SECTION 5: INTEGRATION

ESC 440 – Computational Methods for Engineers

Introduction

Integration

- ***Integration***, or ***quadrature***, has many engineering applications
- A few examples:
  - Mean value

$$\bar{y} = \frac{\int_a^b f(x)dx}{b-a}$$

  - Constitutive physical laws

$$\Delta p = \int F(t)dt$$

$$\Delta v = \frac{1}{C}\int i(t)dt$$

$$\Delta x = \int v(t)dt$$

  - Total flux through a surface

$$Q = \iint U(x,y)dx\,dy$$

  - Etc. …

Numerical Integration

- The numerical integration algorithms we'll look at can be divided into two broad categories:
  - ***Algorithms for integration of data or functions***
    - No flexibility to choose the points, $f(x_i)$, used for calculation of the integral
    - Points, $f(x_i)$, may or may not be uniformly-spaced
    - ***Newton-Cotes formulas***
  - ***Algorithms for the integration of functions***
    - Exploit the ability to calculate $f(x)$ at any value of x
    - Improved accuracy and efficiency
    - ***Adaptive quadrature***, Romberg integration, Gauss quadrature

Newton-Cotes Formulas

This first category of numerical integration algorithms can be applied either to functions or to discrete data sets.

Newton-Cotes Formulas

- Want to approximate the integral of a function or data set

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

- ***Approximate $f(x)$ with something that is easy to integrate***
  - An n^{th}-order polynomial

$$f(x) \approx f_n(x) = a_0 + a_1 x + \cdots + a_n x^n$$

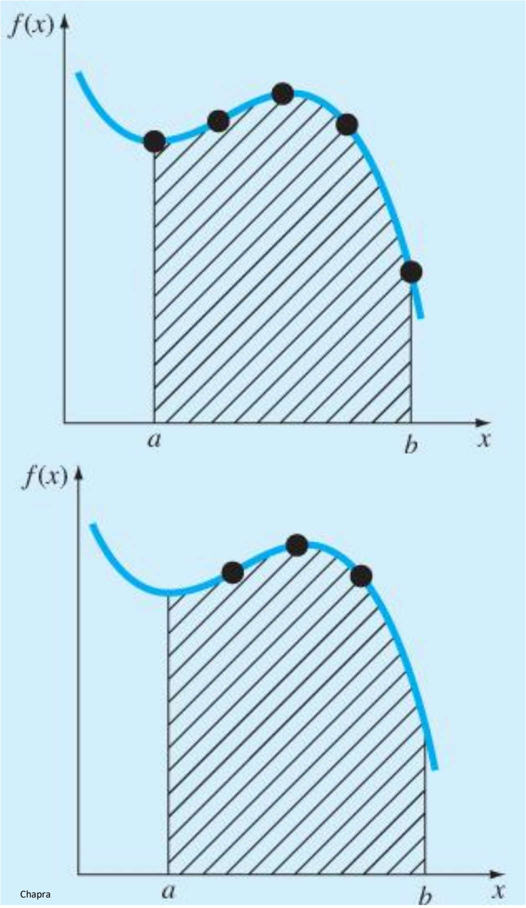
- Integral approximation:

$$\hat{I} = \int_a^b f_n(x)dx \approx I$$

- Unless otherwise noted, Newton-Cotes formulas assume ***evenly-spaced data points***

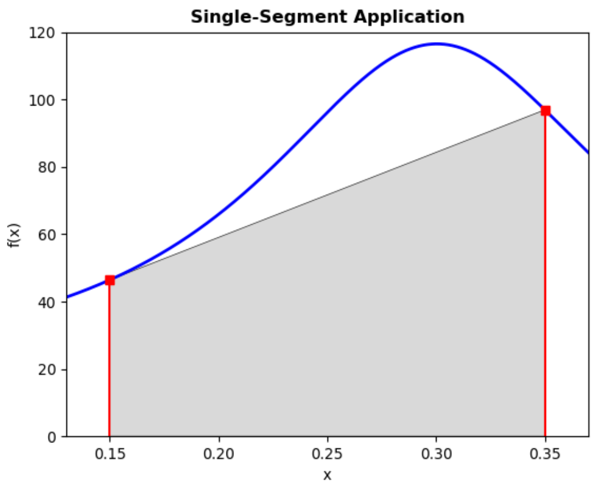
Closed Forms vs. Open Formulas

- Two different versions of the Newton-Cotes integral formulas:
  - ***Closed forms***
    - Values of the function at the limits of integration, $f(a)$ and $f(b)$, are known
  - ***Open forms***
    - $f(a)$ and $f(b)$ are unknown
- We'll focus on closed forms of the Newton-Cotes formulas

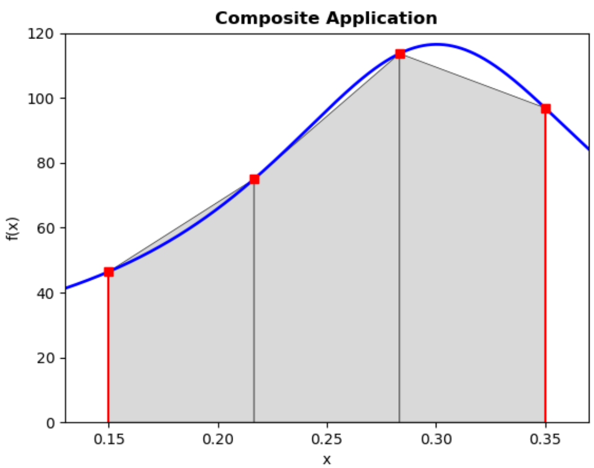


Single-Segment vs. Composite

- Newton-Cotes formulas may be applied in two different ways:
  - ***Single-segment***
    - Entire integration interval, $[a, b]$, approximated with a single polynomial



  - ***Composite***
    - Integration interval divided into multiple segments
    - Integral approximated for each segment – results summed



Trapezoidal Rule

In the following sections, we'll look at three different Newton-Cotes integration formulas:

- Trapezoid rule
- Simpson's 1/3 rule
- Simpson's 3/8 rule

Trapezoidal Rule

- Approximate $f(x)$ as a ***first-order polynomial***

$$f(x) \approx f_1(x) = a_0 + a_1 x$$

$$f_1(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$$

- Integral approximation:

$$\hat{I} = \int_a^b f_1(x)dx = \int_a^b \left[f(a) + \frac{f(b) - f(a)}{b - a}(x - a)\right] dx \approx I$$

- ***Trapezoidal rule formula***:

$$\boxed{\hat{I} = (b - a)\frac{f(a) + f(b)}{2}}$$

Trapezoidal Rule

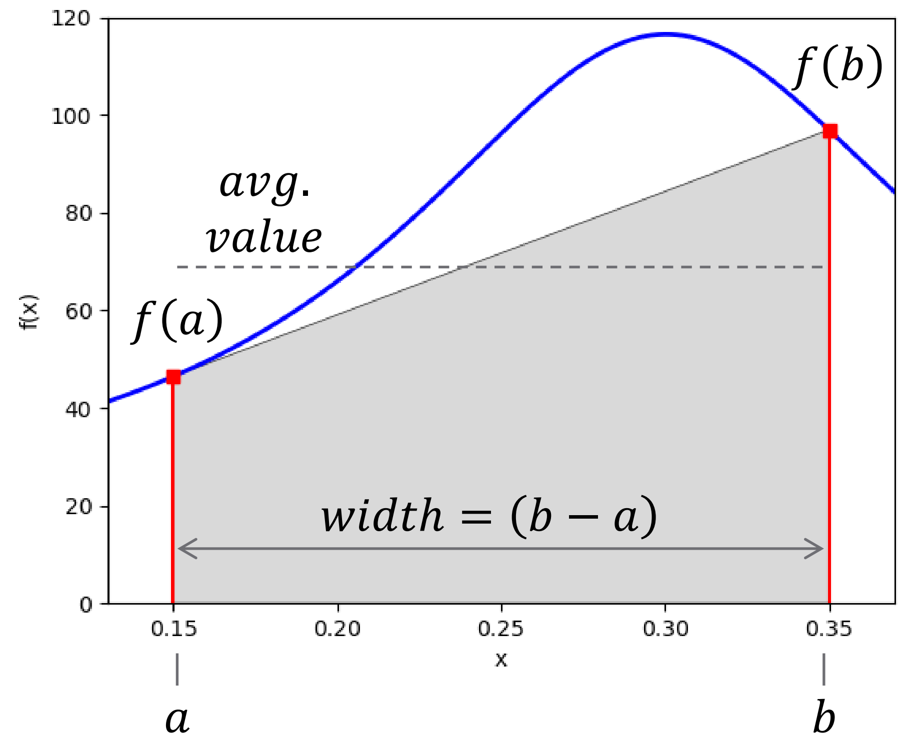
- The trapezoidal rule formula

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

can be interpreted as

$$\hat{I} = (width) \times (avg.value)$$

- All Newton-Cotes formulas can be expressed this way
  - Only the approximation of the average value of $f(x)$ varies
  - More accurate approx. of avg. value yields more accurate integral estimate



- Integral approximation is the area under the polynomial approximation of $f(x)$

Trapezoidal Rule – Error

- The error of the trapezoidal rule estimate is

$$E_t = \hat{I} - I = \frac{1}{12} f''(\xi)(b-a)^3$$

where ξ is some unknown value of x on $[a, b]$

- Since ξ is unknown, approximate the error as

$$E_a = \frac{1}{12} \bar{f}''(b-a)^3$$

where $\bar{f}''$ is the ***mean curvature*** of $f(x)$ on $[a, b]$

Trapezoidal Rule – Error

- The error of the trapezoidal rule estimate is

$$E_t = \frac{1}{12} f''(\xi)(b-a)^3$$

- If the curvature of $f(x)$ is zero on $[a, b]$

$$f''(x) = 0, \text{ for } a \leq x \leq b$$

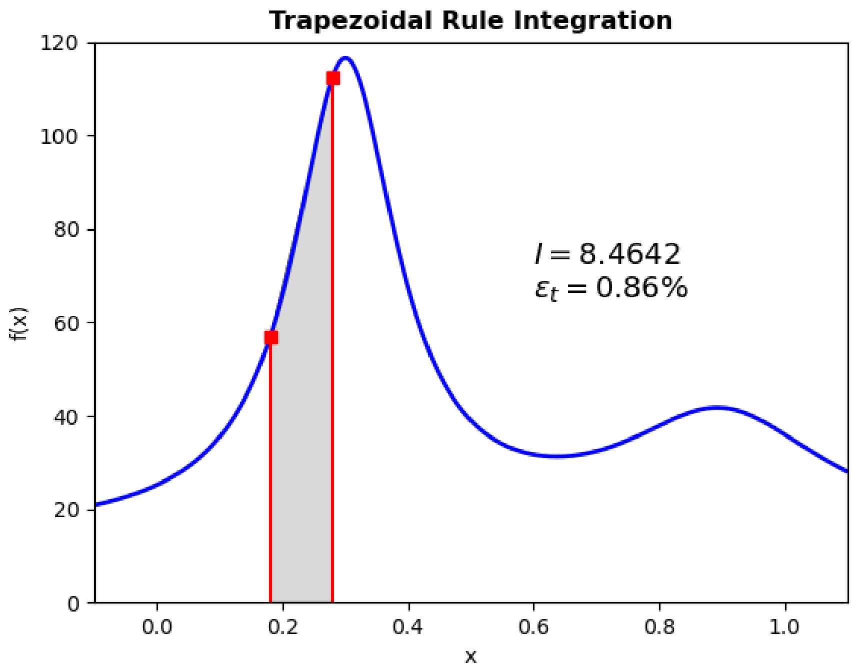
- Then the trapezoidal rule approximation is exact

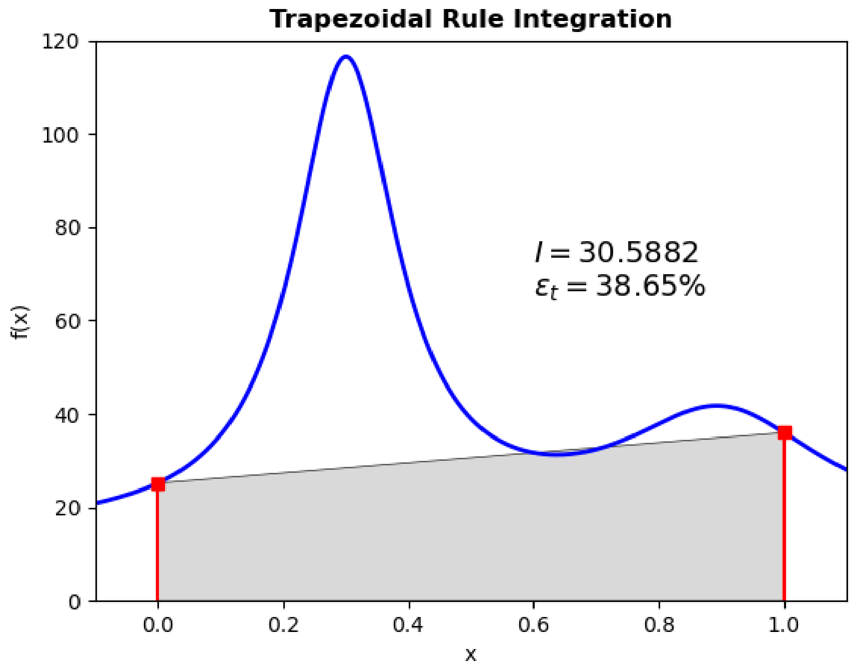
$$E_t = 0$$

- First-order polynomial is an exact representation of a linear $f(x)$

Trapezoidal Rule – Example

- Trapezoidal rule may provide an accurate integral estimate
  - Over regions with low curvature
  - Where $f(x)$ is reasonably approximated as linear





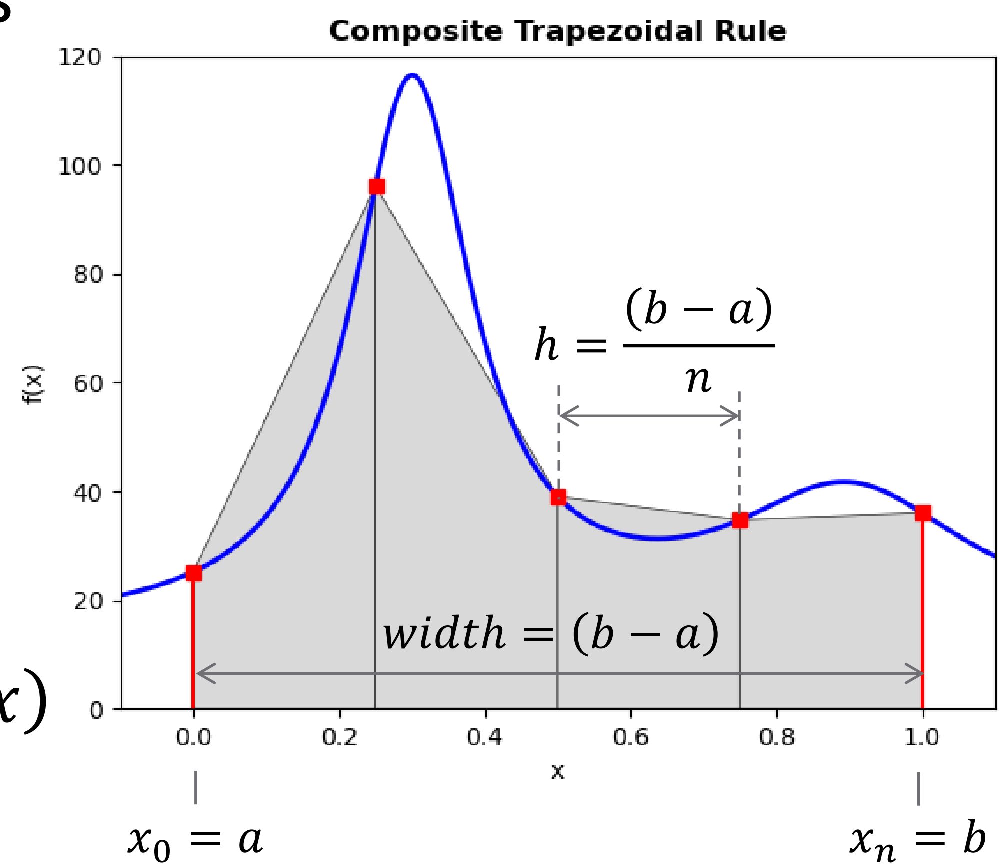
- Or, large errors may result
  - Over regions with large curvature
  - Where a linear approximation is unacceptable

Composite Trapezoidal Rule

- Accuracy can be improved by dividing the interval $[a, b]$ into n segments
  - $n + 1$ evenly-spaced sample points of $f(x)$: $x_0 \dots x_n$
  - Segment width:

$$h = \frac{b - a}{n}$$

- Now approximating $f(x)$ as ***piece-wise linear***



Composite Trapezoidal Rule

- Divide the integral into n segments

$$I = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx$$

- Approximate each term using the trapezoidal rule

$$\hat{I} = h\frac{f(x_0) + f(x_1)}{2} + h\frac{f(x_1) + f(x_2)}{2} + \cdots + h\frac{f(x_{n-1}) + f(x_n)}{2}$$

- Using summation notation

$$\boxed{\hat{I} = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]}$$

- Or, in $(width) \times (avg. value)$ form

$$\hat{I} = (b-a)\frac{\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]}{2n}$$

Composite Trapezoidal Rule – Error

- Total error is the sum of the individual errors

$$E_t = \sum_{i=1}^{n} E_{t,i} = \frac{1}{12} h^3 \sum_{i=1}^{n} f''(\xi_i) = \frac{(b-a)^3}{12n^3} \sum_{i=1}^{n} f''(\xi_i)$$

- Again, approximate using $\bar{f}''$, the ***mean curvature***

$$E_a = \frac{(b-a)^3}{12n^3} \sum_{i=1}^{n} \bar{f}''$$

where

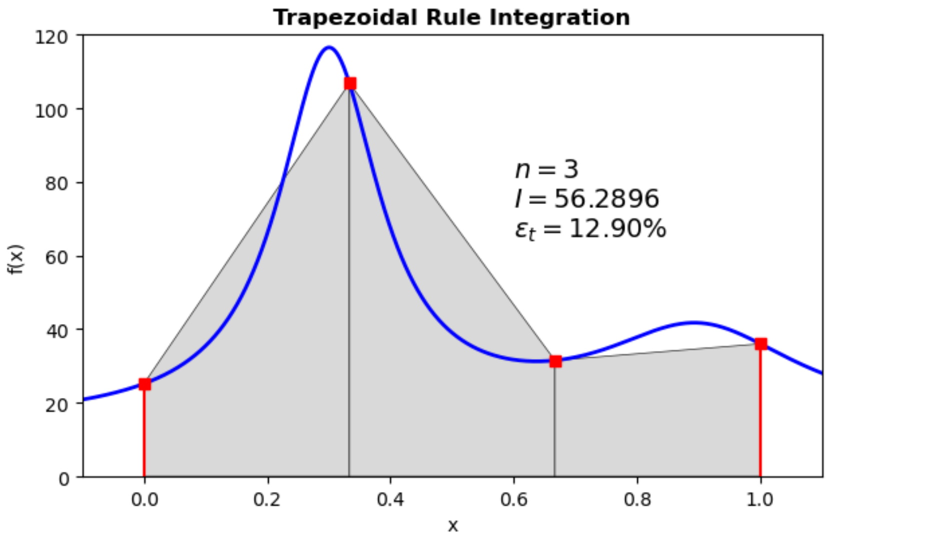
$$\sum_{i=1}^{n} \bar{f}'' = n\bar{f}''$$

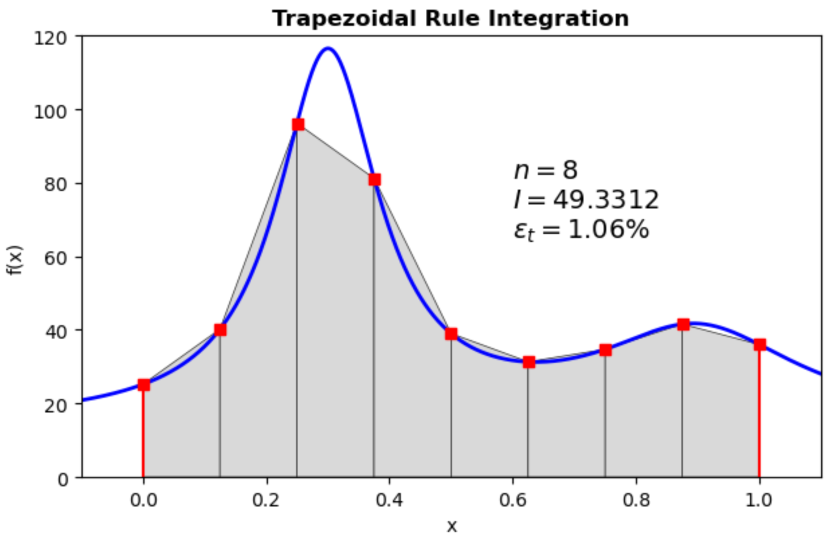
so

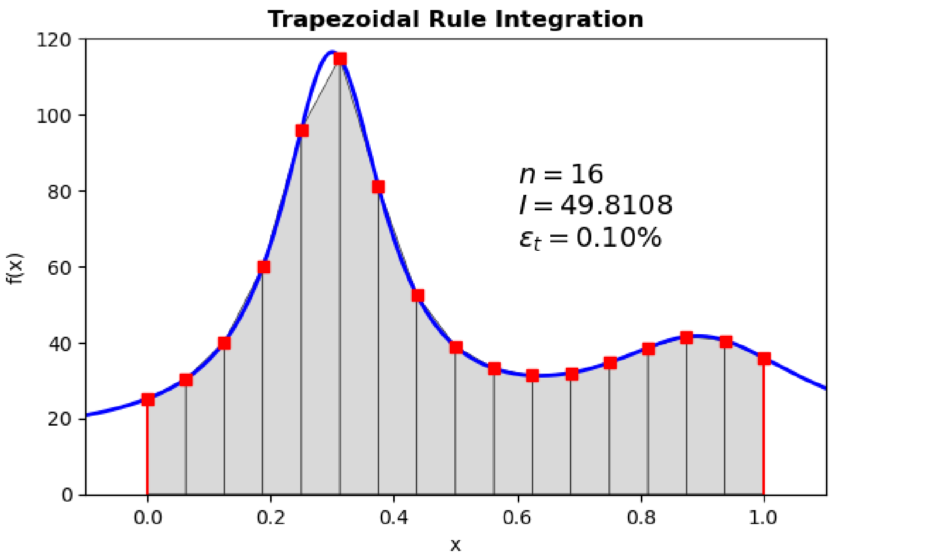
$$\boxed{E_a = \frac{(b-a)^3}{12n^2} \bar{f}''}$$

Composite Trapezoidal Rule – Example

- Accuracy improves as the number of segments increases
  - Average curvature over each segment decreases
  - $f(x)$ better approximated as linear over smaller regions







Trapezoidal Rule – Unequally-Spaced Data

- Trapezoidal rule can be easily modified to accommodate unequally-spaced data points
  - Account for the width of each of the n individual segments explicitly

$$\hat{I} = h_1 \frac{f(x_0) + f(x_1)}{2} + h_2 \frac{f(x_1) + f(x_2)}{2} + \cdots + h_n \frac{f(x_{n-1}) + f(x_n)}{2}$$

- Useful for measured data, where uneven spacing is not uncommon

Trapezoidal Rule in Python – `trapezoid()`

- The integrate module from the SciPy package includes several integration functions, including trapezoid rule
  - Import it first:

```
from scipy import integrate
```

```
I = integrate.trapezoid(y, x)
```

  - `y`: vector of dependent variable data
  - `x`: vector of independent variable data
  - `I`: trapezoidal rule approximation to the integral of `y` with respect to `x` (a scalar)

- Data need not be equally-spaced
  - Segment widths calculated from `x` values

Cumulative Integral – `cumulative_trapezoid()`

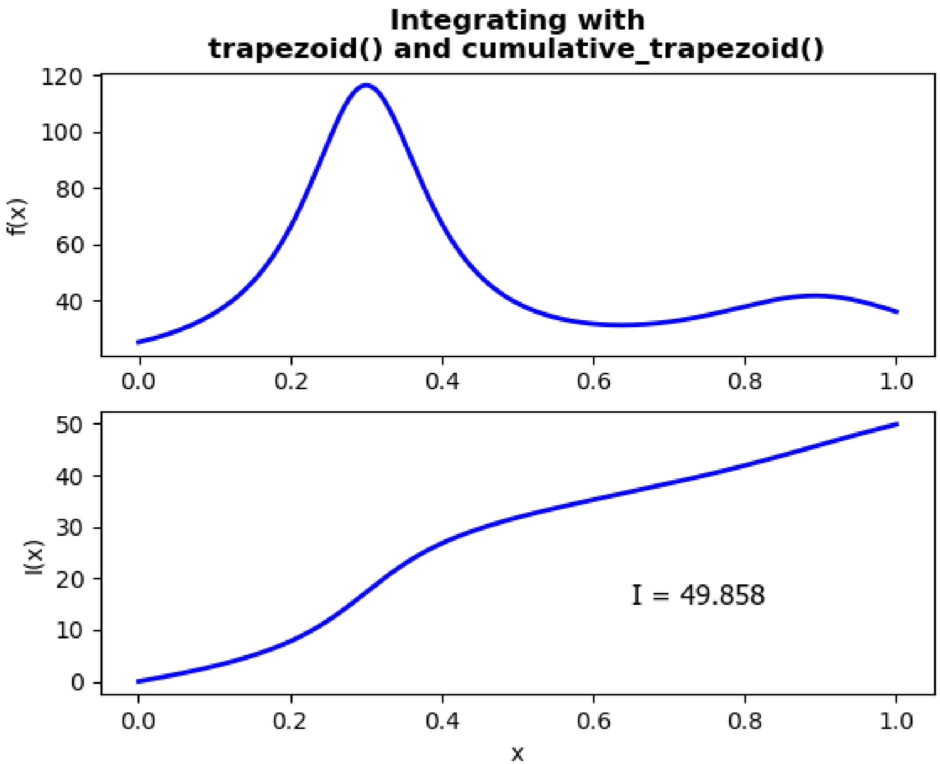

```
I = integrate.cumulative_trapezoid(y, x,
                initial=0)
```

- `y`: n-vector of dependent variable data
- `x`: n-vector of independent variable data
- `initial`: optional initial value inserted as the first value in `I` – if not given, `I` is an (n-1)-vector
- `I`: trapezoidal rule approximation to the ***cumulative integral*** of `y` with respect to `x` (an n-vector)

- Result is a vector – equivalent to:

$$I(x) = \int_{x_1}^{x} y(\tilde{x})\, d\tilde{x}$$

trapezoid() and cumulative_trapezoid()




```python
# trapz_test.py

import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate

x = np.linspace(0,1,2000)
f = lambda x: 1 / ((x-0.3)**2 + .01) + 1 / ((x-0.9)**2 + 0.04) + 14

y = f(x)

I = integrate.trapezoid(y, x)
Ic = integrate.cumulative_trapezoid(y, x)

plt.figure(1); plt.clf()
plt.subplot(211)
plt.plot(x,y,'-b',linewidth=2)
plt.ylabel('f(x)')
plt.title('Integrating with\ntrapezoid() and cumulative_trapezoid()',
    fontweight='bold')

plt.subplot(212)
plt.plot(x[1:],Ic,'-b',linewidth=2)
plt.xlabel('x'); plt.ylabel('I(x)')
plt.text(0.65,15,f'I = {I:1.3f}',
    fontsize=12,fontname='Tahoma')
```

Simpson's 1/3 Rule

Simpson's 1/3 Rule

- Approximate $f(x)$ with a second-order polynomial

$$f(x) \approx f_2(x)$$

where $f_2(x)$ can be expressed as a Lagrange polynomial:

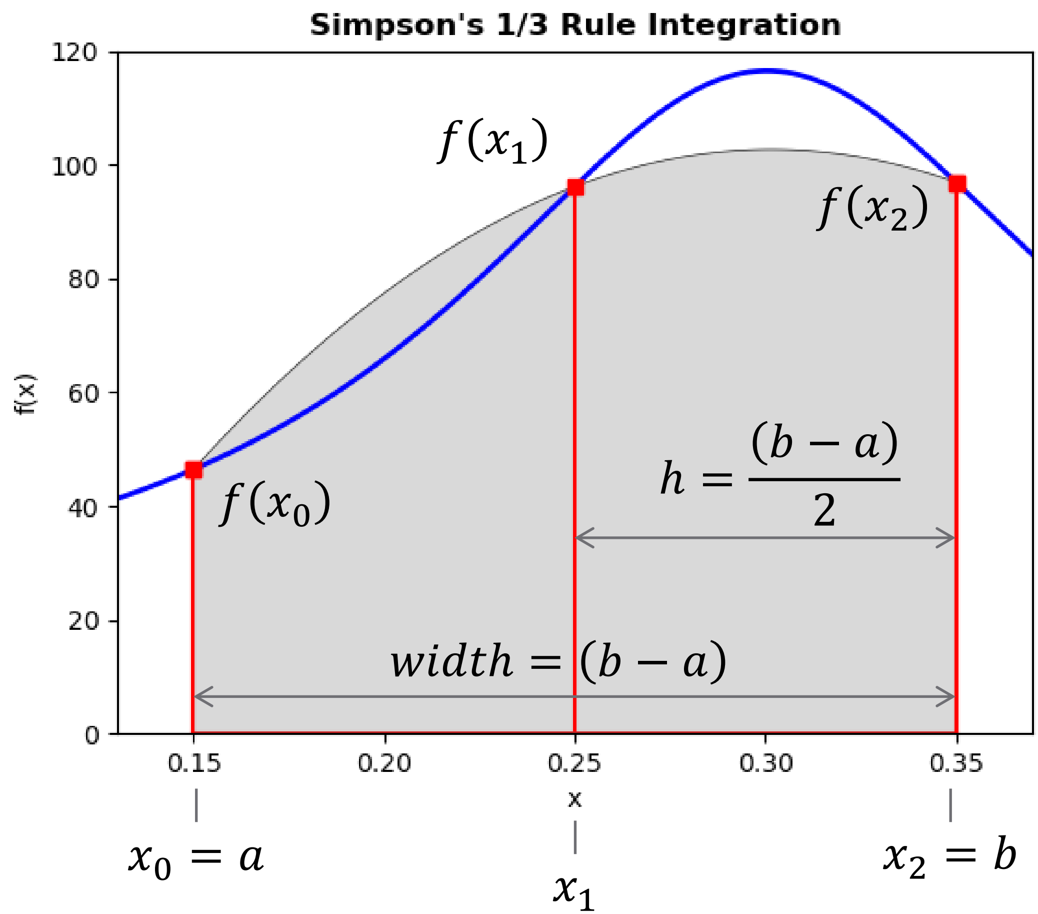
$$f_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$

- Approximate the integral of $f(x)$ as the integral of the quadratic approximation

$$I \approx \hat{I} = \int_a^b f_2(x)\, dx$$

Simpson's 1/3 Rule

- Now fitting a parabola to $f(x)$
- ***Three points*** required: x_0, x_1, and x_2
- Integration interval, $[a, b]$ divided into ***two segments***
- Points must be ***evenly spaced***



Simpson's 1/3 Rule

- Evaluating the integral of the quadratic approximation, $f_2(x)$, yields ***Simpson's 1/3 rule***:

$$\boxed{\hat{I} = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right]}$$

- Or, in $\hat{I} = (width) \times (avg.value)$ form:

$$\boxed{\hat{I} = (b - a)\,\frac{f(x_0) + 4f(x_1) + f(x_2)}{6}}$$

Simpson's 1/3 Rule – Error

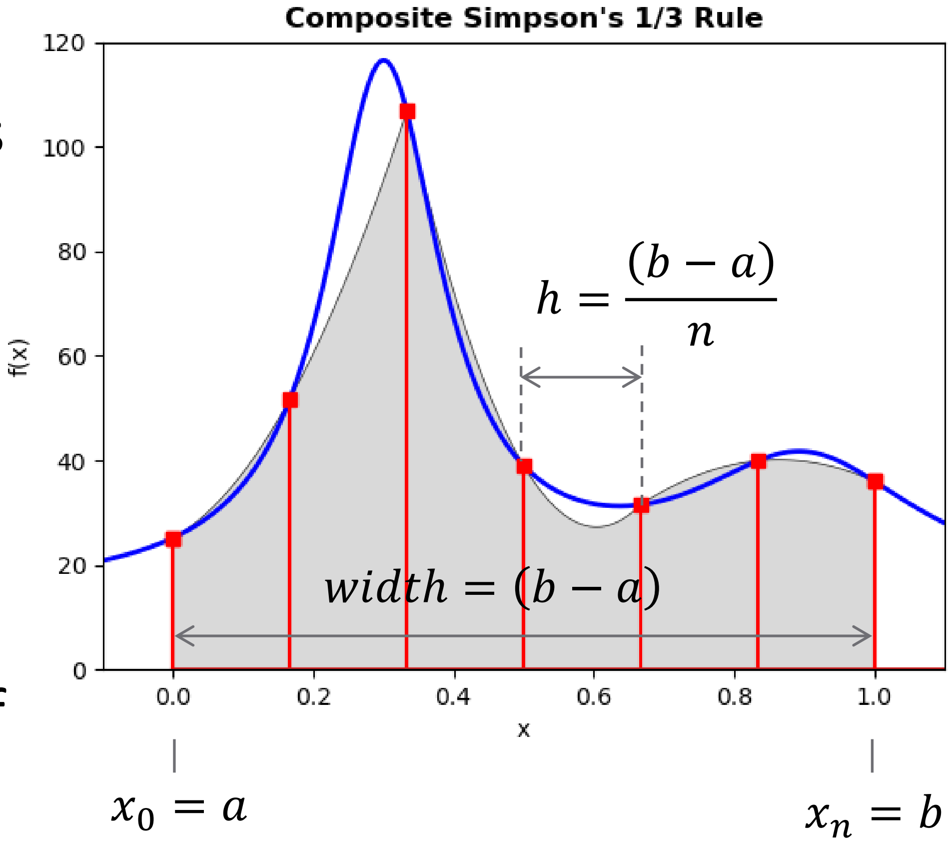
- The error associated with Simpson's 1/3 rule is

$$E_t = \frac{1}{90} h^5 f^{(4)}(\xi) = \frac{(b-a)^5}{2880} f^{(4)}(\xi)$$

- Error is proportional to the fourth derivative of $f(x)$
  - For third- and lower-order polynomials, $f^{(4)} = 0$
  - ***The Simpson's 1/3 rule integral estimate is exact for cubic and lower-order polynomials***
  - An interesting result, given that $f(x)$ is approximated with only a quadratic

Composite Simpson's 1/3 Rule

- Accuracy can be improved by dividing the interval $[a, b]$ into n segments
- Each application of Simpson's 1/3 rule requires three points, and two segments
  - Total number of ***segments must be even***
  - Total number of ***points must be odd***
- $f(x)$ approximated as a ***quadratic*** over each pair of adjacent segments



Composite Simpson's 1/3 Rule

- Divide $[a, b]$ into n segments, and the integral into $n/2$ segments

$$I = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)dx$$

- Approximate each term using Simpson's 1/3 rule

$$\hat{I} = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] + \frac{h}{3}[f(x_2) + 4f(x_3) + f(x_4)] + \cdots + \frac{h}{3}[f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$$

- Using summation notation

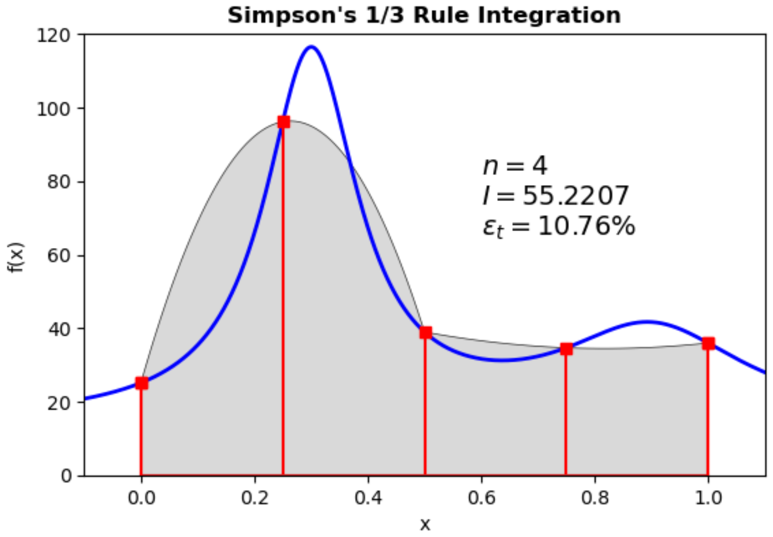
$$\boxed{\hat{I} = \frac{h}{3}\left[f(x_0) + 4\sum_{i=1,3,5\ldots}^{n-1} f(x_i) + 2\sum_{j=2,4,6\ldots}^{n-2} f\left(x_j\right) + f(x_n)\right]}$$

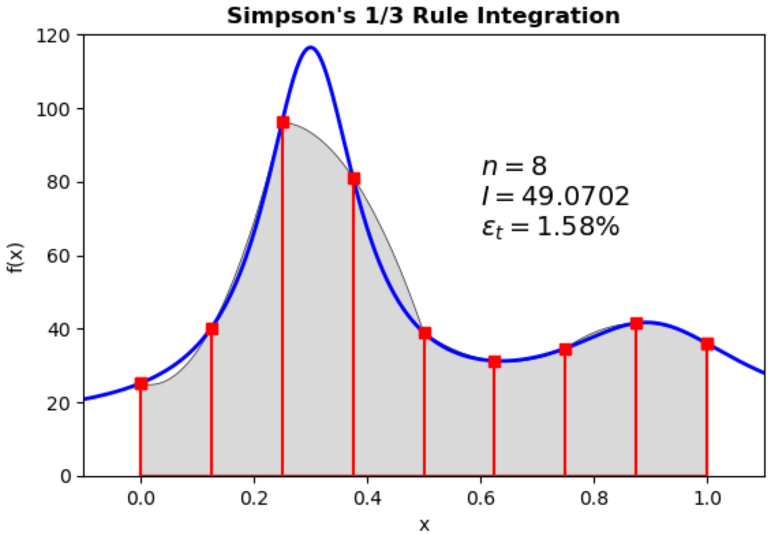
- Or, in $(width) \times (avg.value)$ form

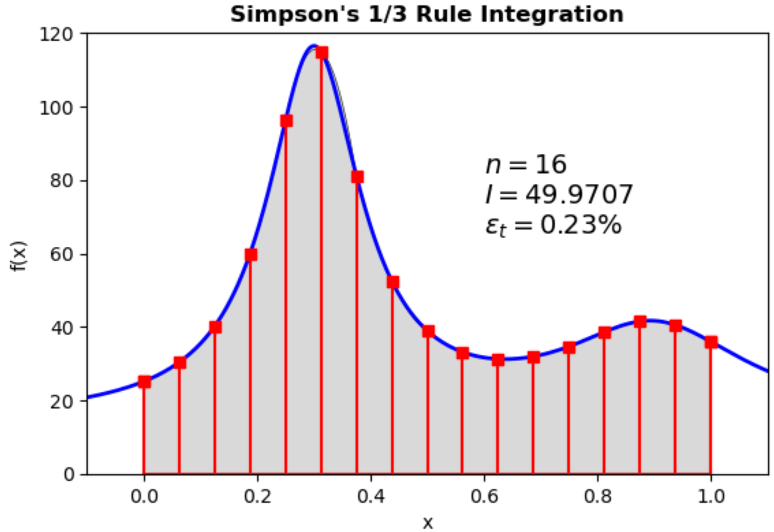
$$\hat{I} = (b-a)\frac{\left[f(x_0) + 4\sum_{i=1,3,5\ldots}^{n-1} f(x_i) + 2\sum_{j=2,4,6\ldots}^{n-2} f\left(x_j\right) + f(x_n)\right]}{3n}$$

Composite Simpson's 1/3 Rule – Example

- Accuracy improves as the number of segments increases
  - $\bar{f}^{(4)}$ over each segment decreases
  - $f(x)$ better approximated as quadratic over smaller regions







Simpson's 3/8 Rule

Simpson's 3/8 Rule

- Approximate $f(x)$ with a third-order polynomial

$$f(x) \approx f_3(x)$$

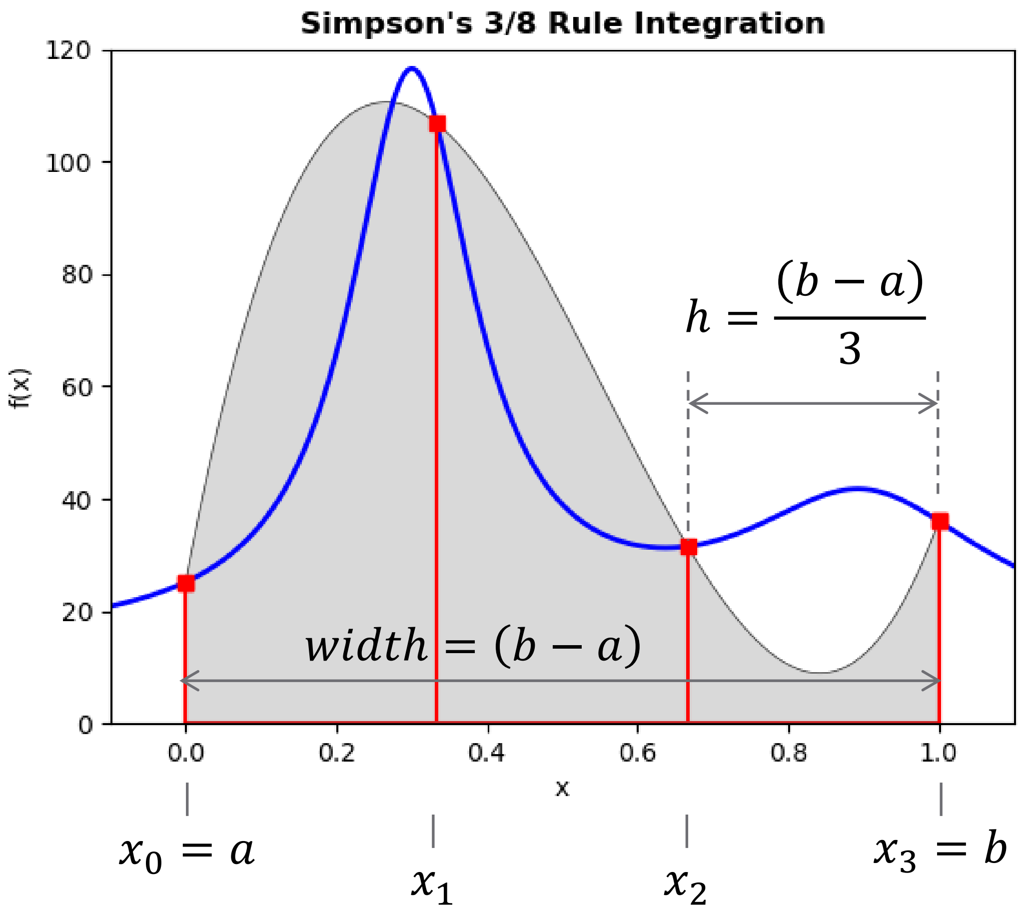
  where $f_3(x)$ can, again, be expressed as a Lagrange polynomial

- Approximate the integral of $f(x)$ as the integral of the cubic approximation

$$I \approx \hat{I} = \int_a^b f_3(x)\, dx$$

Simpson's 3/8 Rule

- Now fitting a ***cubic*** to $f(x)$
- ***Four points*** required: x_0, x_1, x_2, and x_3
- Integration interval, $[a, b]$ divided into ***three segments***
- Points must be ***evenly spaced***



Simpson's 3/8 Rule

- Evaluating the integral of the cubic approximation, $f_3(x)$, yields ***Simpson's 3/8 rule***:

$$\boxed{\hat{I} = \frac{3h}{8}\left[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)\right]}$$

- Or, in $\hat{I} = (width) \times (avg.value)$ form:

$$\boxed{\hat{I} = (b-a)\frac{f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)}{8}}$$

Simpson's 3/8 Rule – Error

- The error associated with Simpson's 3/8 rule is

$$E_t = \frac{3}{80} h^5 f^{(4)}(\xi) = \frac{(b-a)^5}{6480} f^{(4)}(\xi)$$

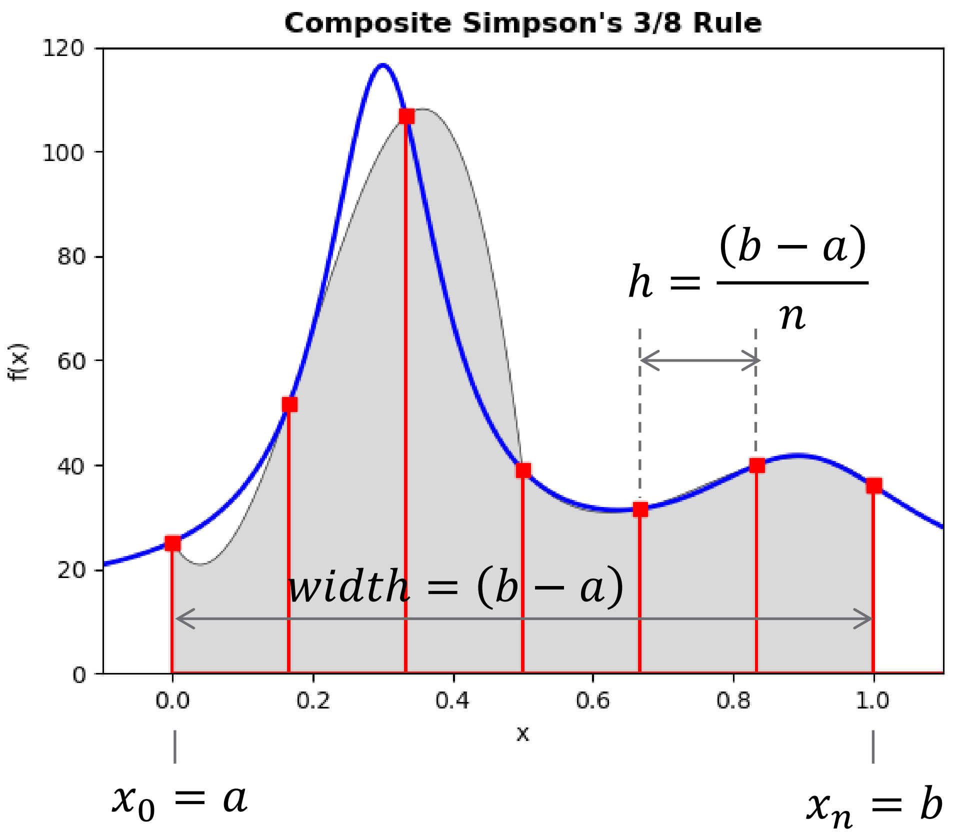
- Error is proportional to the fourth derivative of $f(x)$
  - ***Third-order accuracy***
    - ***Same as Simpson's 1/3 rule***
  - For nonzero $f^{(4)}$, error is slightly lower than Simpson's 1/3 rule

Composite Simpson’s 3/8 Rule

- Accuracy can be improved by dividing the interval $[a, b]$ into n segments

- Each application of Simpson’s 3/8 rule requires four points, and three segments
  - Total number of ***segments must be divisible by three***
  - Can be used in conjunction with Simpson’s 1/3 rule to accommodate an odd number of segments

- $f(x)$ ***approximated as a cubic*** over each group of three adjacent segments



Composite Simpson's 3/8 Rule

- Divide $[a, b]$ into n segments, and the integral into $n/3$ segments

$$I = \int_{x_0}^{x_3} f(x)dx + \int_{x_3}^{x_6} f(x)dx + \cdots + \int_{x_{n-3}}^{x_n} f(x)dx$$

- Approximate each term using Simpson's 3/8 rule

$$\begin{aligned}\hat{I} \\ &= \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)] + \frac{3h}{8}[f(x_3) + 3f(x_4) + 3f(x_5) + f(x_6)] + \cdots \\ &+ \frac{3h}{8}[f(x_{n-3}) + 3f(x_{n-2}) + 3f(x_{n-1}) + f(x_n)]\end{aligned}$$

- Using summation notation

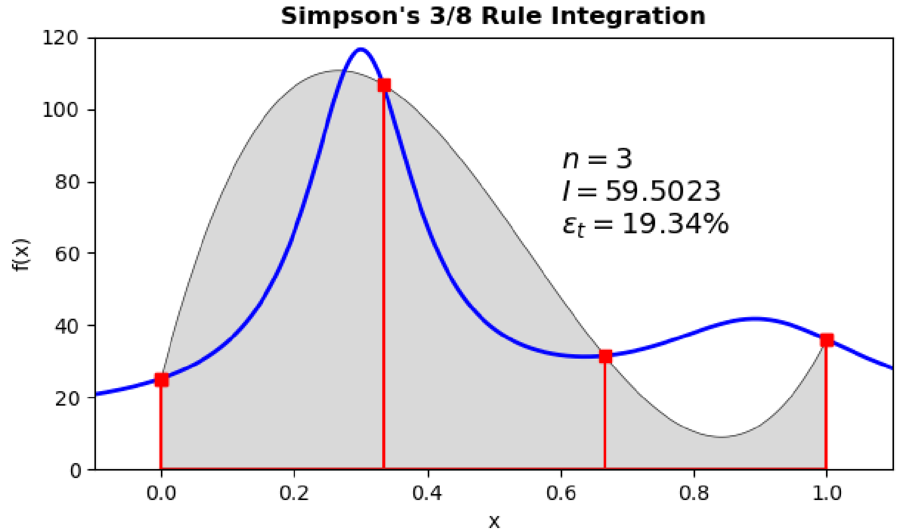
$$\boxed{\hat{I} = \frac{3h}{8}\left[f(x_0) + 3\sum_{i=1,4,7\ldots}^{n-2} f(x_i) + 3\sum_{j=2,5,8\ldots}^{n-1} f\left(x_j\right) + 2\sum_{k=3,6,9\ldots}^{n-3} f(x_k) + f(x_n)\right]}$$

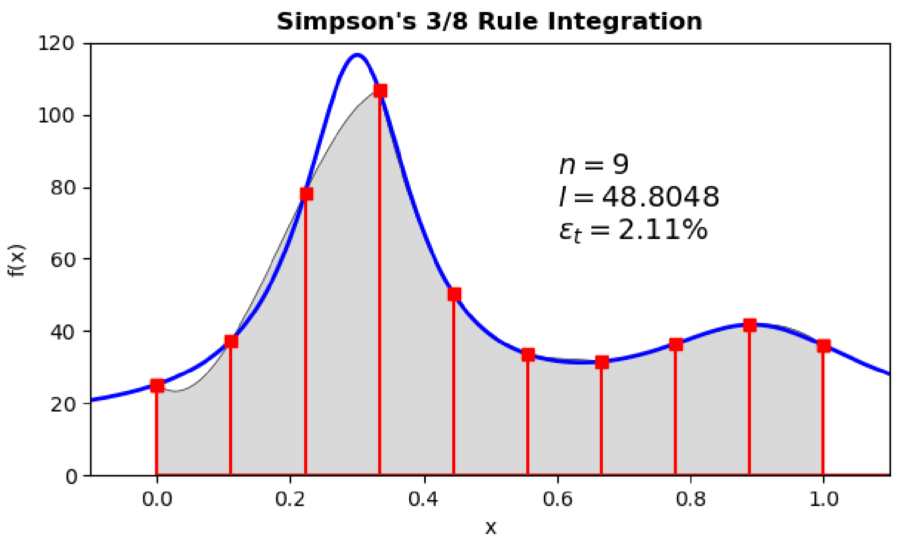
- Or, in $(width) \times (avg.value)$ form

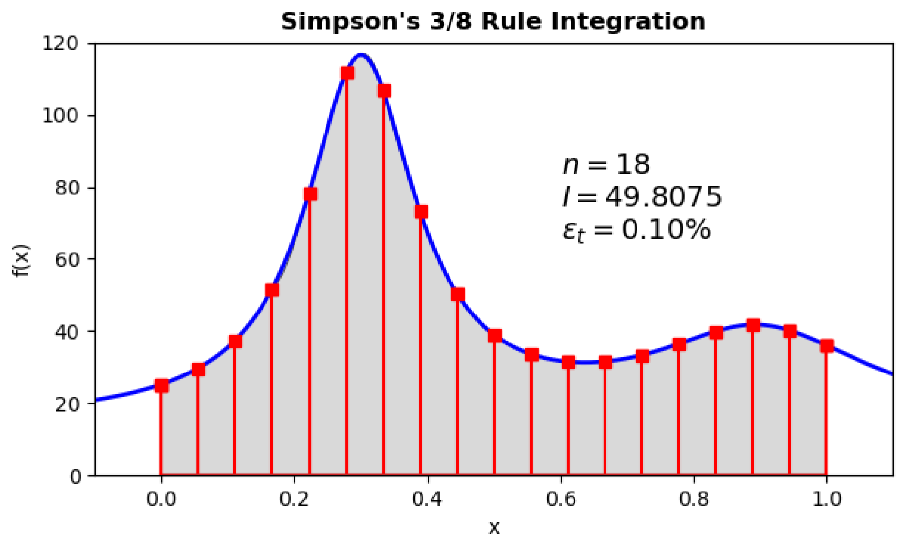
$$\hat{I} = (b-a)\frac{3\left[f(x_0) + 3\sum_{i=1,4,7\ldots}^{n-2} f(x_i) + 3\sum_{j=2,5,8\ldots}^{n-1} f\left(x_j\right) + 2\sum_{k=3,6,9\ldots}^{n-3} f(x_k) + f(x_n)\right]}{8n}$$

Composite Simpson's 3/8 Rule – Example

- Accuracy improves as the number of segments increases
  - $\bar{f}^{(4)}$ over each segment decreases
  - $f(x)$ better approximated as a cubic over smaller regions







Higher-Order Formulas

Higher-Order Formulas

- Typically, Simpson's 1/3 rule, used in conjunction with Simpson's 3/8 rule (for odd n), is sufficient
- Possible to use ***higher-order polynomials*** to approximate $f(x)$
  - n segments and $n+1$ points needed for n^{th}-order polynomial approximation
- ***Closed*** and ***open*** integration formulas exist
- Boole's rule will show up in a different form later when we cover adaptive quadrature

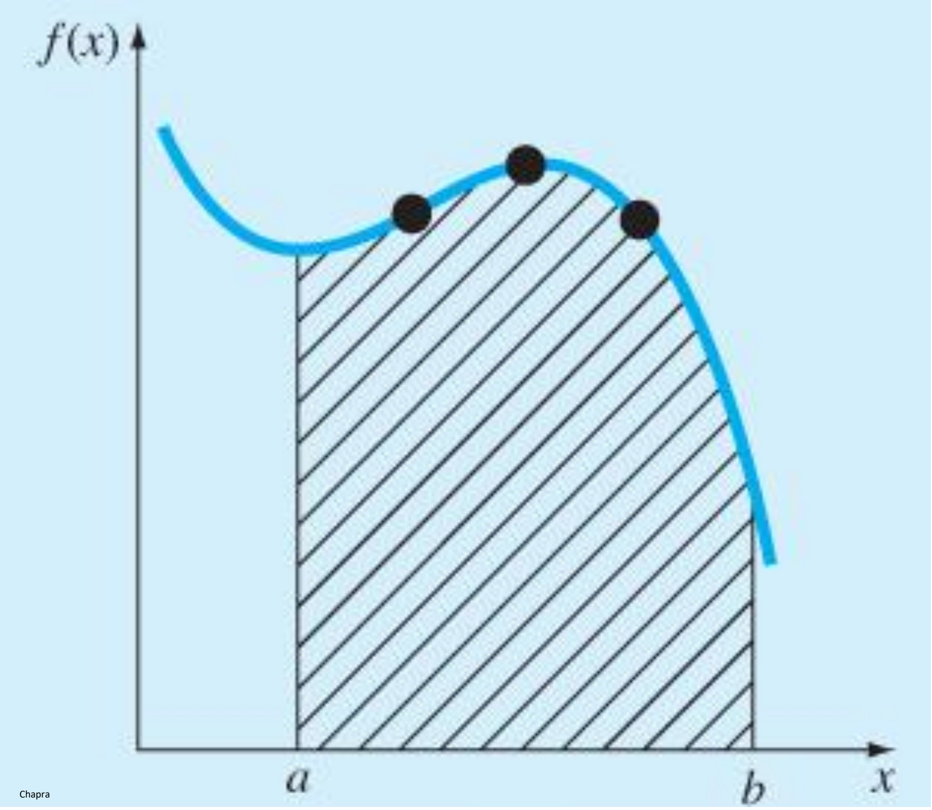
Higher-Order Newton-Cotes Formulas – Closed

n	Name	Formula	Error prop. to
1	Trapezoidal rule	$\hat{I} = \frac{h}{2}\frac{f(x_0) + f(x_1)}{2}$	$f''(\xi)$
2	Simpson's 1/3 rule	$\hat{I} = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$	$f^{(4)}(\xi)$
3	Simpson's 3/8 rule	$\hat{I} = \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$	$f^{(4)}(\xi)$
4	Boole's rule	$\hat{I} = \frac{2h}{45}[7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)]$	$f^{(6)}(\xi)$
5	-	$\hat{I} = \frac{5h}{288}[19f(x_0) + 75f(x_1) + 50f(x_2) + 50f(x_3) + 75f(x_4) + 19f(x_5)]$	$f^{(6)}(\xi)$

The step size in the above formulas is: $h = \frac{(b-a)}{n}$

Open Integration Formulas

- Function values not know at the limits of integration
  - n segments
  - $(n-1)$ points
  - $(n-2)^{nd}$-order polynomial approximation



Higher-Order Newton-Cotes Formulas – Open

Segments (n)	Points	Formula	Error prop. to
2	1	$\hat{I} = (b-a)f(x_1)$	$f''(\xi)$
3	2	$\hat{I} = (b-a)\dfrac{f(x_1)+f(x_2)}{2}$	$f^{(4)}(\xi)$
4	3	$\hat{I} = (b-a)\dfrac{2f(x_1)+f(x_2)+2f(x_3)}{3}$	$f^{(4)}(\xi)$
5	4	$\hat{I} = (b-a)\dfrac{11f(x_1)+f(x_2)+f(x_3)+11f(x_4)}{24}$	$f^{(6)}(\xi)$
6	5	$\hat{I} = (b-a)\dfrac{11f(x_1)-14f(x_2)+26f(x_3)-14f(x_4)+11f(x_5)}{20}$	$f^{(6)}(\xi)$

Integration of Functions

Integration of Functions

- Newton-Cotes formulas can be used to integrate functions or discrete data points
  - Evenly-spaced data points are assumed
- ***If $f(x)$ is known, spacing of x-values can be chosen to improve accuracy***
  - Spacing need not be uniform
  - Can locate points specific distances from limits of integration or segment edges to improve accuracy
  - Can use larger step size where acceptable, reduced step size where necessary
    - Effectively trade off accuracy and efficiency

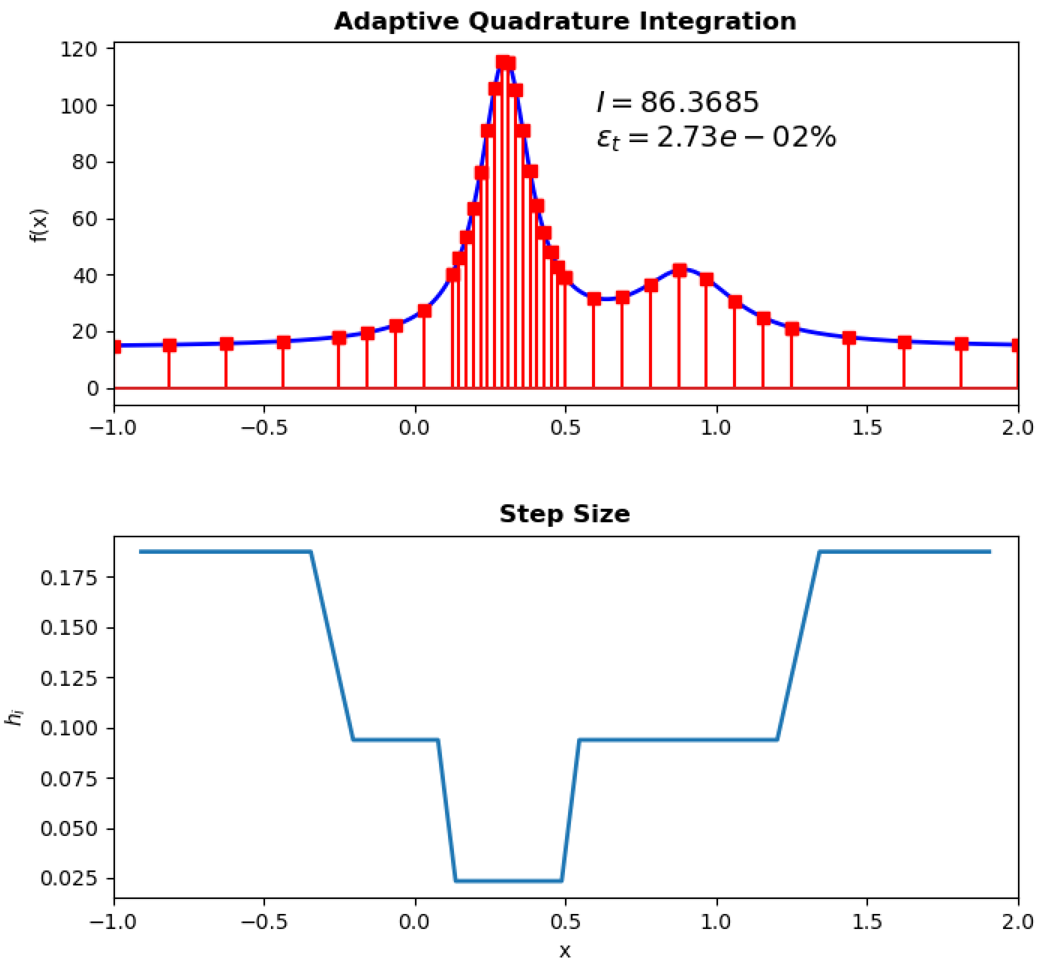
Methods for integrating functions

- ***Romberg integration***
  - Combine two trapezoidal rule estimates with different step sizes to yield a third, more accurate estimate
- ***Gauss quadrature***
  - Spacing of points within the integration segments chosen to improve accuracy of Newton-Cotes formulas
- ***Adaptive Quadrature***
  - Adaptively refine step size to achieve desired accuracy
  - Smaller step size in some regions, larger in others
  - Uses some of the techniques used by Romberg integration

Adaptive Quadrature

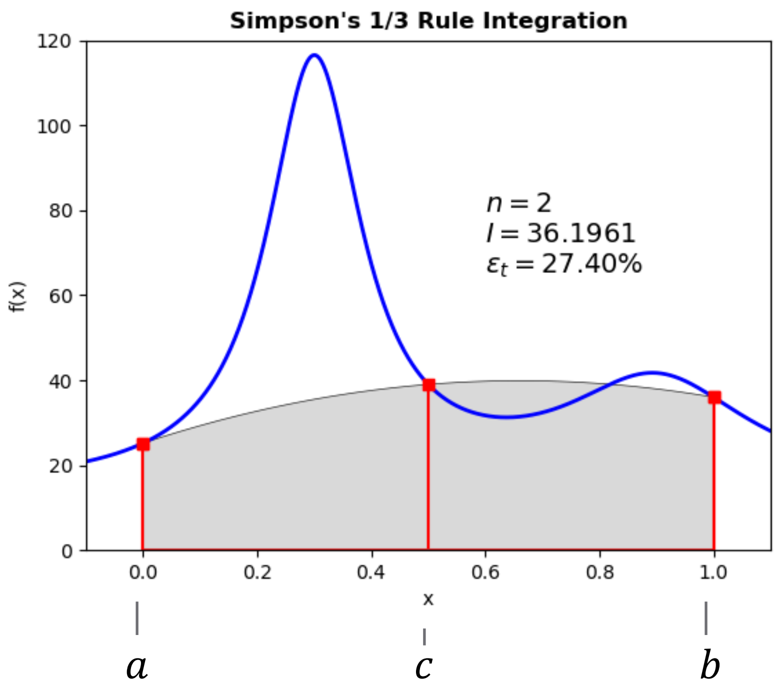
Adaptive Quadrature

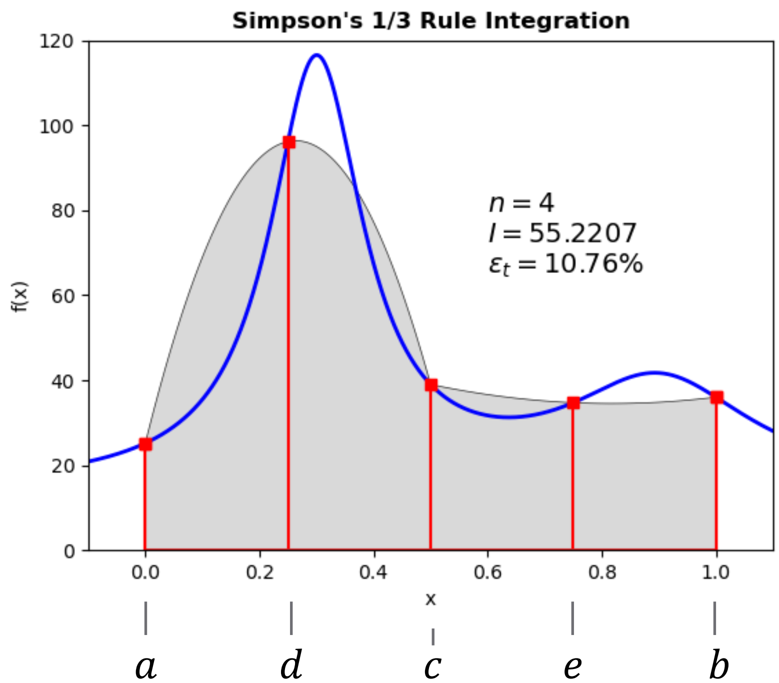
- ***Vary step size to achieve desired accuracy over each segment***
  - Smaller step size where $f(x)$ varies rapidly
  - Larger step size where $f(x)$ varies gradually
- Integration method used is ***Simpson's 1/3 rule***



Adaptive Quadrature

- Apply Simpson's 1/3 rule to approximate the integral at two different step sizes, $\hat{I}(h_1)$ and $\hat{I}(h_2)$, where $h_2 = h_1/2$





$$\hat{I}(h_1) = \frac{h_1}{3}[f(a) + 4f(c) + f(b)]$$

$$\hat{I}(h_2) = \frac{h_2}{3}[f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)]$$

Adaptive Quadrature

- Use $\hat{I}(h_1)$ and $\hat{I}(h_2)$ to approximate the error:

$$E_a = \hat{I}(h_2) - \hat{I}(h_1) \tag{1}$$

- Two possible ways to proceed:
  - If $E_a \leq abstol$
    - Using an approach similar to ***Romberg integration***, combine $\hat{I}(h_1)$ and $\hat{I}(h_2)$ to yield a third, more accurate estimate of the integral
  - If $E_a > abstol$
    - Divide [a,b] into two segments: $[a, c]$ and $[c, b]$
    - Calculate $\hat{I}(h_1)$ and $\hat{I}(h_2)$ for each segment
      - Single- and double-segment Simpson's 1/3 approximations
    - Use (1) to approximate the error for each sub-interval

Adaptive Quadrature – $E_a \leq abstol$

- If E_a as calculated by (1) is acceptable (i.e. $< abstol$) we can use $\hat{I}(h_1)$ and $\hat{I}(h_2)$ to calculate a third, more accurate approximation
  - This is the basic principal used in ***Romberg integration***
  - We'll now derive the formula used to combine $\hat{I}(h_1)$ and $\hat{I}(h_2)$
- Each estimate is the true integral plus some error

$$I = \hat{I}(h_1) - E(h_1) = \hat{I}(h_2) - E(h_2) \qquad (2)$$

- We've seen that Simpson's 1/3 rule error can be approximated as

$$E_a(h) = \frac{(b-a)h^4}{180}\bar{f}^{(4)} \qquad (3)$$

where $\bar{f}^{(4)}$ is the average value of $f^{(4)}(x)$ over the integration interval

Adaptive Quadrature – $E_a \leq abstol$

- Equation (3) gives approximate error at each step size:

$$E_a(h_1) = \frac{(b-a)h_1^4}{180}\bar{f}^{(4)} \tag{4}$$

$$E_a(h_2) = \frac{(b-a)h_2^4}{180}\bar{f}^{(4)} \tag{5}$$

- Divide (4) by (5)

$$\frac{E_a(h_1)}{E_a(h_2)} = \frac{h_1^4}{h_2^4} \tag{6}$$

- Solve for $E_a(h_1)$

$$E_a(h_1) = E_a(h_2)\frac{h_1^4}{h_2^4} \tag{7}$$

Adaptive Quadrature – $E_a \leq abstol$

- Restate (2) as an approximation

$$\hat{I}(h_1) - E_a(h_1) \approx \hat{I}(h_2) - E_a(h_2) \tag{8}$$

- Substitute (7) into (8)

$$\hat{I}(h_1) - E_a(h_2)\frac{h_1^4}{h_2^4} \approx \hat{I}(h_2) - E_a(h_2) \tag{9}$$

- Solve (9) for the error of the more accurate approximation

$$E_a(h_2) = -\left[\frac{\hat{I}(h_1)-\hat{I}(h_2)}{1-\left(\frac{h_1}{h_2}\right)^4}\right] \tag{10}$$

Adaptive Quadrature – $E_a \leq abstol$

- According to (2)

$$I = \hat{I}(h_2) - E(h_2) \tag{11}$$

so

$$I \approx \hat{I} = \hat{I}(h_2) - E_a(h_2) \tag{12}$$

- Substituting (10) into (12)

$$\hat{I} = \hat{I}(h_2) + \frac{\hat{I}(h_1) - \hat{I}(h_2)}{1 - \left(\frac{h_1}{h_2}\right)^4}$$

- And, since $h_1 = 2 \cdot h_2$, the integral approximation is

$$\boxed{\hat{I} = \hat{I}(h_2) + \frac{1}{15}\left[\hat{I}(h_2) - \hat{I}(h_1)\right]} \tag{13}$$

- Which can be shown to be equivalent to ***Boole's rule***

Adaptive Quadrature – $E_a \leq abstol$

- ***Summarizing***:
  - Want to numerically integrate $f(x)$ over $[a, b]$
  - Calculate $\hat{I}(h_1)$ and $\hat{I}(h_2)$
  - Approximate the error as

$$E_a = \hat{I}(h_2) - \hat{I}(h_1)$$

  - If $E_a \leq abstol$, calculate the integral using Boole's rule

$$\hat{I} = \hat{I}(h_2) + \frac{1}{15}\left[\hat{I}(h_2) - \hat{I}(h_1)\right]$$

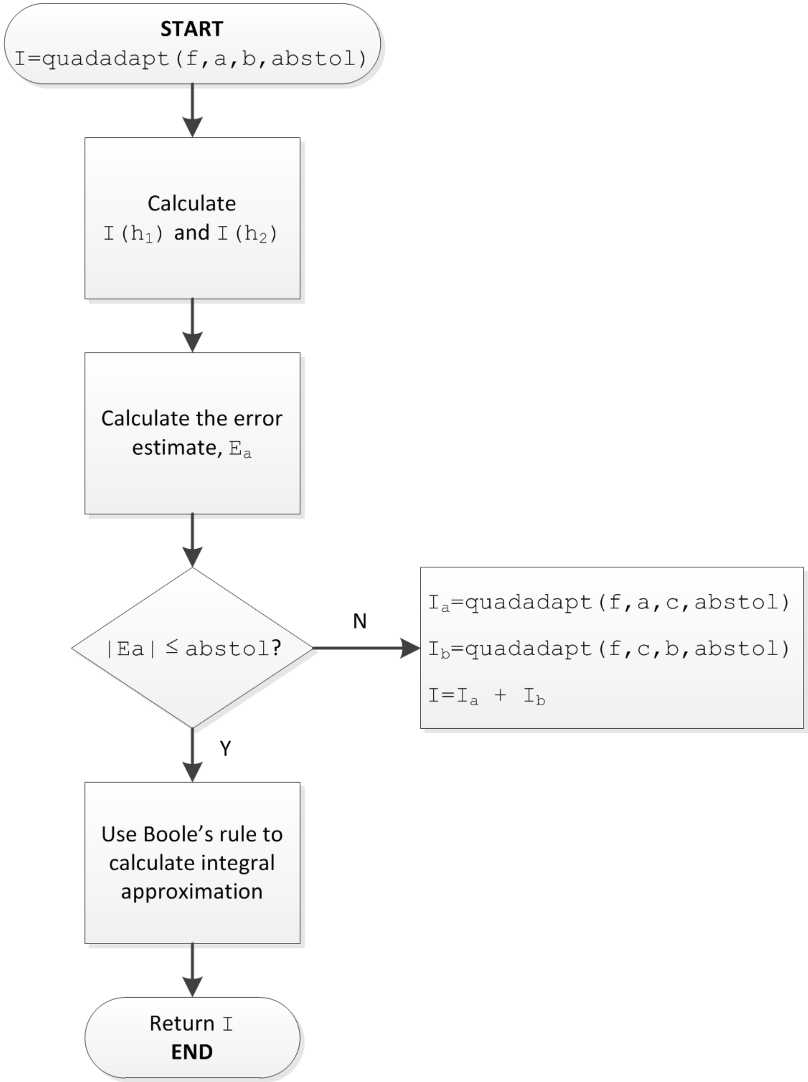
  - Next, we'll look at what to do if $E_a > abstol$

Adaptive Quadrature – $E_a > abstol$

- If $E_a > abstol$, reduce the step size and try again
  - Subdivide the integration interval, $[a, b]$, into two sub-intervals, $[a, c]$ and $[c, b]$
  - For each subinterval:
    - Calculate $\hat{I}(h_1)$ and $\hat{I}(h_2)$
    - Calculate E_a
    - If $E_a \leq abstol$, calculate $\hat{I}$ for that sub-interval using Boole's rule
    - If $E_a > abstol$, further subdivide the sub-interval into two smaller sub-intervals
      - Calculate $\hat{I}(h_1)$ and $\hat{I}(h_2)$, then E_a…
  - Eventually, total integral approximation is the sum of all individual sub-interval integral approximations

Adaptive Quadrature – Recursive Algorithm

- Adaptive quadrature uses a ***recursive algorithm***
  - A function that calls itself
- Integration interval is continually subdivided until approximate error is acceptable
- $\hat{I}$ returned by function is the sum of the individual $\hat{I}$ values

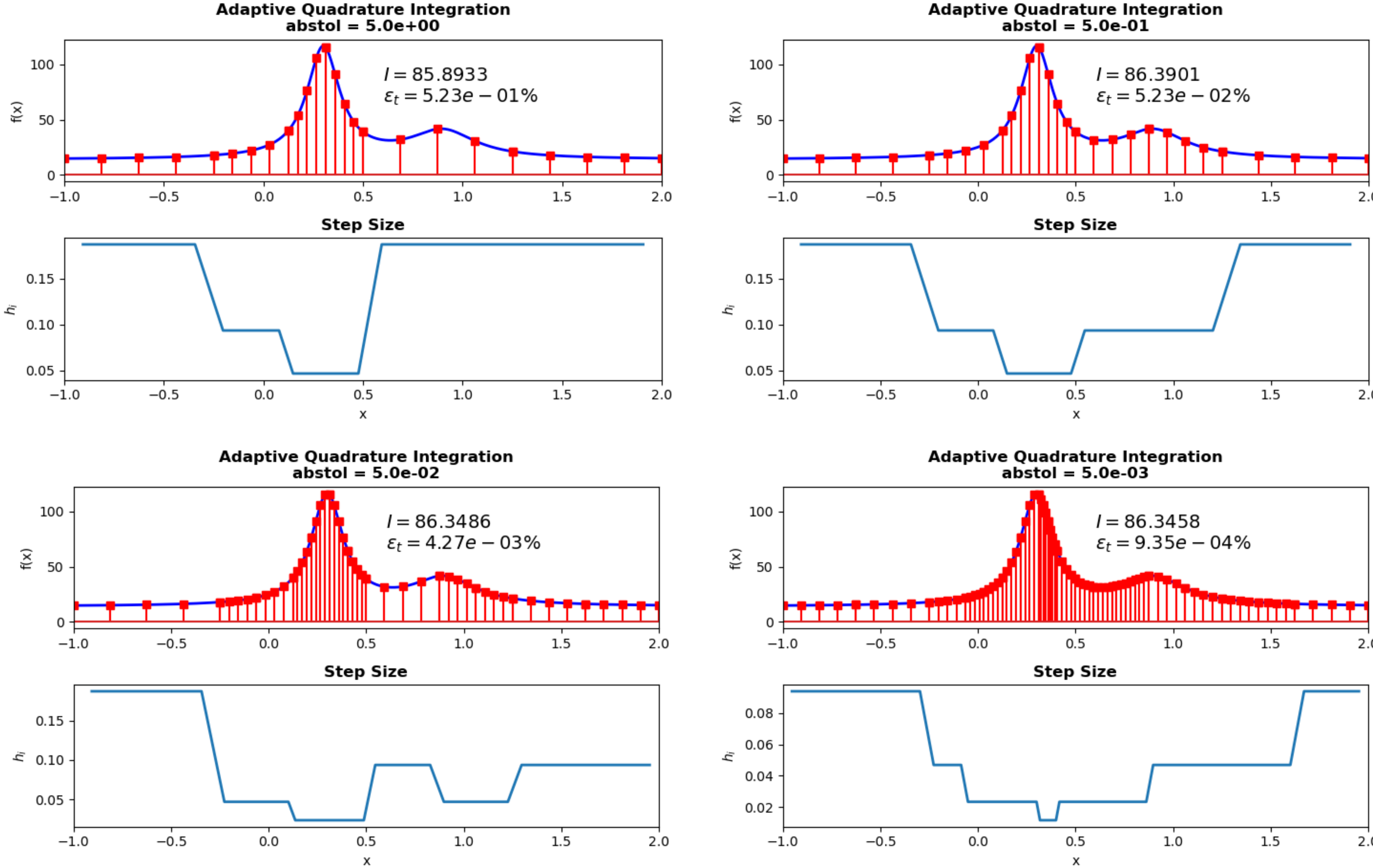


Adaptive Quadrature – quadadapt( )


```python
import numpy as np

def quadadapt(f,a,b,abstol):
    '''
    Integrates f(x) over [a, b] using adaptive quadrature.

    Parameters
    ----------
    f : function to be integrated
    a : lower limit of integration
    b : upper limit of integration
    abstol : stopping criterion - absolute (not relative) value

    Returns
    -------
    I : approximation of the integral

    '''

    h1 = (b-a)/2
    h2 = h1/2
    c = (a+b)/2
    d = (a+c)/2
    e = (c+b)/2

    # single-application simpson's 1/3 rule
    Ih1 = h1/3*(f(a) + 4*f(c) + f(b))

    # double-application simpson's 1/3 rule
    Ih2 = h2/3*(f(a) + 4*f(d) + 2*f(c) + 4*f(e) + f(b))

    # approximate the error
    Ea = abs(Ih2 - Ih1)

    # If error is less than abstol, return Boole's rule
    # estimate. Otherwise, invoke quadadapt on each
    # half of the interval.
    if Ea <= abstol:
        I = Ih2 + 1/15*(Ih2 - Ih1)
    else:
        Ia = quadadapt(f,a,c,abstol)
        Ib = quadadapt(f,c,b,abstol)
        I = Ia + Ib

    return I
```

- Inputs: function handle, limits of integration, and tolerance
  - On subsequent recursive calls, `a` and `b` will be sub-interval limits
- Step sizes and x-values for two Simpson's 1/3 rule estimates
- Integral estimates at two different step sizes
- Approximate error
- Boole's rule estimate
- Recursive function calls
- Sum the sub-interval integral estimates

Adaptive Quadrature – Examples



Integrating Functions in Python

Integrating Functions – `integrate.quad()`

- When we have an expression for the function to be integrated, we can use SciPy's `integrate.quad()` function:

```
I, err = integrate.quad(f,a,b)
```

  - `f`: the function to be integrated
  - `a`: lower integration limit
  - `b`: upper integration limit
  - `I`: numerical approximation of the integral
  - `err`: approximate absolute error

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

Exercise – Integration in Python

Exercise

- The impulse response of a certain 2nd-order system is given by

$$h(t) = 5.2414e^{-\alpha t}\sin(\omega_d t)$$

where $\alpha = 1.5$ and $\omega_d = 4.7697 rad/sec$

- A system's step response is the integral of its impulse response. For this system, the step response is

$$g(t) = 1 - e^{-\alpha t}\cos(\omega_d t) - 0.3145e^{-\alpha t}\sin(\omega_d t)$$

- Plot $g(t)$ for $0 \leq t \leq 10sec$ using a small sampling interval (e.g. $1msec$)
- For a variety of step sizes (e.g. $500,\ 200,\ 100,\ 10, 1msec$)
  - Calculate $\hat{g}(t)$ using `cumulative_trapezoid()` and superimpose on the plot of $g(t)$
  - Calculate the steady-state value of the step response using `trapezoid()`
  - Notice the effect of step size on the accuracy of the integral
- Also, calculate the steady-state step response value using `quad()`