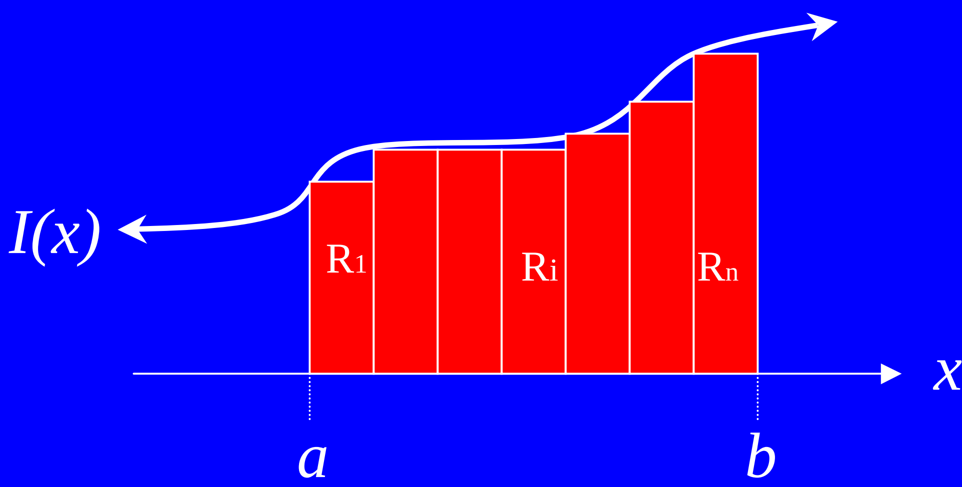
# Numerical Integration

- Various techniques for estimating area under a function




$$\int_a^b I(x)dx \approx \sum_{i=1}^{n} R_i$$

- In general, we assume functions are smooth and continuous

# Numerical Quadrature

- Quadrature rules based on polynomial interpolation
- Integrand function *f* sampled at points
- Unique polynomial interpolating these points is determined
- Integral of interpolant is taken as approximation to integral of original function.

# Numerical Quadrature

- *n*-point quadrature formula

$$I(f) = \int_a^b f(x)dx = \sum_{i=1}^{n} w_i f(x_i) + R_n$$

$$I(f) \approx \sum_{i=1}^{n} w_i f(x_i)$$

$x_i$: nodes where f is evaluated
$w_i$: weights
$R_n$: Error term

- In practice, interpolation is used to choose only the weights in the quadrature rule

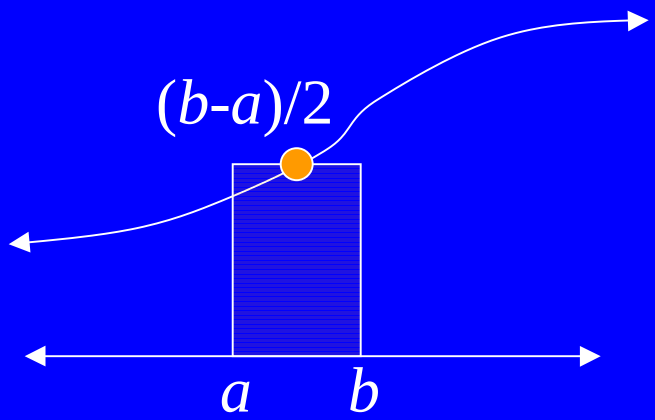
# Newton-Cotes Quadrature

- Equally spaced points, whether we use tabular data, or evaluate function directly
- For *n* points, polynomial interpolation of degree *n*-1 can be used to generate n-point quadrature rule
- A polynomial function of degree *n*-1 will be integrated exactly with an *n*-point quadrature rule

# Common Newton-Cotes Quadrature Rules

- Midpoint (or rectangle) rule - 1 point
- Trapezoid rule - 2 points
- Simpson's rule - 3 points

# Midpoint (rectangle) Rule




Interpolation function derived at midpoint of interval by a constant (polynomial of degree zero)

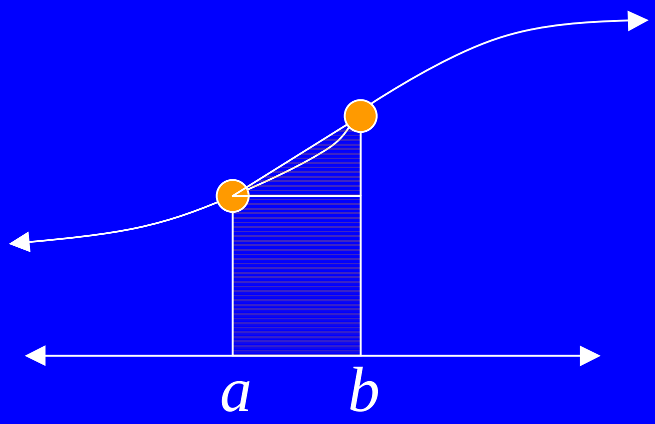
$$I(f) \approx M(f) = (b-a)\, f\left(\frac{a+b}{2}\right)$$

Area of rectangle

$$I(f) \approx \sum_{i=1}^{1} w_i f(x_i)$$

$$w_1 = b - a, \quad x_1 = \frac{a+b}{2}$$

# Trapezoid Rule




Interpolation function derived at endpoints of interval by a straight line (polynomial of degree one)

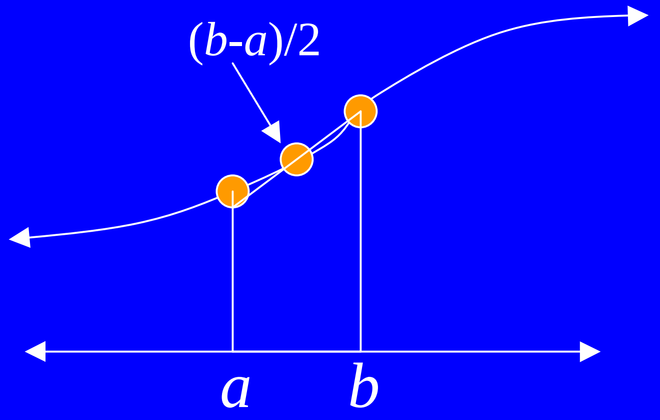
$$I(f) \approx T(f) = \frac{b-a}{2}\big(f(a) + f(b)\big)$$

Area of trapezoid

$$I(f) \approx \sum_{i=1}^{2} w_i f(x_i)$$

$$w_1 = \frac{b-a}{2}, \quad x_1 = a, \quad w_2 = \frac{b-a}{2}, \quad x_2 = b$$

# Simpson's Rule




Interpolation function derived at endpoints and midpoint of interval by a quadratic (polynomial of degree two)

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

$$I(f) \approx \sum_{i=1}^{3} w_i f(x_i)$$

$$w_1 = \frac{b-a}{6}, \quad x_1 = a, \quad w_2 = \frac{2(b-a)}{3}, \quad x_2 = \frac{a+b}{2},$$

$$w_3 = \frac{b-a}{6}, \quad x_3 = b$$

# Method of Undetermined Coefficients

- Another approach for finding weights of quadrature rules
- Use monomial basis functions and form system of equations by forcing quadrature rule to integrate each of the basis functions exactly
- Solve system of equations for weights

# Example - Finding Weights for Three-Point Rule

$$\int_a^b g(x)dx \approx w_1 g(x_1) + w_2 g(x_2) + w_3 g(x_3)$$

*Use monomial basis functions and equally-spaced points*

$$g_1(x) = 1, \quad g_2(x) = x, \quad g_3(x) = x^2$$

$$x_1 = a, \quad x_2 = \frac{a+b}{2}, \quad x_3 = b$$

Form system of equations by forcing quadrature rule to integrate each of the basis functions exactly

$$\sum_{i=1}^{3} w_i g_1(x_i) = \int_a^b g_1(x)dx$$

$$\sum_{i=1}^{3} w_i g_2(x_i) = \int_a^b g_2(x)dx$$

$$\sum_{i=1}^{3} w_i g_3(x_i) = \int_a^b g_3(x)dx$$

# Finding Weights for Three-Point Rule

$$\sum_{i=1}^{3} w_i g_1(x_i) = \int_a^b g_1(x)dx$$

$$\sum_{i=1}^{3} w_i g_2(x_i) = \int_a^b g_2(x)dx$$

$$\sum_{i=1}^{3} w_i g_3(x_i) = \int_a^b g_3(x)dx$$

$$w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1 = \int_a^b 1dx = x\Big|_a^b = b - a$$

$$w_1 \cdot a + w_2 \cdot \frac{a+b}{2} + w_3 \cdot b = \int_a^b xdx = \frac{x^2}{2}\Big|_a^b = \frac{b^2 - a^2}{2}$$

$$w_1 \cdot a^2 + w_2 \cdot \left(\frac{a+b}{2}\right)^2 + w_3 \cdot b^2 = \int_a^b x^2dx = \frac{x^3}{3}\Big|_a^b = \frac{b^3 - a^3}{3}$$

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

*Weights for Simpson's Rule*

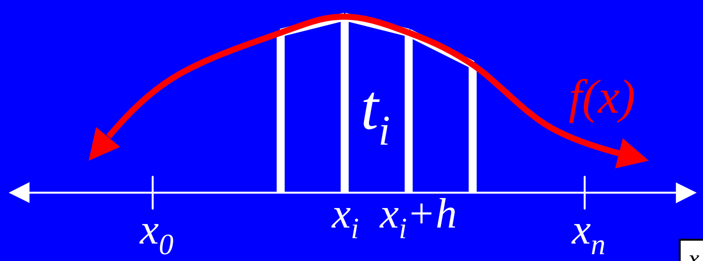
$$\begin{Bmatrix} w_1 \\ w_2 \\ w_3 \end{Bmatrix} = \begin{Bmatrix} (b-a)/6 \\ 2(b-a)/3 \\ (b-a)/6 \end{Bmatrix}$$

# Disadvantages of Newton-Cotes

- Large number of points will result in high-degree interpolant
- Requirement for evenly spaced points
- More accurate methods exist at less computational cost

# Trapezoid Method

- Newton-Cotes rules can be used in piecewise manner over large intervals




$$t_i = \frac{h}{2}\left(f(x_i) + f(x_i + h)\right)$$

$$\int_{x_0}^{x_n} f(x)dx \approx T = \sum_{i=0}^{n-1} \frac{h}{2}\left(f(x_i) + f(x_i + h)\right)$$

$$= \frac{h}{2}\left[\begin{array}{l} f(x_0) + f(x_0 + h) + f(x_1) + f(x_1 + h) + \cdots \\ + f(x_{n-1}) + f(x_{n-1} + h) \end{array}\right]$$

$$= \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right]$$

# Trapezoid Method - Example

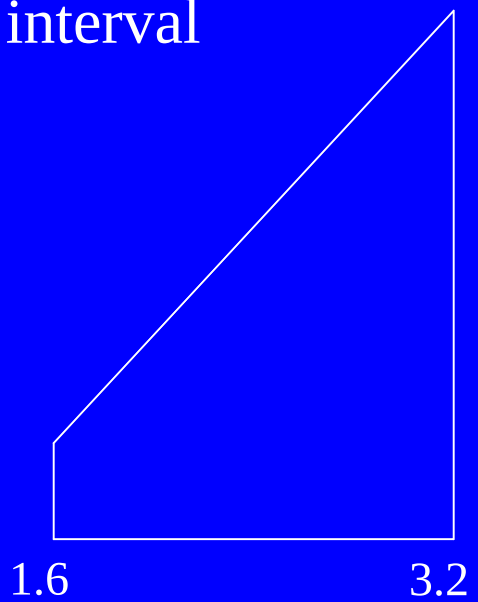
| *x* | *f(x)* |
| --- | --- |
| 1.6 | 4.953 |
| 1.8 | 6.050 |
| 2.0 | 7.389 |
| 2.2 | 9.025 |
| 2.4 | 11.023 |
| 2.6 | 13.464 |
| 2.8 | 16.445 |
| 3.0 | 20.086 |
| 3.2 | 24.553 |

Approximate

$$\int_{1.6}^{3.2} f(x)dx$$

using 1, 2, 4, and 8 intervals

1 interval




$$\int_{1.6}^{3.2} f(x)dx \approx \frac{h}{2}\left(f(1.6) + f(3.2)\right)$$

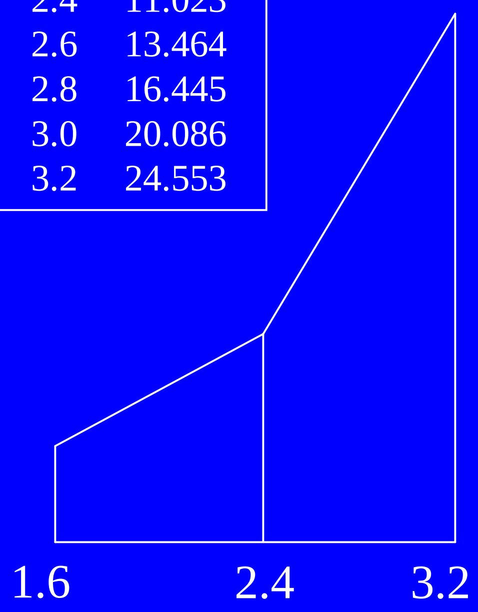
$$= 0.8(4.953 + 24.553)$$

$$= 23.605$$

# Trapezoid Method - Example

| *x* | *f(x)* |
|---|---|
| 1.6 | 4.953 |
| 1.8 | 6.050 |
| 2.0 | 7.389 |
| 2.2 | 9.025 |
| 2.4 | 11.023 |
| 2.6 | 13.464 |
| 2.8 | 16.445 |
| 3.0 | 20.086 |
| 3.2 | 24.553 |

2 intervals



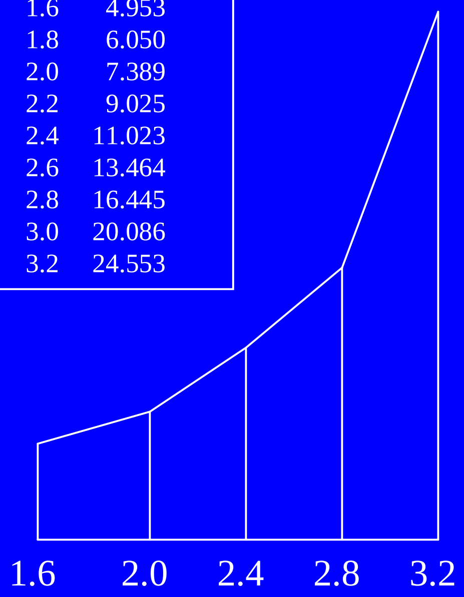

$$\int_{1.6}^{3.2} f(x)dx \approx \frac{h}{2}\left(f(1.6)+2f(2.4)+f(3.2)\right)$$

$$= 0.4(4.953+2\cdot 11.023+24.553)$$

$$= 20.621$$

# Trapezoid Method - Example

| *x* | *f(x)* |
|---|---|
| 1.6 | 4.953 |
| 1.8 | 6.050 |
| 2.0 | 7.389 |
| 2.2 | 9.025 |
| 2.4 | 11.023 |
| 2.6 | 13.464 |
| 2.8 | 16.445 |
| 3.0 | 20.086 |
| 3.2 | 24.553 |




4 intervals

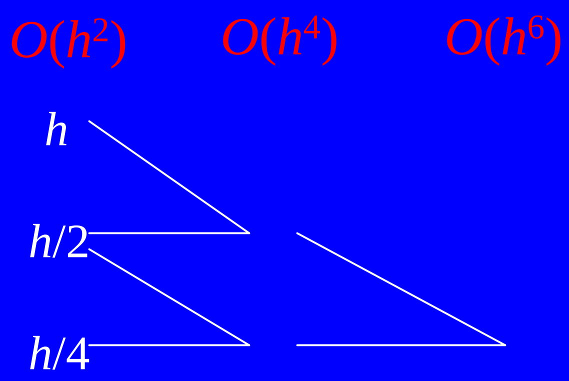
$$\int_{1.6}^{3.2} f(x)dx \approx \frac{h}{2}\left(f(1.6)+2f(2.0)+2f(2.4)+2f(2.8)+f(3.2)\right)$$
$$=0.2(4.953+14.778+22.046+32.890+24.553)$$
$$=19.844$$

8 intervals

$$\int_{1.6}^{3.2} f(x)dx \approx \frac{h}{2}\left(f(1.6)+2f(1.8)+2f(2.0)+2f(2.2)+2f(2.4)+2f(2.6)+2f(2.8)+2f(3.0)+f(3.2)\right)$$
$$=0.1(4.953+12.100+14.778+18.050+22.046+26.928+32.890+40.172+24.553)$$
$$=19.647$$

# Romberg Integration

- Trapezoid method with Richardson's extrapolation
  – Use trapezoid method with step size $h$
  – Use trapezoid method with step size $h/2$
  – Use results of step sizes $h$ and $h/2$, and extrapolate a more accurate solution



# Richardson's Extrapolation

Given an order $n$ numerical method, solutions $Q_1$ and $Q_2$ obtained with step sizes $h_1$ and $h_2$, we can extrapolate a more accurate solution $Q$:

$$Q = \frac{\left(\frac{h_1}{h_2}\right)^n Q_2 - Q_1}{\left(\frac{h_1}{h_2}\right)^n - 1}$$

# Theory of Richardson's Extrapolation

Assume error is of order $h^n$ for some numerical algorithm:

Let $Q$ be true solution (which we don't know yet)
Let $Q_1$ be an approximation using $h_1$
Let $Q_2$ be an approximation using $h_2$

*Size of error*

$$Q - Q_1 = ch_1^n$$
$$Q - Q_2 = ch_2^n$$

$\Rightarrow$

$$\frac{Q - Q_1}{h_1^n} = c$$
$$\frac{Q - Q_2}{h_2^n} = c$$

$\Rightarrow$

$$\frac{Q - Q_1}{h_1^n} = \frac{Q - Q_2}{h_2^n}$$

# Theory of Richardson's Extrapolation (continued)

$$\frac{Q-Q_1}{h_1^n} = \frac{Q-Q_2}{h_2^n}$$

$$\longrightarrow \quad Q-Q_1 = \left(\frac{h_1}{h_2}\right)^n (Q-Q_2)$$

$$\left(\frac{h_1}{h_2}\right)^n Q - Q = \left(\frac{h_1}{h_2}\right)^n Q_2 - Q_1$$

$$\longrightarrow \quad Q = \frac{\left(\frac{h_1}{h_2}\right)^n Q_2 - Q_1}{\left(\frac{h_1}{h_2}\right)^n - 1}$$

# Romberg Integration Example

- Use Trapezoid method for $O(h^2)$ approximations using interval sizes of $h$, $h/2$, $h/4$ and $h/8$

| $x$ | $f(x)$ |
|---|---|
| 1.6 | 4.953 |
| 1.8 | 6.050 |
| 2.0 | 7.389 |
| 2.2 | 9.025 |
| 2.4 | 11.023 |
| 2.6 | 13.464 |
| 2.8 | 16.445 |
| 3.0 | 20.086 |
| 3.2 | 24.553 |

Derived from $f(x) = e^x$

| Interval Size | $O(h^2)$ | $O(h^4)$ | $O(h^6)$ | $O(h^8)$ |
|---|---|---|---|---|
| $h$ | 23.605 | | | |
| $h/2$ | 20.621 | 19.626 | | |
| $h/4$ | 19.844 | 19.585 | 19.627 | |
| $h/8$ | 19.647 | 19.581 | 19.585 | 19.585 |

$$\int_{1.6}^{3.2} e^x dx = 19.57949777$$

*True solution*

# Gaussian Quadrature

- Gaussian rules based on polynomial interpolation
- Nodes not evenly spaced - locations of nodes chosen to maximize accuracy
- As before seeking

$$I(f) \approx \sum_{i=1}^{n} w_i f(x_i)$$

where choices of $x_i$ and $w_i$ are derived for optimality through nonlinear procedures

# Deriving Gaussian Quadrature Rules

- Use method of undetermined coefficients, where both the weights AND the nodes are unknown.
- Example - trying to derive a two-point rule

$$\int_{-1}^{1} f(x)dx \approx w_1 f(x_1) + w_2 f(x_2)$$

where $x_i$ and $w_i$ represent four unknowns. Therefore, we need four equations.

# Gaussian Quadrature Rules

$$\int_{-1}^{1} f(x)dx \approx w_1 f(x_1) + w_2 f(x_2)$$

*Use monomial basis functions*

$$g_1(x) = 1, \quad g_2(x) = x,$$
$$g_3(x) = x^2, \quad g_4(x) = x^3$$

Because of the use of higher degree basis functions, a polynomial of degree 2*n*-1 will be integrated exactly with an *n*-point Gaussian quadrature rule

Form system of equations by forcing quadrature rule to integrate each of the basis functions exactly

$$\sum_{i=1}^{2} w_i g_1(x_i) = \int_{-1}^{1} g_1(x)dx$$

$$\sum_{i=1}^{2} w_i g_2(x_i) = \int_{-1}^{1} g_2(x)dx$$

$$\sum_{i=1}^{2} w_i g_3(x_i) = \int_{-1}^{1} g_3(x)dx$$

$$\sum_{i=1}^{2} w_i g_4(x_i) = \int_{-1}^{1} g_4(x)dx$$

# Gaussian Quadrature Rules

$$\sum_{i=1}^{2} w_i g_1(x_i) = \int_{-1}^{1} g_1(x)dx$$

$$\sum_{i=1}^{2} w_i g_2(x_i) = \int_{-1}^{1} g_2(x)dx$$

$$\sum_{i=1}^{2} w_i g_3(x_i) = \int_{-1}^{1} g_3(x)dx$$

$$\sum_{i=1}^{2} w_i g_4(x_i) = \int_{-1}^{1} g_4(x)dx$$

$$w_1 \cdot 1 + w_2 \cdot 1 = \int_{-1}^{1} 1dx = x\Big|_{-1}^{1} = 1 + 1 = 2$$

$$w_1 x_1 + w_2 x_2 = \int_{-1}^{1} xdx = \frac{x^2}{2}\Big|_{-1}^{1} = \frac{1}{2} - \frac{1}{2} = 0$$

$$w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^{1} x^2 dx = \frac{x^3}{3}\Big|_{-1}^{1} = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

$$w_1 x_1^3 + w_2 x_2^3 = \int_{-1}^{1} x^3 dx = \frac{x^4}{4}\Big|_{-1}^{1} = \frac{1}{4} - \frac{1}{4} = 0$$

$$\begin{aligned} w_1 + w_2 &= 2 \\ w_1 x_1 + w_2 x_2 &= 0 \\ w_1 x_1^2 + w_2 x_2^2 &= \frac{2}{3} \\ w_1 x_1^3 + w_2 x_2^3 &= 0 \end{aligned}$$

*"One" solution*

$$x_1 = -\frac{1}{\sqrt{3}}, \quad x_2 = \frac{1}{\sqrt{3}}$$

$$w_1 = 1, \quad w_2 = 1$$

# Gaussian Quadrature Rule

$$\int_{-1}^{1} f(x)dx \approx w_1 f(x_1) + w_2 f(x_2)$$

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

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

Typically, we will need to transform general interval of integration [*a*, *b*], to a standard interval [α, β] (e.g. [-1, 1]) for which nodes and weights have been tabulated.

# Interval Transformations

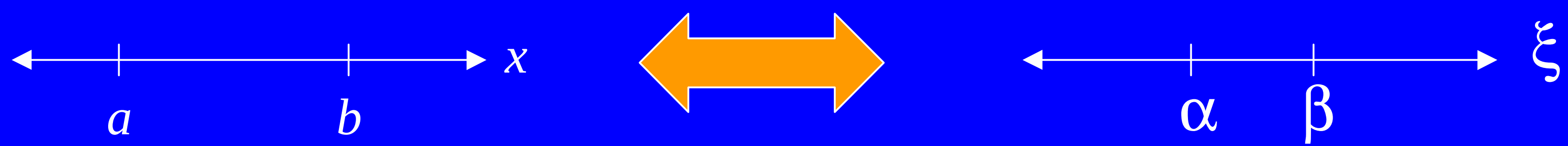
Given an integral

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

We want to apply a Gaussian quadrature rule of the form

$$\int_\alpha^\beta f(\xi)d\xi = \sum_{i=1}^n w_i f(\xi_i)$$

Therefore, we need a transformation between the two coordinate systems




$$I(g) = \int_a^b g(x)dx = \int_\alpha^\beta g(\chi(\xi))J(\xi)d\xi$$

where $\chi(\xi)$ is coordinate transformation

and $J(\xi) = \dfrac{dx}{d\xi} = \dfrac{d\chi(\xi)}{dx}$ is the "Jacobian"

# Linear Coordinate Transformation

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

$$J(\xi) = \frac{dx}{d\xi} = \frac{d\chi(\xi)}{dx} = \frac{b-a}{\beta-\alpha}$$

$$I(g) = \int_a^b g(x)dx = \int_\alpha^\beta g(\chi(\xi))J(\xi)d\xi$$

$$= \int_\alpha^\beta g\left(\frac{(b-a)\xi + a\beta - b\alpha}{\beta - \alpha}\right)\left(\frac{b-a}{\beta-\alpha}\right)d\xi$$

$$= \frac{b-a}{\beta-\alpha}\int_\alpha^\beta g\left(\frac{(b-a)\xi + a\beta - b\alpha}{\beta - \alpha}\right)d\xi$$

Application of quadrature rule

$$\approx \frac{b-a}{\beta-\alpha}\sum_{i=1}^{n} w_i g\left(\frac{(b-a)\xi_i + a\beta - b\alpha}{\beta - \alpha}\right)$$

# Gaussian Quadrature Example

Approximate with previously defined two-point rule

$$I(g)=\int_0^1 e^{-x^2}dx$$

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

$$x=\chi(\xi)=\frac{(b-a)\xi+a\beta-b\alpha}{\beta-\alpha}=\frac{(1-0)\xi+0+1}{1--1}=\frac{\xi+1}{2}$$

$$J(\xi)=\frac{dx}{d\xi}=\frac{d\chi(\xi)}{dx}=\frac{1}{2}$$

*Coordinate Transformation*

$$I(g)\approx\frac{b-a}{\beta-\alpha}\sum_{i=1}^{n} w_i g\left(\frac{(b-a)\xi_i+a\beta-b\alpha}{\beta-\alpha}\right)$$

$$=\frac{1}{2}\left(e^{-\left(\frac{(-1/\sqrt{3})+1}{2}\right)^2}+e^{-\left(\frac{(1/\sqrt{3})+1}{2}\right)^2}\right)$$

$$\approx 0.746595$$

*True solution*

$$\int_0^1 e^{-x^2}dx = 0.746824$$

# Values for Gaussian Quadrature

- Weights and nodes for numerous *n*-point Gaussian quadrature rules have already been tabulated
- See handout

# Multiple Integrals

- Straightforward extension of one dimension case

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

where $A$ is bounded by the lines $x=a$, $x=b$, $y=c$, and $y=d$.

$$\iint_A f(x,y)\,dA \approx \sum_{j=1}^{m} v_j \sum_{i=1}^{n} w_i f(x_i, y_j)$$

# Monte Carlo Methods

- For three or more dimensions, traditional quadrature methods become expensive
- Use of “random” techniques offers relatively inexpensive approach for approximating higher integrals

# Monte Carlo Algorithm

- Generate random points within interval, $\Omega$, of integration
- Evaluate the integrand at each random point
- Sum all of the integrand evaluations and divide by the number of evaluations, to get the *mean* function value
- Multiply this mean value by "size" of interval

# Monte Carlo Algorithm

$$\int_{\Omega} f(a,b,\ldots,z)\,d\Omega \approx \frac{\Omega}{n}\sum_{i=1}^{n} f(a_i,b_i,\ldots,z_i)$$

General $n$-dimensional form

- Error approximately $n^{-1/2}$
- Example - to gain extra decimal place of accuracy, $n$ must be increased by factor of 100
- Not competitive for one or two dimensions
- Convergence rate independent of number of dimensions!

# 1D Monte Carlo Example

1D Monte Carlo Formula

$$\int_a^b f(x)dx \approx \frac{b-a}{n}\sum_{i=1}^{n} f(x_i)$$

$$I(g) = \int_0^1 e^{-x^2} dx \rightarrow \int_0^1 e^{-x^2} dx \approx \frac{1}{n}\sum_{i=1}^{n} e^{-x^2}$$

Octave script

```
# The function we're integrating
function val = f(x)
     val = exp(-x^2);
endfunction

MAXN = 100;

n = 0;
sum = 0;
while (n < MAXN)

   # Generates random number
   #  between 0 and 1
   x = rand;

   sum = sum + f(x);

   n++;
endwhile

result = sum/n
```

| *n* | Sample Results | | | |
|---|---|---|---|---|
| 10 | 0.68897 | 0.85972 | 0.72331 | 0.82431 |
| 100 | 0.75350 | 0.74103 | 0.74532 | 0.77929 |
| 1000 | 0.75231 | 0.74837 | 0.74338 | 0.74616 |
| 10000 | 0.74438 | 0.74454 | 0.74635 | 0.74840 |

# 2D Monte Carlo Example

2D Monte Carlo Formula

$$\int_a^b \int_c^d f(x,y)\,dxdy \approx \frac{(b-a)(c-d)}{n} \sum_{i=1}^{n} f((x,y)_i)$$

$$I(g) = \int_0^1 \int_0^1 e^{-x^2 y} dx$$

↓

$$\int_0^1 \int_0^1 e^{-x^2 y} dxdy \approx \frac{1}{n} \sum_{i=1}^{n} e^{-x^2 y}$$

Octave script

```
# The function we're integrating
function val = f(x, y)
     val = exp(-y*x^2);
endfunction

MAXN = 10000;

n = 0;
sum = 0;
while (n < MAXN)

   # Generates random number
   #  between 0 and 1
   x = rand;
   y = rand;

   sum = sum + f(x,y);

   n++;
endwhile

result = sum/n
```

| $n$ | Sample Results | | | |
|---|---|---|---|---|
| 10 | 0.88563 | 0.82103 | 0.88990 | 0.86539 |
| 100 | 0.85712 | 0.87623 | 0.88007 | 0.87531 |
| 1000 | 0.86726 | 0.85909 | 0.86455 | 0.86057 |
| 10000 | 0.86232 | 0.85982 | 0.86063 | 0.86130 |

# 4D Monte Carlo Example

$$\int_a^b\int_c^d\int_e^f\int_g^h f(w,x,y,z)\,dw\,dx\,dy\,dz \approx$$

$$\frac{(b-a)(d-c)(f-e)(h-g)}{n}\sum_{i=1}^{n} f((w,x,y,z)_i)$$

$$I(g)=\int_0^1\int_0^1\int_0^1\int_0^1 e^{-wx^2yz}\,dw\,dx\,dy\,dz$$

$$\int_0^1\int_0^1\int_0^1\int_0^1 e^{-wx^2yz}\,dw\,dx\,dy\,dz \approx \frac{1}{n}\sum_{i=1}^{n} e^{-wx^2yz}$$

Octave script

```
# The function we're integrating
function val = f(w, x, y, z)
     val = exp(-y*w*z*x^2);
endfunction

MAXN = 100;

n = 0;
sum = 0;
while (n < MAXN)

   # Generates random number
   #  between 0 and 1
   w = rand;
   x = rand;
   y = rand;
   z = rand;

   sum = sum + f(w,x,y,z);

   n++;
endwhile

result = sum/n
```

| $n$ | Sample Results | | | |
|---|---|---|---|---|
| 10 | 0.97300 | 0.99244 | 0.97744 | 0.98583 |
| 100 | 0.96397 | 0.95508 | 0.95703 | 0.95573 |
| 1000 | 0.96288 | 0.96175 | 0.96482 | 0.96202 |
| 10000 | 0.96113 | 0.96192 | 0.96253 | 0.96230 |