i	Approximately, w_i
1	0	0	2	2
2	$\pm\frac{1}{\sqrt{3}}$	±0.57735	1	1
3	0	0	$\frac{8}{9}$	0.888889
	$\pm\sqrt{\frac{3}{5}}$	±0.774597	$\frac{5}{9}$	0.555556
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.339981	$\frac{18+\sqrt{30}}{36}$	0.652145
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.861136	$\frac{18-\sqrt{30}}{36}$	0.347855
5	0	0	$\frac{128}{225}$	0.568889
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	±0.538469	$\frac{322+13\sqrt{70}}{900}$	0.478629
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	±0.90618	$\frac{322-13\sqrt{70}}{900}$	0.236927

Nodes and Weights for higher order (up to $n = 64$) are available at

https://pomax.github.io/bezierinfo/legendre-gauss.html

Example 20

Approximate $\int_{-1}^{1} e^x \cos(x)dx$ using Gaussian quadrature with $n = 3$.

Solution

- Extract the value for $n = 3$ from the Gaussian quadrature table.

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9}f(-\sqrt{3/5}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{3/5})$$

- Evaluating $f(x) = e^x \cos(x)$ we have

$$\begin{aligned}\int_{-1}^{1} e^x \cos(x)dx &\approx \frac{5}{9}e^{-\sqrt{\frac{3}{5}}}\cos\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}e^0\cos(0) + \frac{5}{9}e^{\sqrt{\frac{3}{5}}}\cos\left(\sqrt{\frac{3}{5}}\right)\\ &= 1.9333904.\end{aligned}$$

- The exact integration is 1.9334215. The absolute error is 3.1×10^{-5}.

Remark

- Gaussian quadrature with 3 points

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9}f(-\sqrt{3/5}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{3/5})$$

  The absolute error is 3.1×10^{-5}.
- The Simpson's Rule (also 3-point) gives

$$S = \frac{1}{3}[f(-1) + 4f(0) + f(1)] \approx 1.8891533.$$

  The absolute error is 4.4×10^{-2}.
- The composite Simpson's Rule with 13 points gives

$$S_{12} \approx 1.93338827$$

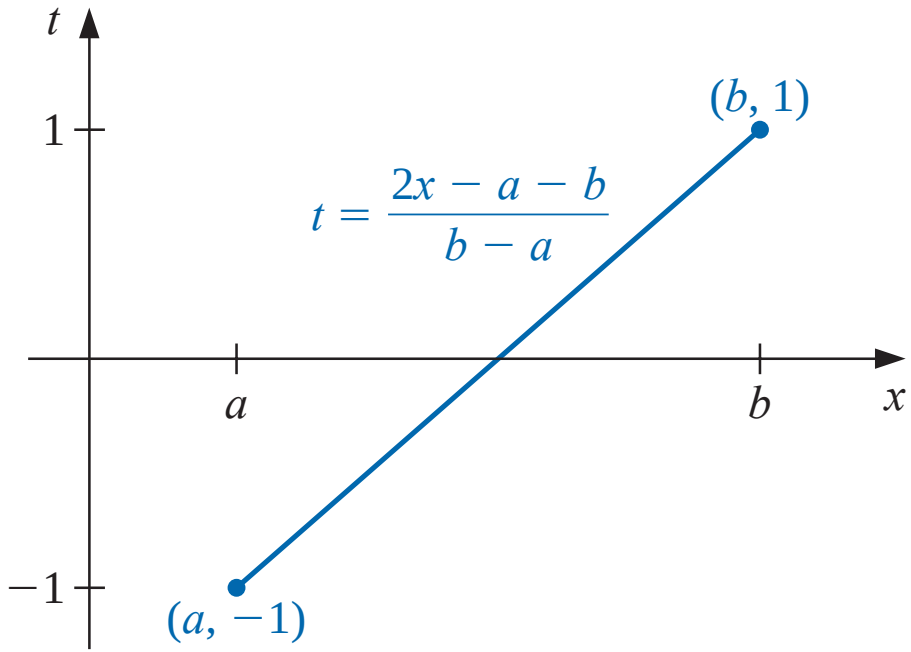
  the error is 3.3×10^{-5}.
- The adaptive Simpsons' Rule also requires at least 15 points to get a more accurate approximation (error $= 2.5 \times 10^{-5}$)

Gaussian Quadrature on Arbitrary Intervals

- Use the mapping

$$t = \frac{2x - a - b}{b - a} \quad \Longleftrightarrow \quad x = \frac{1}{2}[(b - a)t + a + b]$$

we can map the arbitary interval $[a, b]$ to the reference interval $[-1, 1]$.



- Using the mapping $x = \frac{1}{2}[(b-a)t + a + b]$, we have

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\frac{b-a}{2}dt.$$

- Using the Gaussian quadrature on the reference interval $[-1, 1]$ we obtain

Gaussian Quadrature on $[a, b]$

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i),$$

where

- the nodes: $x_i = \frac{b-a}{2}t_i + \frac{a+b}{2}$, and
- the weight: $c_i = \frac{b-a}{2}w_i$
- t_i and w_i are nodes and weights on the reference interval listed on page 86.

Theorem 21 (Error Estimate for Gaussian Quadrature)

Let $f \in C^{2n}[a,b]$, the Gaussian quadrature has the error estimate

$$E = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b \prod_{i=1}^{n} (x - x_i)^2 dx$$

where ξ is a number between a and b.

Remark

- The degree of precision of the n-point Gaussian quadrature is $2n - 1$.

Example 22

Consider the integral $\int_1^3 x^6 - x^2 \sin(2x) dx = 317.3442466$. Compare the n-point Gaussian quadrature formula where $n = 1, 2, 3, 4, 5$.

Solution. (1/2)

- Note that $a = 1$, $b = 3$, $f(x) = x^6 - x^2 \sin(2x)$.
- Let $n = 1$. $t_1 = 0$, $w_1 = 2$, $c_1 = \frac{3-1}{2} w_1 = 2$, and $x_1 = \frac{3-1}{2} t_1 + \frac{3+1}{2} = 2$.

$$\int_1^3 f(x)dx \approx c_1 f(x_1) = 2f(2) = 134.0544. \qquad \text{abs. err} = 183.2898$$

- Let $n = 2$. $t_{1,2} = \pm \frac{1}{\sqrt{3}}$, $w_{1,2} = 1$, $c_{1,2} = 2$, and $x_{1,2} = 2 \pm \sqrt{1/3}$.

$$\int_1^3 f(x)dx \approx c_1 f(x_1) + c_2 f(x_2) = 306.8199. \qquad \text{abs. err} = 10.5243$$

Solution. (2/2)

- For $n = 3, 4, 5$, we have

$$\int_1^3 f(x)dx \approx \sum_{i=1}^{3} c_i f(x_i) = 317.2642. \qquad \text{abs. err} = 0.0801$$

$$\int_1^3 f(x)dx \approx \sum_{i=1}^{4} c_i f(x_i) = 317.3454. \qquad \text{abs. err} = 0.0011$$

$$\int_1^3 f(x)dx \approx \sum_{i=1}^{5} c_i f(x_i) = 317.3442. \qquad \text{abs. err} = 1.99 \times 10^{-5}.$$

MATLAB File for Gaussian Quadrature

```
function y = gaussQuad(fun,a,b,n)

%% On reference interval [-1,1]
if n == 1
    t = 0; w = 2;
elseif n == 2
    t = [-sqrt(3)/3,sqrt(3)/3];
    w = [1,1];
elseif n == 3
    t = [-sqrt(3/5),0,sqrt(3/5)];
    w = [5/9,8/9,5/9];
elseif n == 4
    t1 = sqrt(3/7-2/7*sqrt(6/5));
    t2 = sqrt(3/7+2/7*sqrt(6/5));
    t = [-t1,t1,-t2,t2];
    c = (18+sqrt(30))/36; d = (18-sqrt(30))/36;
    w = [c,c,d,d];
elseif n == 5
    t1 = 1/3*sqrt(5-2*sqrt(10/7));
    t2 = 1/3*sqrt(5+2*sqrt(10/7));
    t = [0,-t1,t1,-t2,t2];
    c = (322+13*sqrt(70))/900; d = (322-13*sqrt(70))/900;
    w = [128/225,c,c,d,d];
end

%% Map to the actual interval
X = (b-a)/2*t+(a+b)/2;
C = (b-a)/2*w;

y = 0;
for i = 1:n
    y = y + C(i)*fun(X(i));
end
```

A driver file for Example 22

```
% ex_4_7_2
clc
fun = @(x) x.^2.*(x.^4 - sin(2*x));
a = 1;
b = 3;
int = 317.3442466;

disp('-------------------------------------------------')
disp('               Gaussian Quadrature')
disp('-------------------------------------------------')
disp(' n    true              Gaussian           Error')
disp('-------------------------------------------------')
formatSpec = '%2d    %.7f      % .7f     % 12.7f \n';

for n = 1:5
    Q = gaussQuad(fun,a,b,n);
    fprintf(formatSpec,[n,int,Q,abs(int-Q)])
end
```

MATLAB Output

```
-------------------------------------------------
               Gaussian Quadrature
-------------------------------------------------
 n    true              Gaussian           Error
-------------------------------------------------
 1    317.3442466       134.0544200      183.2898266
 2    317.3442466       306.8199345       10.5243121
 3    317.3442466       317.2641517        0.0800949
 4    317.3442466       317.3453903        0.0011437
 5    317.3442466       317.3442267        0.0000199
>>
```

Example 23

Approximate the circumference of the ellipse $4x^2 + 9y^2 = 36$ using Gaussian quadrature. The true value is 15.86544. There is no elementary formula for the circumference of the an ellipse.

Solution (1/3).

- Due to symmetry, we only approximate the curve length in the first quadrant $x > 0, y > 0$.Then

$$y = \frac{2}{3}\sqrt{9 - x^2}, \quad y'(x) = \frac{2}{3} \cdot \frac{1}{2}(9 - x^2)^{-1/2}(-2x) = \frac{-2x}{3\sqrt{9 - x^2}}$$

- The circumference of the ellipse is

$$l = 4\int_0^3 \sqrt{1 + [y'(x)]^2}dx = 4\int_0^3 \left(1 + \frac{4x^2}{9(9 - x^2)}\right)^{1/2} dx$$

Solution (2/2).

- We use Gaussian quadrature to approximate this integral

  Gauss Quad 1pt = 12.8582010147
  Gauss Quad 2pt = 13.9568784560
  Gauss Quad 3pt = 14.4788351496
  Gauss Quad 4pt = 14.7801510738
  Gauss Quad 5pt = 14.9748548061

- It does not converges obviously. The reason is the the high variation of the function around $x = 3$.
- We can try to manually divide the interval into m subintervals, and apply Gaussian quadrature on each of them.

  Gauss Quad on 2 intervals = 15.2337481743
  Gauss Quad on 4 intervals = 15.4180879236
  Gauss Quad on 8 intervals = 15.5488761931
  Gauss Quad on 16 intervals = 15.6415116260
  Gauss Quad on 32 intervals = 15.7070690191

- Still not converges very obviously.

Solution (3/3).

- Note that the parametric formula of ellipse $\frac{x^2}{9} + \frac{y^2}{4} = 1$ is

$$x = 3\cos(t), \quad y = 2\sin(t), \quad 0 \le t \le 2\pi$$

- The circumference of the ellipse is

$$l = 4\int_0^{\pi/2} \sqrt{[x'(t)]^2 + [y'(t)]^2}dt = 4\int_0^{\pi/2} \sqrt{9\sin^2(t) + 4\cos^2(t)}dx$$

- Now there is no singular point. Apply Gaussian quadrature then we have
  Gauss Quad 1pt = 16.0190422444
  Gauss Quad 2pt = 15.8297617432
  Gauss Quad 3pt = 15.8679352978
  Gauss Quad 4pt = 15.8654872322
  Gauss Quad 5pt = 15.8654236216
- The convergence pattern is obvious. The relative error for the 5pt quadrature is 1.03×10^{-6}.

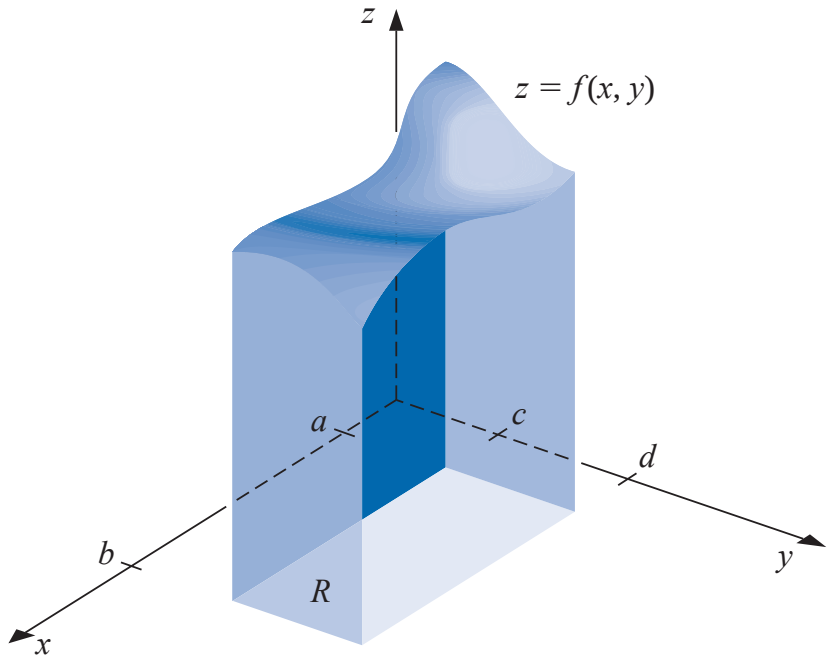
4.6 Multiple Integral

- In this section, we extend the numerical quadratures to multiple integrals. For example, the double integral

$$\iint_R f(x,y)dA$$

where R is some region in $\mathbb{R}^2$.

- First we consider the simple case where $R = [a,b] \times [c,d]$ is a rectangle.



Trapezoidal Rule

- We first consider the Trapezoidal rule. Recall that

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\Big[f(a)+f(b)\Big].$$

- Now we apply it to the multiple integral

$$\begin{aligned}
\iint_R f(x,y)dA &= \int_a^b \left(\int_c^d f(x,y)dy\right)dx \\
&\approx \int_a^b \left(\frac{d-c}{2}\Big[f(x,c)+f(x,d)\Big]\right)dx \\
&= \frac{d-c}{2}\left[\int_a^b f(x,c)dx + \int_a^b f(x,d)dx\right] \\
&\approx \frac{d-c}{2}\left[\frac{b-a}{2}\Big(f(a,c)+f(b,c)\Big) + \frac{b-a}{2}\Big(f(a,d)+f(b,d)\Big)\right] \\
&= \frac{d-c}{2}\frac{b-a}{2}\left[f(a,c)+f(b,c)+f(a,d)+f(b,d)\right].
\end{aligned}$$

Composite Trapezoidal Rule

- Recall that the composite Trapezoidal rule is

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

where $h = (b-a)/m$, $x_i = a + ih$ for each $i = 0, 1 \cdots, m$.

- For multiple integral, we have

$$\iint_R f(x,y)dA = \int_a^b \left(\int_c^d f(x,y)dy\right) dx$$

$$\approx \int_a^b \frac{h_y}{2}\left[f(x,c) + 2\sum_{j=1}^{n-1} f(x,y_j) + f(x,d)\right]dx$$

$$\approx \frac{h_y}{2}\frac{h_x}{2}\left[f(a,c) + 2\sum_{i=1}^{m-1} f(x_i,c) + f(b,c) + f(a,d) + 2\sum_{i=1}^{m-1} f(x_i,d) + f(b,d)\right.$$

$$\left. +2\sum_{j=1}^{n-1}\left(f(a,y_j) + 2\sum_{i=1}^{m-1} f(x_i,y_j) + f(b,y_j)\right)\right],$$

where $h_x = (b-a)/m$, $x_i = a + ih_x$ for each $i = 0, 1, \cdots, m$,
and $h_y = (d-c)/n$, $y_j = c + jh_y$ for each $j = 0, 1, \cdots, n$.

Composite Trapezoidal Rule

$$\int_a^b \left(\int_c^d f(x,y)dy \right) dx \approx \frac{h_y}{2}\frac{h_x}{2}\Big[f(a,c) + f(b,c) + f(a,d) + f(b,d) + 2\sum_{i=1}^{m-1}\big(f(x_i,c) + f(x_i,d)\big) + 2\sum_{j=1}^{n-1}\big(f(a,y_j) + f(b,y_j)\big) + 4\sum_{i=1}^{m-1}\sum_{j=1}^{n-1} f(x_i,y_j)\Big].$$

- $h_x = \dfrac{b-a}{m}$, $h_y = \dfrac{d-c}{n}$.
- $x_i = a + ih_x$, $i = 1, 2, \cdots, m$.
- $y_j = c + jh_y$, $j = 1, 2, \cdots, n$.

MATLAB File for Composite Trapezoidal 2D

```
function Q = trapezoidComp2D(fun,a,b,c,d,m,n)

hx = (b-a)/m;
hy = (d-c)/n;

% 1. Four corners
Q1 = fun(a,c)+fun(b,c)+fun(a,d)+fun(b,d);

% 2. top and bottom boundary nodes
Q2 = 0;
for i = 1:m-1
    xi = a+i*hx;
    Q2 = Q2 + 2*(fun(xi,c) + fun(xi,d));
end

% 3. left and right boundary nodes
Q3 = 0;
for j = 1:n-1
    yj = c+j*hy;
    Q3 = Q3 + 2*(fun(a,yj) + fun(b,yj));
end

% 4. internal nodes
Q4 = 0;
for i = 1:m-1
    for j = 1:n-1
        xi = a+i*hx; yj = c+j*hy;
        Q4 = Q4 + 4*fun(xi,yj);
    end
end

Q = (hx/2)*(hy/2)*(Q1+Q2+Q3+Q4);
```

Gaussian Quadrature

- Applying the Gaussian quadrature in each direction, we have

$$\begin{aligned}\iint_R f(x,y)dA &= \int_a^b \left(\int_c^d f(x,y)dy\right)dx \approx \int_a^b \left(\sum_{j=1}^{n}\beta_j f(x,y_j)\right)dx \\ &\approx \sum_{i=1}^{m}\alpha_i\left(\sum_{j=1}^{n}\beta_j f(x_i,y_j)\right) = \sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_i\beta_j f(x_i,y_j).\end{aligned}$$

- m and n are numbers of Gaussian nodes in x and y direction, respectively.
  α_i and β_j are Gaussian weights in x and y direction, respectively.
  x_i and y_j are Gaussian nodes in x and y direction, respectively.
- That is

$$\alpha_i = \frac{b-a}{2}w_i, \quad x_i = \frac{b-a}{2}t_i + \frac{a+b}{2}, \quad i = 1,\cdots,m.$$
$$\beta_j = \frac{d-c}{2}w_j, \quad y_j = \frac{d-c}{2}t_j + \frac{c+d}{2}, \quad j = 1,\cdots,n.$$

MATLAB File for Gaussian Quadrature 2D

```
function Q = gaussQuad2D(fun,a,b,c,d,m,n)
% 1. On reference interval [-1,1]
[w1,t1] = gaussRef(m);
[w2,t2] = gaussRef(n);
% 2. Map to the actual interval
X = (b-a)/2*t1+(a+b)/2; % nodes in x direction
A = (b-a)/2*w1;  % weight in x direction
Y = (d-c)/2*t2+(c+d)/2;  % nodes in y direction
B = (d-c)/2*w2;  % weight in y direction
Q = 0;
for i = 1:m
    for j = 1:n
        Q = Q + A(i)*B(j)*fun(X(i),Y(j));
    end
end

%% Subroutine of Gaussian Quadrature
function [w,t] = gaussRef(n)
if n == 1
    t = 0; w = 2;
elseif n == 2
    t = [-sqrt(3)/3,sqrt(3)/3];
    w = [1,1];
elseif n == 3
    t = [-sqrt(3/5),0,sqrt(3/5)];
    w = [5/9,8/9,5/9];
elseif n == 4
    t1 = sqrt(3/7-2/7*sqrt(6/5));
    t2 = sqrt(3/7+2/7*sqrt(6/5));
    t = [-t1,t1,-t2,t2];
    c = (18+sqrt(30))/36; d = (18-sqrt(30))/36;
    w = [c,c,d,d];
elseif n == 5
    t1 = 1/3*sqrt(5-2*sqrt(10/7));
    t2 = 1/3*sqrt(5+2*sqrt(10/7));
    t = [0,-t1,t1,-t2,t2];
    c = (322+13*sqrt(70))/900; d = (322-13*sqrt(70))/900;
    w = [128/225,c,c,d,d];
end
```

Example 24

Use Composite Trapezoidal rule and Gaussian Quadrature with $m = 2$ and $n = 3$ to approximate

$$\int_{1.4}^{2.0} \int_{1.0}^{1.5} \log(x + 2y) dy dx$$

The true integral is 0.4295545265.

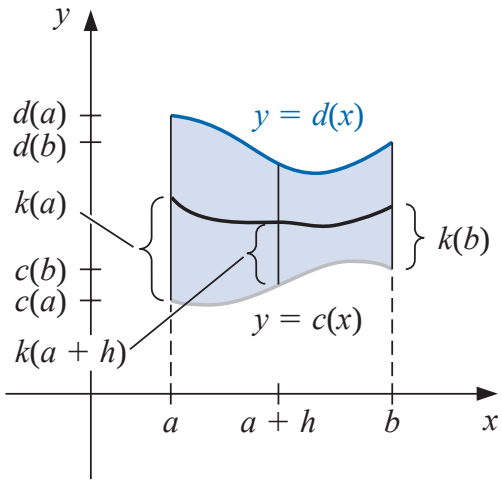
Solution.

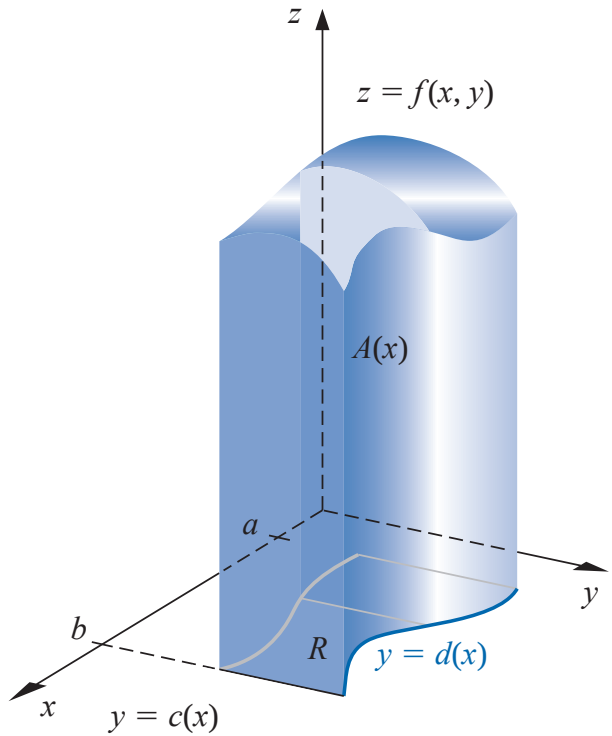
```
% ex_4_8_1
clc
fun = @(x,y) log(x+2*y);
a = 1.4; b = 2; c = 1.0; d = 1.5;
m = 2; n = 3;

Q1 = trapezoidComp2D(fun,a,b,c,d,m,n);
Q2 = gaussQuad2D(fun,a,b,c,d,m,n);

disp('------------------------------')
disp(['True  Integral  = ', num2str(Q, '%12.10f')])
disp(['Trapezoid Quad  = ', num2str(Q1,'%12.10f')])
disp(['Gaussian  Quad  = ', num2str(Q2,'%12.10f')])
disp(' ');disp('-------- Error ----------------')
disp(['Error Trapezoid = ', num2str(abs(Q-Q1),'%12.10f')])
disp(['Error  Gaussian = ', num2str(abs(Q-Q2),'%12.10f')])
```

```
------------------------------
True  Integral  = 0.4295545265
Trapezoid Quad  = 0.4292636231
Gaussian  Quad  = 0.4295547137

-------- Error ----------------
Error Trapezoid = 0.0002909034
Error  Gaussian = 0.0000001872
```

Nonrectangular Regions

- Let $R = \{(x, y) : a \le x \le b,\ c(x) \le y \le d(x)\}$ be a nonrectangular region





- On this region, the double integral

$$\iint_R f(x, y)dA = \int_a^b \left(\int_{c(x)}^{d(x)} f(x, y)dy \right) dx$$

Gaussian Quadrature on Nonrectangular Regions

- We consider the Gaussian quadrature

$$\iint_R f(x,y)dA = \int_a^b \left(\int_{c(x)}^{d(x)} f(x,y)dy \right) dx \approx \int_a^b \left(\sum_{j=1}^{n} \beta_j(x) f(x, y_j(x)) \right) dx$$

where $\beta_j(x) = \dfrac{d(x)-c(x)}{2} w_j$, $y_j(x) = \dfrac{d(x)-c(x)}{2} t_j + \dfrac{c(x)+d(x)}{2}$.

- Continue to apply Gaussian quadrature in x direction, we have

$$\iint_R f(x,y)dA \approx \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j(x_i) f(x_i, y_j(x_i))$$

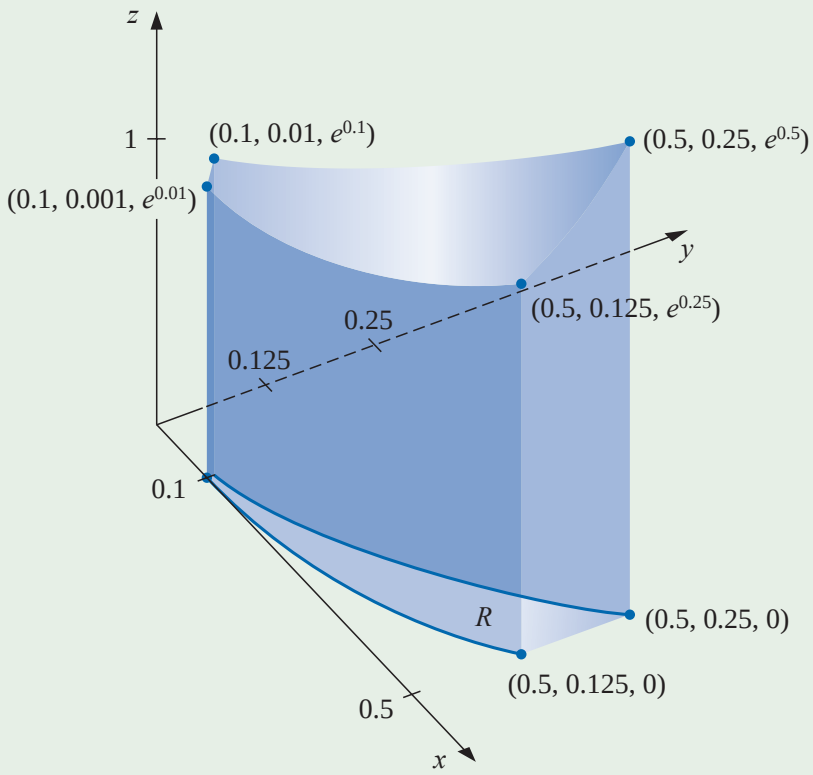
where $\alpha_i = \dfrac{b-a}{2} w_i$, $\quad x_i = \dfrac{b-a}{2} t_i + \dfrac{a+b}{2}$.

Example 25

The volume of the solid on the right is calculated by the integral

$$\int_{0.1}^{0.5} \int_{x^3}^{x^2} e^{y/x} dy dx.$$

Use Gaussian quadrature to approximate this integral.



Solution. (1/2) MATLAB Subroutine for Gaussian Quadrature

```
function z = gaussQuad2DnonrectY(fun,a,b,funC,funD,m,n)

%% On reference interval [-1,1]
[w1,t1] = gaussRef(m);
[w2,t2] = gaussRef(n);

%% Map to the actual interval
X = (b-a)/2*t1+(a+b)/2; % nodes in x direction
A = (b-a)/2*w1;  % weight in x direction

z = 0;
for i = 1:m
    for j = 1:n
        Bji = (funD(X(i))-funC(X(i)))/2*w2(j);
        Yji = (funD(X(i))-funC(X(i)))/2*t2(j)+(funD(X(i))+funC(X(i)))/2;
        z = z + A(i)*Bji.*fun(X(i),Yji);
    end
end
```

Solution. (2/2) MATLAB Driver File

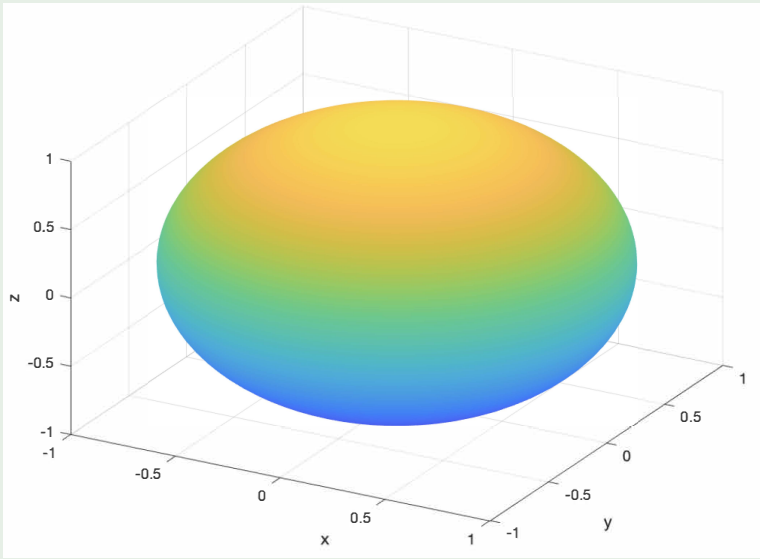
```
% ex_4_8_2
clc
format long
fun = @(x,y) exp(y./x);
funC = @(x) x.^3;
funD = @(x) x.^2;
a = 0.1; b = 0.5;

m = 1; n = 1;
Q1 = gaussQuad2DnonrectY(fun,a,b,funC,funD,m,n);
m = 2; n = 2;
Q2 = gaussQuad2DnonrectY(fun,a,b,funC,funD,m,n);
m = 3; n = 3;
Q3 = gaussQuad2DnonrectY(fun,a,b,funC,funD,m,n);
m = 4; n = 4;
Q4 = gaussQuad2DnonrectY(fun,a,b,funC,funD,m,n);
m = 5; n = 5;
Q5 = gaussQuad2DnonrectY(fun,a,b,funC,funD,m,n);

disp('------------------------------')
disp(['Gauss Quad 1x1  = ', num2str(Q1, '%12.10f')])
disp(['Gauss Quad 2x2  = ', num2str(Q2, '%12.10f')])
disp(['Gauss Quad 3x3  = ', num2str(Q3, '%12.10f')])
disp(['Gauss Quad 4x4  = ', num2str(Q4, '%12.10f')])
disp(['Gauss Quad 5x5  = ', num2str(Q5, '%12.10f')])
```

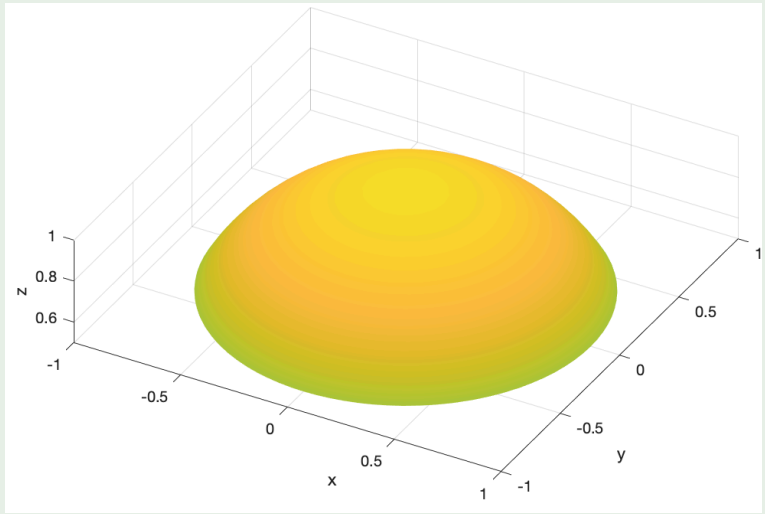
```
------------------------------
Gauss Quad 1x1  = 0.0306258369
Gauss Quad 2x2  = 0.0333453875
Gauss Quad 3x3  = 0.0333058313
Gauss Quad 4x4  = 0.0333055671
Gauss Quad 5x5  = 0.0333055661
```

The Gaussian Quadrature results converge. The volume is approximately 0.0333056.

Example 26

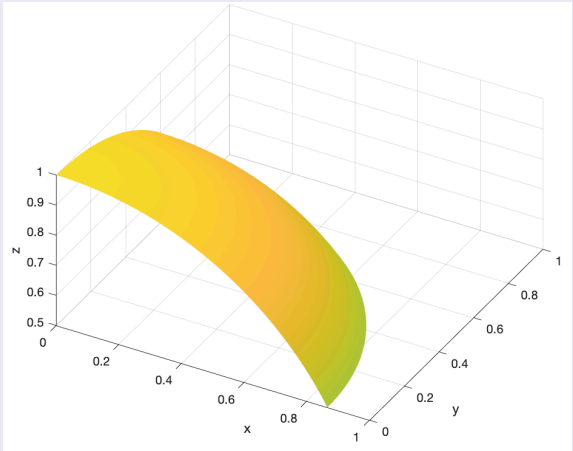
Use Gaussian quadrature to approximate the surface area of the upper part of the unit sphere $x^2 + y^2 + z^2 = 1$ intersecting intersecting with the plane $z = 1/2$





Solution. (1/2)

- We approximate the surface area of the sphere in the first octant



- The surface area formula $z = f(x, y)$ is

$$\iint_R \sqrt{1 + [f_x(x, y)]^2 + [f_y(x, y)]^2} dA$$

- Note that in the first octant $z = f(x, y) = \sqrt{1 - x^2 - y^2}$. The region R is a circle with radius $\sqrt{3/4}$. That is

$$R = \{(x, y) : 0 \le x \le \sqrt{\frac{3}{4}}, \ \ 0 \le y \le \sqrt{\frac{3}{4} - x^2}\}$$

Solution. (2/3)

- The partial derivatives

$$f_x(x,y) = \frac{1}{2}(1-x^2-y^2)^{-1/2}(-2x) = \frac{-x}{\sqrt{1-x^2-y^2}}$$

$$f_y(x,y) = \frac{1}{2}(1-x^2-y^2)^{-1/2}(-2y) = \frac{-y}{\sqrt{1-x^2-y^2}}$$

$$\sqrt{1+f_x^2+f_y^2} = \sqrt{1+\frac{x^2+y^2}{1-x^2-y^2}} = \frac{1}{\sqrt{1-x^2-y^2}}$$

- The surface area formula $z = f(x,y)$ is

$$\iint_R \sqrt{1+f_x^2+f_y^2}\,dA = \int_0^{\sqrt{3/4}} \int_0^{\sqrt{3/4-x^2}} \frac{1}{\sqrt{1-x^2-y^2}}\,dydx$$

Solution. (3/3)

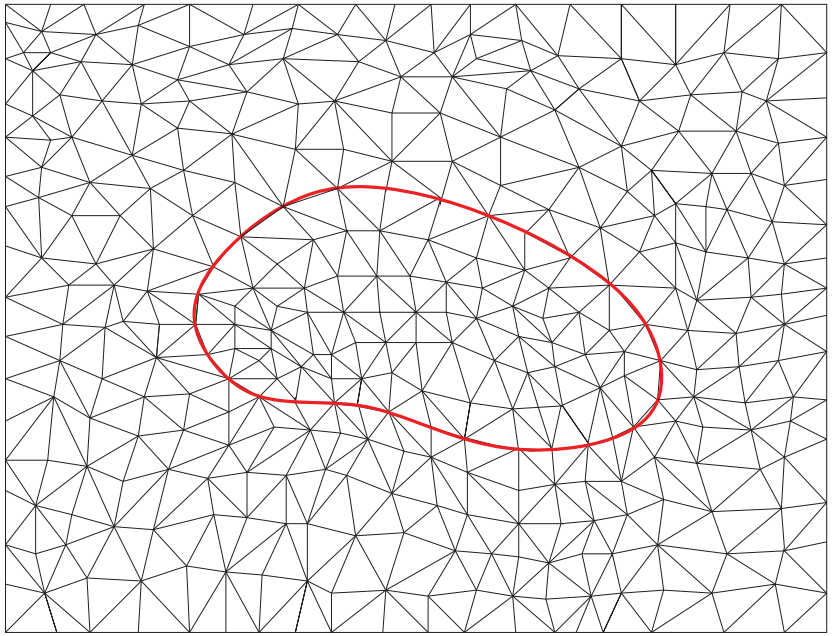
- Using the Gaussian Quadrature, we have

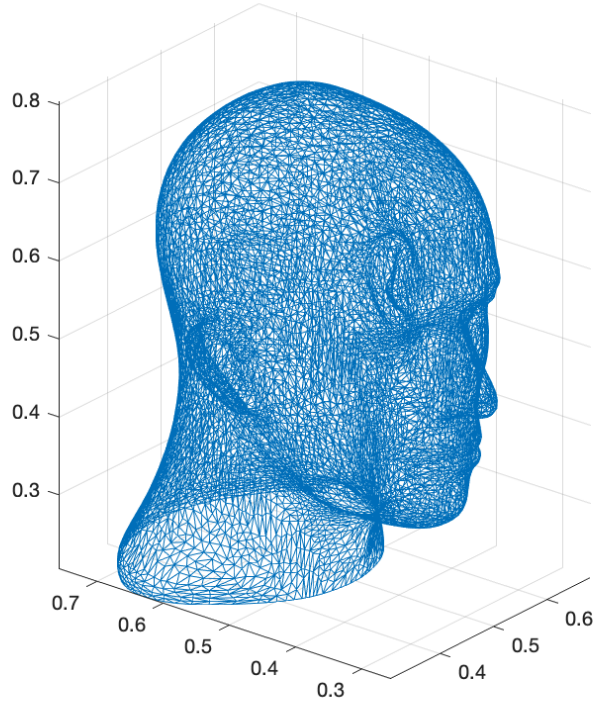
```
------------------------------
Gauss Quad 1x1  = 0.7924058157
Gauss Quad 2x2  = 0.7961419553
Gauss Quad 3x3  = 0.7905556919
Gauss Quad 4x4  = 0.7879893663
Gauss Quad 5x5  = 0.7868151206
Gauss Quad 6x6  = 0.7862434463
Gauss Quad 7x7  = 0.7859405439
Gauss Quad 8x8  = 0.7857667677
Gauss Quad 9x9  = 0.7856601777
```

- The Gaussian quadratures seem to converge. The surface area of the original surface is $4 \times 0.7856601777 = 3.1426407108$.
- The true surface area is π. The absolute error is 0.0010480572.

Concluding Remarks

- The integral over complicated 2D or 3D domains are usually approximated through meshes (triangulations). That is to divide the integral domain into triangles in 2D or tetrahedra in 3D, then apply Gaussian quadrature on these simplicial geometries.





- This technique is frequently used in numerical methods for partial differential equations, such as **finite element method (FEM)**.