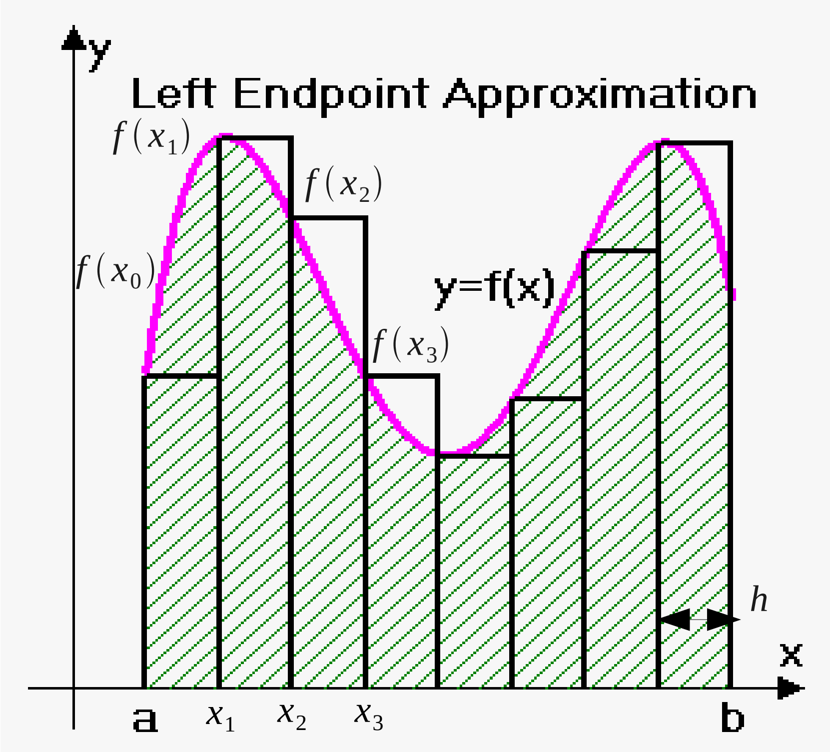
Numerical Integration

1D $$\int_a^b dx\, f(x) = \int_a^b f(x)\, dx$$

2D $$\int_a^b dx \int_c^d dy\, f(x, y)$$

Integrate f(x) from a to b: Elementary method: Endpoint Rule



$$\int_a^b dx\, f(x) = h \sum_{n=0}^{N-1} f(x_n) \qquad h = \frac{b-a}{N} \qquad x_n = a + nh$$

Endpoint rule implementation

```
function y = endpoint(f, a, b, n)
  % This function implements the simple endpoint rule.  Integration
  % is performed over n sub-intervals on the interval a <= x < b

  h = (b-a)/n;       % Step size
  x = a:h:(b-h);     % Sample x values -- must drop point at end
  s = f(x);          % f(x) values
  y = h*sum(s);      % Perform integration
end
```

Test Endpoint Rule for Different N

- How accurate is method (error)?
- How does error scale with N?

```
function test_endpoint()
  % This tests the endpoint integrator by integrating
  % x^2 from 0 -> 3 for different numbers of intervals.
  % The analytic result is (1/3)x^3, which should be 9.000.

  f = @(x) x.*x  % Anonymous function

  a = 0;
  b = 3;

  true = b*b*b/3;

  N = 1;
  for idx = 1:20
    N = N*2;
    act = endpoint(f, a, b, N);
    err = abs(true - act);
    printf('N = %d, true = %12.8f,  act = %12.8f,  err = %12.8f\n',
                 N, true, act, err);
  end

end
```

$$f(x) = x^2$$

$$\int_a^b x^2 \, dx = \frac{1}{3}\left(b^3 - a^3\right)$$

Test results

```
octave:5> test_endpoint
N = 2, true =   9.00000000,  act =   3.37500000,  err =   5.62500000
N = 4, true =   9.00000000,  act =   5.90625000,  err =   3.09375000
N = 8, true =   9.00000000,  act =   7.38281250,  err =   1.61718750
N = 16, true =   9.00000000,  act =   8.17382812,  err =   0.82617188
N = 32, true =   9.00000000,  act =   8.58251953,  err =   0.41748047
N = 64, true =   9.00000000,  act =   8.79016113,  err =   0.20983887
N = 128, true =   9.00000000,  act =   8.89480591,  err =   0.10519409
N = 256, true =   9.00000000,  act =   8.94733429,  err =   0.05266571
N = 512, true =   9.00000000,  act =   8.97364998,  err =   0.02635002
N = 1024, true =   9.00000000,  act =   8.98682070,  err =   0.01317930
N = 2048, true =   9.00000000,  act =   8.99340928,  err =   0.00659072
N = 4096, true =   9.00000000,  act =   8.99670437,  err =   0.00329563
N = 8192, true =   9.00000000,  act =   8.99835212,  err =   0.00164788
N = 16384, true =   9.00000000,  act =   8.99917604,  err =   0.00082396
N = 32768, true =   9.00000000,  act =   8.99958802,  err =   0.00041198
N = 65536, true =   9.00000000,  act =   8.99979401,  err =   0.00020599
N = 131072, true =   9.00000000,  act =   8.99989700,  err =   0.00010300
N = 262144, true =   9.00000000,  act =   8.99994850,  err =   0.00005150
N = 524288, true =   9.00000000,  act =   8.99997425,  err =   0.00002575
N = 1048576, true =   9.00000000,  act =   8.99998713,  err =   0.00001287
```

- Error decreases as 1/N
- Error decreases as h

Similar method: Midpoint rule

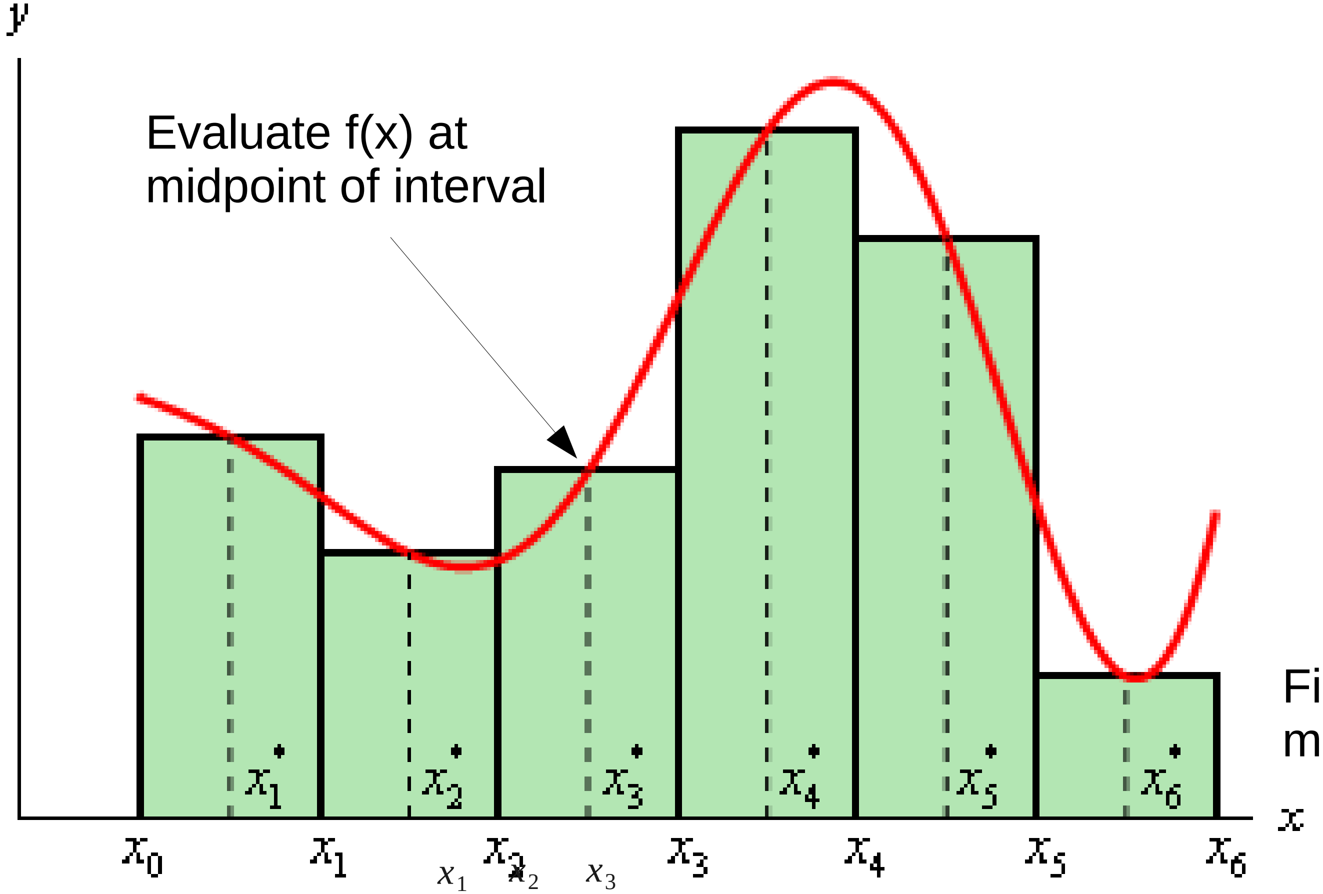
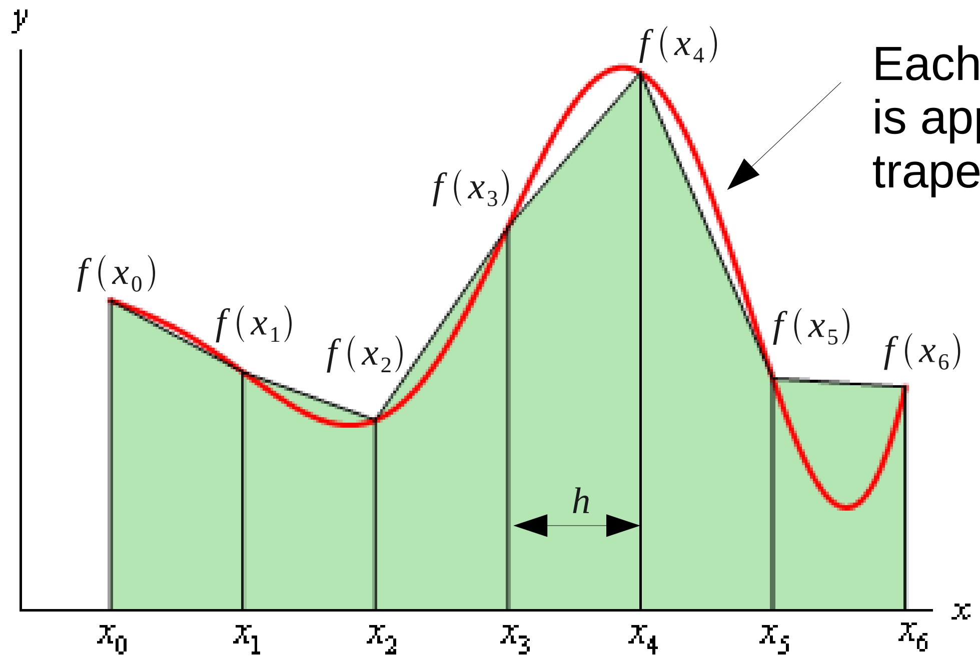


Figure from math.lamar.edu

$$\int_a^b dx\, f(x) = h \sum_{n=0}^{N-1} f(x_n^*) \qquad h = \frac{b-a}{N} \qquad x_n^* = a + n h + \frac{h}{2}$$

- Error properties similar to endpoint rule.

Elementary method: Trapezoidal Rule



Each interval's area is approximated as a trapezoid.

$$\text{area} = h\frac{f(x_{n+1})+f(x_n)}{2}$$

Figure from math.lamar.edu

- Sum up all trapezoids to get total area

$$\int_a^b dx\, f(x) = \frac{h}{2}\sum_{n=0}^{N-1}\left(f(x_{n+1})+f(x_n)\right) \qquad h=\frac{b-a}{N} \qquad x_n = a+nh$$

$$= \frac{h}{2}\left(f(x_1)+2f(x_2)+2f(x_3)+\cdots+f(x_N)\right)$$

Trapezoidal rule implementation

```
function y = trapezoid(f, a, b, n)
  % This function implements the trapezoidal rule.  Integration
  % is performed over n points on the interval a <= x < b

  h = (b-a)/n;      % Step size
  x = a:h:b;    % Sample x values
  s = f(x);
  y = h*( s(1) + 2*sum( s(2:(end-1)) ) + s(end) )/2;
end
```

- Test using same test wrapper as for endpoint rule.

Test results

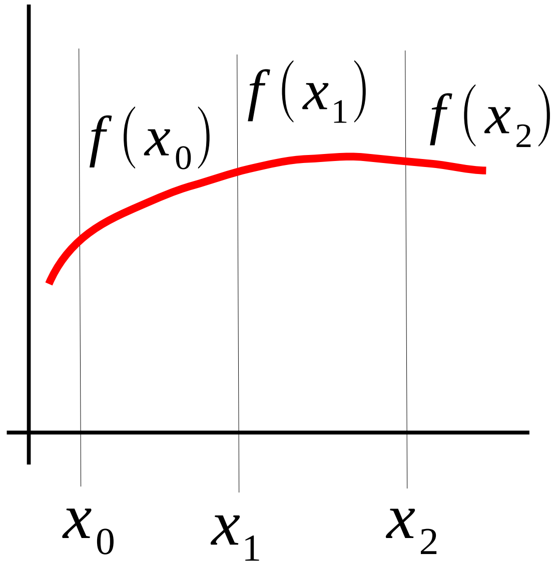
```
octave:7> test_trapezoid
N = 2, true =   9.00000000,  act =  10.12500000,  err =   1.12500000
N = 4, true =   9.00000000,  act =   9.28125000,  err =   0.28125000
N = 8, true =   9.00000000,  act =   9.07031250,  err =   0.07031250
N = 16, true =   9.00000000,  act =   9.01757812,  err =   0.01757812
N = 32, true =   9.00000000,  act =   9.00439453,  err =   0.00439453
N = 64, true =   9.00000000,  act =   9.00109863,  err =   0.00109863
N = 128, true =   9.00000000,  act =   9.00027466,  err =   0.00027466
N = 256, true =   9.00000000,  act =   9.00006866,  err =   0.00006866
N = 512, true =   9.00000000,  act =   9.00001717,  err =   0.00001717
N = 1024, true =   9.00000000,  act =   9.00000429,  err =   0.00000429
N = 2048, true =   9.00000000,  act =   9.00000107,  err =   0.00000107
N = 4096, true =   9.00000000,  act =   9.00000027,  err =   0.00000027
N = 8192, true =   9.00000000,  act =   9.00000007,  err =   0.00000007
N = 16384, true =   9.00000000,  act =   9.00000002,  err =   0.00000002
N = 32768, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 65536, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 131072, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 262144, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 524288, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 1048576, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
```

- Error decreases as $1/N^2$
- MATLAB: trapz

Simpson's rule

- Consider only one subinterval

$$\int_{x_0}^{x_2} dx\, f(x) = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right]$$

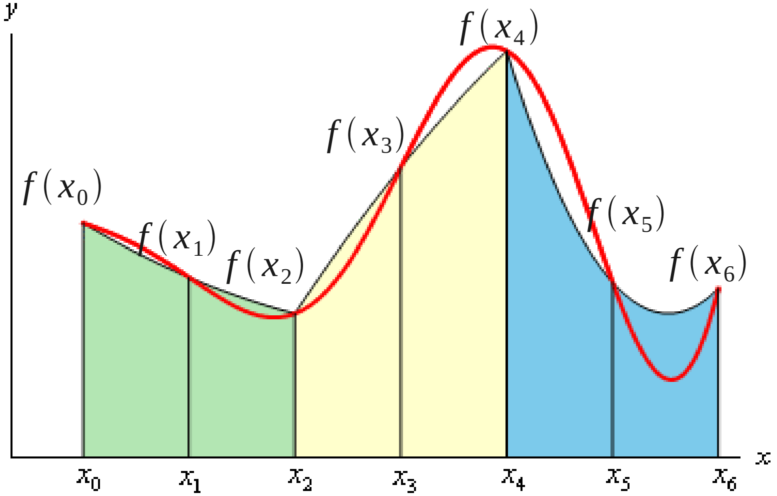


- Over many subintervals

$$\int_{x_0}^{x_2} dx\, f(x) = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] +$$

$$\frac{h}{3}\left[f(x_2) + 4f(x_3) + f(x_4)\right] +$$

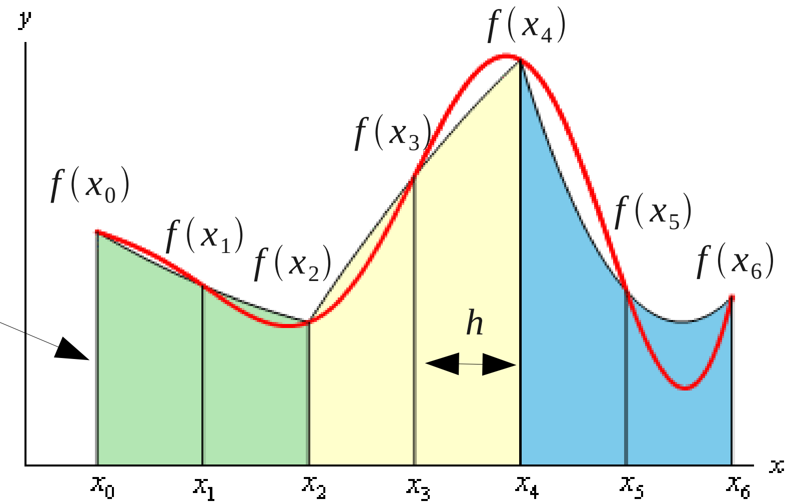
$$\frac{h}{3}\left[f(x_4) + 4f(x_5) + f(x_6)\right] + \cdots$$



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

Simpson's composite rule

$$\int_{x_0}^{x_2} dx\, f(x) = \frac{h}{3}\left[f(x_0)+4f(x_1)+f(x_2)\right]$$



even

odd

$$\int_a^b dx\, f(x) = \frac{h}{3}\left[f(x_0)+2\sum_j f(x_{2j})+4\sum_j f(x_{2j-1})+f(x_n)\right]$$

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

$$h=\frac{b-a}{N-1} \qquad x_n = a+nh$$

- Note: choose number of points N odd.

Simpson's Rule Implementation

```
function y = simpsons_rule(f, a, b, n)
  % This function implements Simpson's rule.  Integration
  % is performed over n points on the interval a <= x < b

  h = (b-a)/n;     % Step size

  % Compute sample points.
  x0 = a;
  x2j   = (a+2*h):2*h:(b-h);
  x2jm1 = (a+h):2*h:(b-h);
  xn = b;

  % Compute partial sums.
  s0 = f(x0);
  s2j = 2*sum(f(x2j));
  s2jm1 = 4*sum(f(x2jm1));
  sn = f(xn);

  % Compute full sum.
  y = h*(s0 + s2j + s2jm1 + sn)/3;
end
```

- Can you guess the convergence rate for x^2?

Test results

```
octave:3> test_simpsons_rule
N = 2, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 4, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 8, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 16, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 32, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 64, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 128, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 256, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 512, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 1024, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 2048, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 4096, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 8192, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 16384, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 32768, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 65536, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 131072, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 262144, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 524288, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
N = 1048576, true =   9.00000000,  act =   9.00000000,  err =   0.00000000
```

- Simpson's Rule is exact for quadratic (and cubic).

In general, these are called "Newton-Cotes" methods

- Series of formulas for integration.
- The formulas fit a series of Lagrange polynomials to the curve (Lagrange interpolation) and then report the integral of all the polynomials.
- They assume uniform step size *h*.
- Matlab: quad() or integral(). Uses adaptive Simpson's rule.

Newton-Cotes methods

Formula inside one super-interval.

Order	Name	Formula	Error
1	Trapezoidal method	$\frac{h}{2}[f_1+f_2]$	$O\left(h^3 f^{(2)}(x)\right)$
2	Simpson's rule	$\frac{h}{3}[f_1+4f_2+f_3]$	$O\left(h^5 f^{(4)}(x)\right)$
3	Simpson's 3/8 rule	$\frac{3}{8}h[f_1+3f_2+3f_3+f_4]$	$O\left(h^5 f^{(4)}(x)\right)$
4	Boole's method	$\frac{2}{45}h[7f_1+32f_2+12f_3+32f_4+7f_5]$	$O\left(h^7 f^{(6)}(x)\right)$

Newton-Cotes

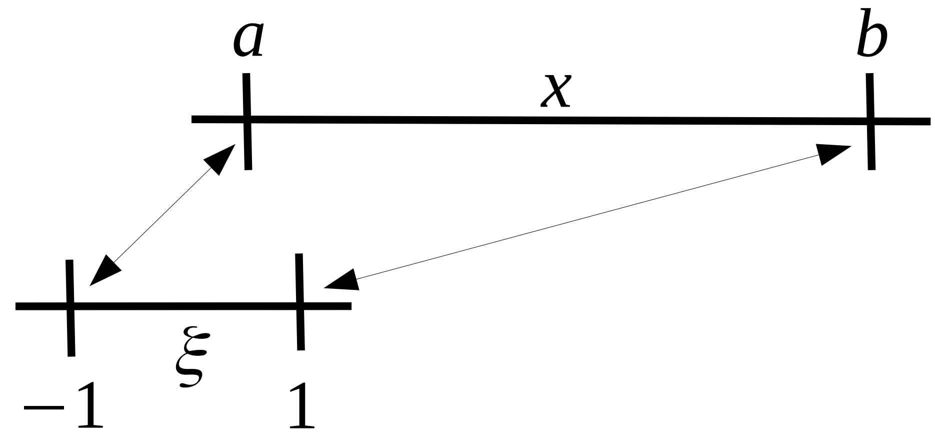
- Based on integrating sections of curve fit by Lagrange interpolation polynomial of order N.
  - Do trapezoidal method on blackboard.
- Order is degree of interpolating polynomial.
  - Recall: N points -> polynomial degree N-1.
- When order N is even, the computed result is exact for polynomials up to degree N+1.
- When order N is odd, the computed result is exact for polynomials up to degree N.

Better method: Gaussian Quadrature

- Consider integration over finite interval [-1, 1].
- Relax restriction to use uniform step size.
- By choosing the sample points you get a higher-degree polynomial fit for the same number of sample points.
  - Remember doing this for interpolation?
- Results exact for polynomials of order 2N-1.

My function lives on [a,b], Gauss quadrature works on [-1,1]

- What to do?
- Use linear map ...



$$x = s\,\xi + t$$

slope

offset

- Now insert info about end points and get coeffs.

$$\boxed{\begin{aligned} a &= -s + t \\ b &= s + t \end{aligned}} \Rightarrow \boxed{\begin{aligned} t &= (b+a)/2 \\ s &= (b-a)/2 \end{aligned}} \Rightarrow \boxed{\begin{aligned} x &= s\,\xi + t \\ \xi &= \frac{x-t}{s} \end{aligned}}$$

You can go back and forth

Basic Gaussian Quadrature

- Approximate integral using weighted sum:

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

- Sample points: x_i
  - Non-uniform, chosen carefully.
- Summation weights: w_i
- Note you need to shift your integral's limits from [*a*, *b*] to [-1, 1].
- But what x_i and w_i to choose?

Consider integrating powers

- General Gauss quadrature formula

$$\int_{-1}^{1} dx\, f(x) \approx a_0 f(x_0) + a_1 f(x_1) + a_2 f(x_2) + a_3 f(x_3) + \cdots$$

$x_i =$ Sample point(s)

$a_i =$ Weights

- Concept:
  - Interpolate on N points.
  - That gives 2N free variables we can adjust: a_i, x_i.
  - Find a_i, x_i to make integration exact for all powers up to 2N-1.

Example

- Example: choose n = 2, and see what makes the 2nd order Gauss quadrature formula **exact** for $f(x) = 1, x, x^2, x^3$

$$\int_{-1}^{1} dx\, f(x) \approx a_0 f(x_0) + a_1 f(x_1)$$

- 4 unknowns: a_0, x_0, a_1, x_1
- 4 equations – 1 each for $1, x, x^2, x^3$.

Goal: Find a_i and x_i to make Gauss quadrature formula exact for powers 0, 1, 2, 3

$$\int_{-1}^{1} dx\, 1 = 2 = a_1 + a_2 \qquad (1)$$

$$\int_{-1}^{1} dx\, x = 0 = a_1 x_1 + a_2 x_2 \qquad (2)$$

$$\int_{-1}^{1} dx\, x^2 = \frac{2}{3} = a_1 x_1^2 + a_2 x_2^2 \qquad (3)$$

$$\int_{-1}^{1} dx\, x^3 = 0 = a_1 x_1^3 + a_2 x_2^3 \qquad (4)$$

- By symmetry, $a_1 = a_2 = a$.
- From (1), $2a = 2 \Rightarrow a_1 = a_2 = 1$.
- Therefore, from (2), $x_1 = -x_2 = x$

From last slide:

$$\int_{-1}^{1} dx\, x^2 = \frac{2}{3} = a_1 x_1^2 + a_2 x_2^2 \qquad (3)$$

$a_1 = a_2 = 1$

$x_1 = -x_2 = x$

- Solve for x: $x = \pm \frac{1}{\sqrt{3}}$

- Therefore, the following formula is exact for all polynomials up to (including) degree 3:

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

- This is 2nd order Gaussian quadrature. Highly accurate for integrating any function.

Gauss quadrature

Number of sample points	Sample points x_i	Weights a_i	Exact to degree
1	0	2	1
2	$-\frac{1}{\sqrt{3}},+\frac{1}{\sqrt{3}}$	1	3
3	$-\sqrt{\frac{3}{5}},0,+\sqrt{\frac{3}{5}}$	$\frac{5}{9},\frac{8}{9},\frac{5}{9}$	5

- General rule: N points => exact for polys up to degree 2N-1.
- Recall Newton-Cotes was exact for polys up to degree N.
- But where to get sample pts and weights for arbitrary degree?

Detour: Legendre Polynomials

- Consider expanding a function f(x) on the interval [-1, 1].

$$f(x) = \sum_{n=0}^{\infty} c_n \phi_n(x)$$

- Many ways to do it – Fourier series, Taylor's series, etc. Consider Taylor's series:

$$f(x) = \sum_{n=0}^{\infty} c_n x^n = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots$$

- We can consider the powers of x to be a basis set of functions useful for creating a series expansion of f(x)

Taylor's series expansion

- Take basis set for expansion as powers of x:

$$f(x)=\sum_{n=0}^{\infty} c_n x^n = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots$$

- Problem with Taylor's series expansion: Basis set is not orthogonal.
- What if we applied the Gram-Schmidt procedure to the basis vectors $[1, x, x^{2,} x^{3,} x^{4,} \cdots]$?
- Answer: Legendre polynomials.

Legendre Polynomials

- Legendre polynomials $P_n(x)$: Another set of orthogonal polynomials.

$$\int_{-1}^{1} dx\, P_n(x)\, P_m(x) = 0 \quad \text{if} \quad n \neq m$$

- Useful for series expansions of functions on interval [-1, 1]:

$$f(x) = \sum_{n=0}^{\infty} c_n P_n(x)\ , \quad \text{where } -1 \leq x \leq 1$$

- Legendre polynomials show up in quantum mechanics, electromagnetism, and other places where wave equations are solved in systems with spherical symmetry.

Some Legendre Polynomials

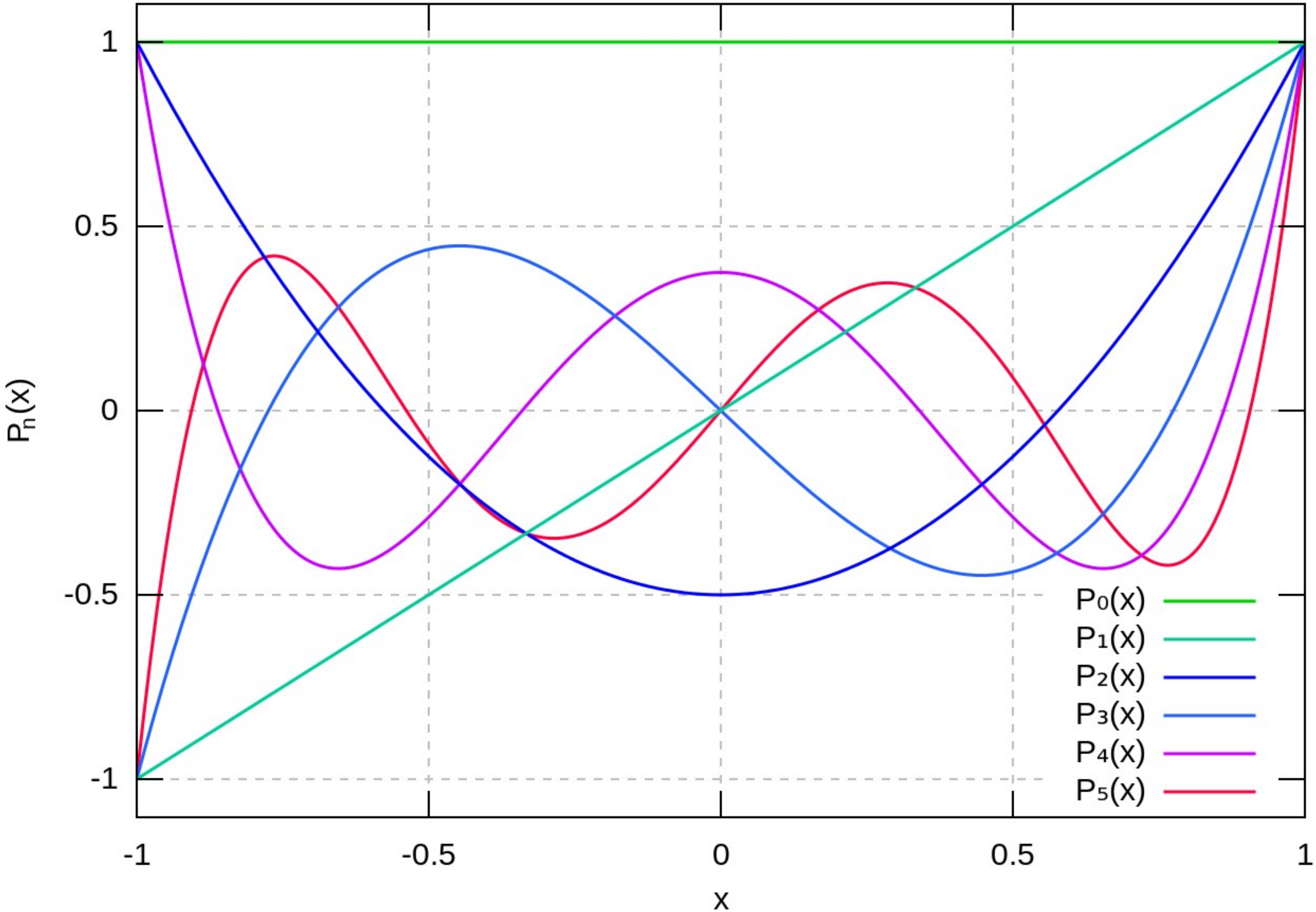
$n=0 \qquad P_0(x)=1$

$n=1 \qquad P_1(x)=x$

$n=2 \qquad P_2(x)=\frac{1}{2}(3x^2-1)$

$n=3 \qquad P_3(x)=\frac{1}{2}(5x^3-3x)$

Legendre Polynomials



Source: Wikipedia

Back to Gaussian Quadrature: Connection to Legendre Polynomials

- For *n* point Gaussian quadrature, sample points x_i are roots of Legendre polynomial $P_n(x)$.

$$n=1 \qquad P_1(x)=x$$

$$n=2 \qquad P_2(x)=\frac{1}{2}\left(3x^2-1\right)$$

$$n=3 \qquad P_3(x)=\frac{1}{2}\left(5x^3-3x\right)$$

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

Weights

Sample points

- And weights are given by:

$$w_i=\frac{2}{(1-x_i^2)[P'_n(x_i)]^2}$$

Gaussian quadrature points and weights (numeric)

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

Gaussian Quadrature Weights and Abscissae – Mozilla Firefox

File Edit View History Bookmarks Tools Help

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

- abscissae (607KB, easily converted to not-PHP)
- weights (615KB, easily converted to not-PHP)

Weights and Abscissae Tables for n=2 to n=64

n = 2

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64

i	weight - w_i	abscissa - x_i
1	1.0000000000000000	-0.5773502691896257
2	1.0000000000000000	0.5773502691896257

n = 3

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64

i	weight - w_i	abscissa - x_i
1	0.8888888888888888	0.0000000000000000
2	0.5555555555555556	-0.7745966692414834
3	0.5555555555555556	0.7745966692414834

n = 4

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64

i	weight - w_i	abscissa - x_i
1	0.6521451548625461	-0.3399810435848563
2	0.6521451548625461	0.3399810435848563
3	0.3478548451374538	-0.8611363115940526
4	0.3478548451374538	0.8611363115940526

n = 5

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64

i	weight - w_i	abscissa - x_i
1	0.5688888888888889	0.0000000000000000
2	0.4786286704993665	-0.5384693101056831
3	0.4786286704993665	0.5384693101056831
4	0.2369268850561891	-0.9061798459386640

Gauss-Legendre quadrature demo

```
function ret = gauss_quadrature(f, a, b, N)

  % Sample points and weights on [-1, 1] interval
  [xi, omega]=lgwt(N, -1, 1);  % lqwt is from Mathworks file share

  % Sample pts on [a, b] interval
  x = ((b-a)/2).*xi + (b+a)/2;  % x is a vector.

  % Weights
  w = ((b-a)/2)*omega;   % w is a vector.

  % Now sample fcn at Gauss points
  y = f(x);   % f(x) must be constructed to return a vector

  % Do sum to compute integral
  ret = dot(w,y);

end
```

Class11/CompareMethods

Gauss-Legendre quadrature demo

```
>> test_gauss_quadrature
---------------------------------------
Testing integral of x^2 ...
N = 3, y = 21.333333333333002, diff = -3.197442e-14 ... Passed!
N = 7, y = 21.333333333333357, diff = 3.552714e-15 ... Passed!
N = 13, y = 21.333333333333215, diff = -1.065814e-14 ... Passed!
---------------------------------------
Testing integral of cos(x) ...
N = 7, y = 1.000000000000002, diff = 2.220446e-16 ... Passed!
N = 13, y = 0.999999999999996, diff = -4.440892e-16 ... Passed!
N = 21, y = 1.000000000000000, diff = 0.000000e+00 ... Passed!
N = 35, y = 0.999999999999999, diff = -1.110223e-16 ... Passed!
N = 67, y = 0.999999999999992, diff = -7.771561e-16 ... Passed!
N = 99, y = 1.000000000000002, diff = 2.220446e-16 ... Passed!
```

Clenshaw-Curtis Quadrature

- Similar to Gauss-Legendre, you must use prescribed sample points and weights.

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

- Sample points

$$x_i = -\cos(k\pi/N) \qquad k = 0, 1, \cdots, N$$

- Weights

$$w_i = \begin{cases} \dfrac{1}{N^2 - 1} & k = 0 \text{ or } k = N \\ \dfrac{4}{N} \displaystyle\sum_{j=0}^{N/2} \dfrac{\cos(2\pi j k/N)}{\gamma_j (1 - 4j^2)} & k = 1, \cdots N-1 \end{cases}$$

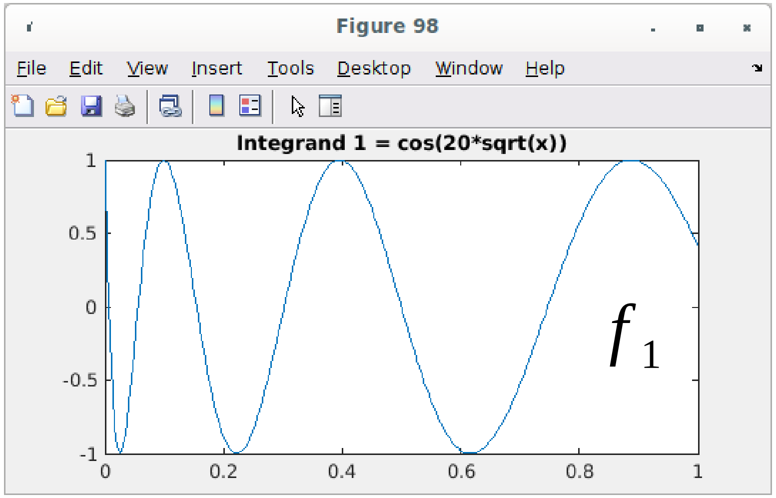
$$\gamma_j = \begin{cases} 2 \text{ for } j = 0 \text{ or } j = N/2 \\ 1 \text{ for } j = 1, 2, \cdots N/2 - 1 \end{cases}$$

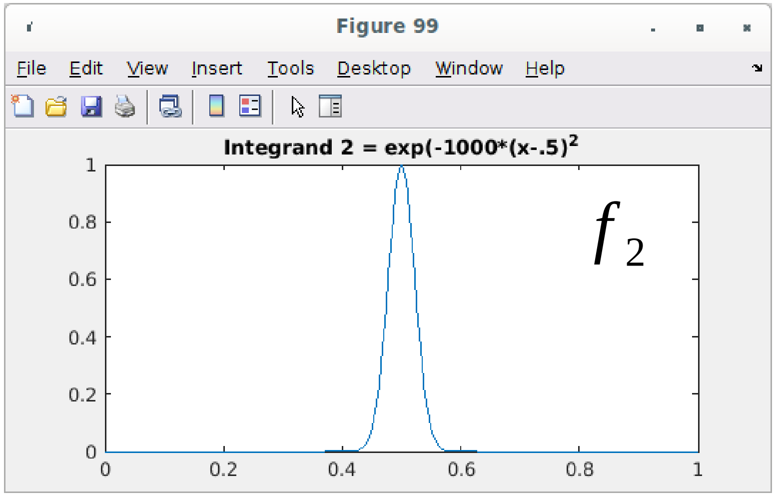
Comparison of methods

- Comparison:
  - Simpson's 1/3 rule
  - Gauss-Legendre
  - Clenshaw-Curtis
- Two difficult test functions:

$$I_1 = \int_0^1 f_1(x)\,dx \qquad f_1(x) = \cos\left(20\sqrt{x}\right)$$

$$I_2 = \int_0^1 f_2(x)\,dx \qquad f_2(x) = e^{-1000(x-1/2)^2}$$

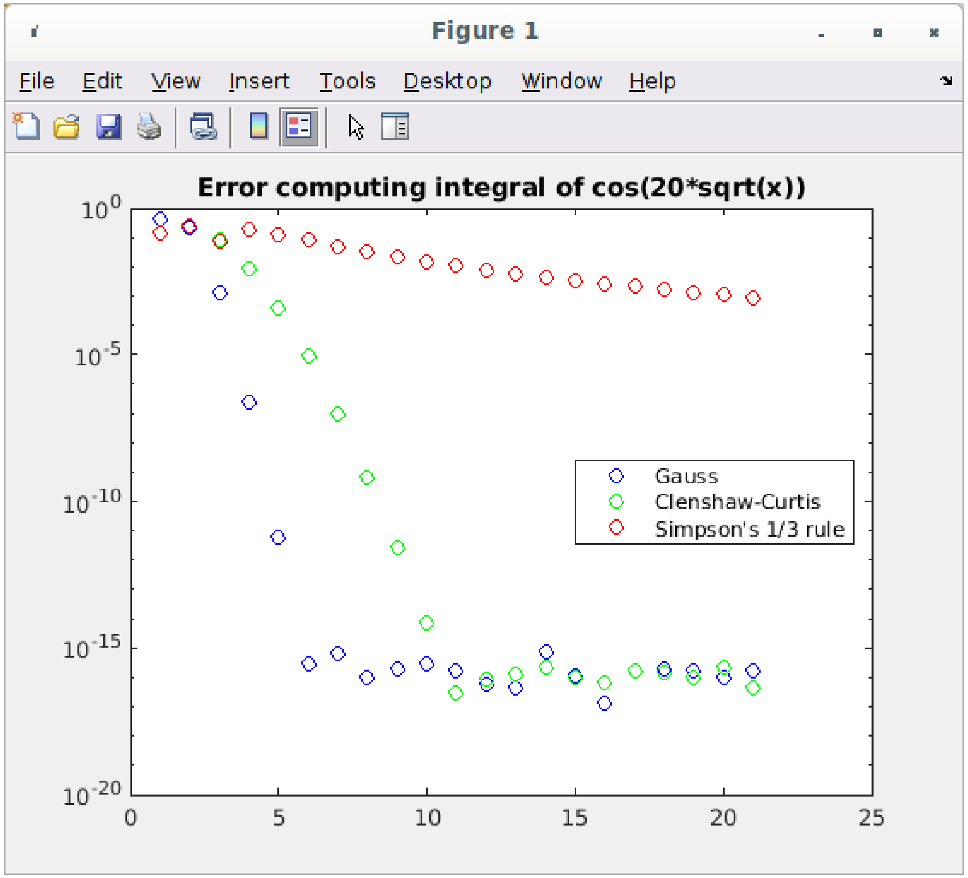


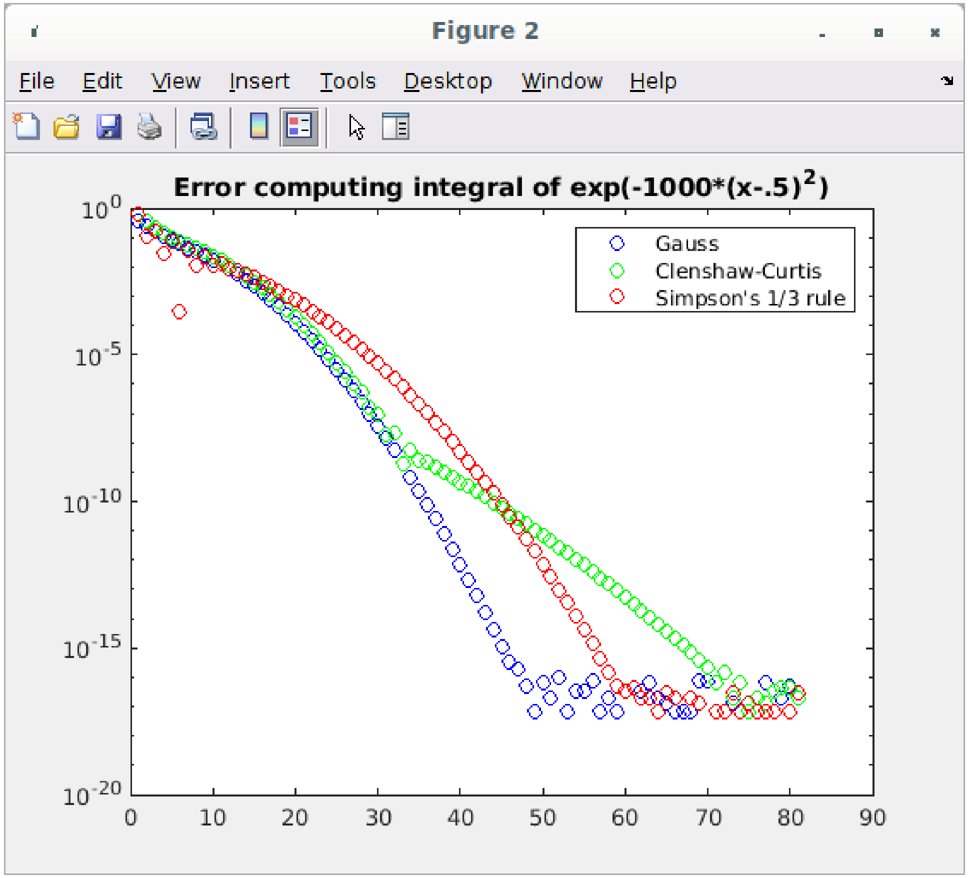


- Examine error for increasing N.

Comparison of methods

- Plots show error vs. increasing number of sample points N.





Reference: “Improving the Accuracy of the Trapezoidal Rule”, Bengt Fornberg, SIAM Rev., 63(1), 167–180.

Different integrands, different polynomials

- Gauss-Legendre quadrature – simplest method which we just looked at. Integrals of form:

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

- Gauss-Chebyshev quadrature. Integrals of form:

$$\int_{-1}^{1} dx \frac{f(x)}{\sqrt{1-x^2}} \approx \sum_{i=1}^{N} w_i f(x_i)$$

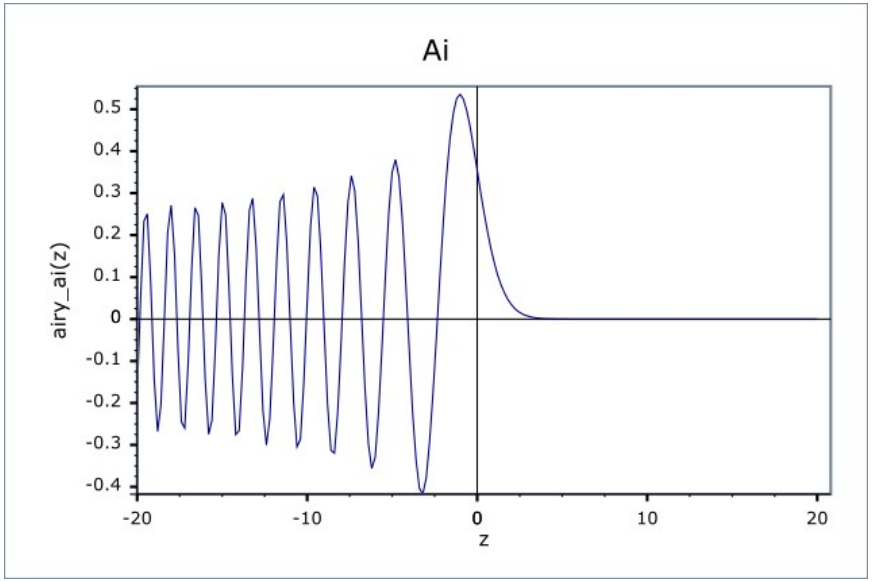
Homework problem

- Gauss-Hermite quadrature. Integrals of form:

$$\int_{-\infty}^{\infty} dx\, f(x) e^{-x^2} \approx \sum_{i=1}^{N} w_i f(x_i)$$

- Many others...
- Many not in Matlab. You must roll your own.

Next topic: Adaptive quadrature



$$\frac{d^2 y}{dx^2} = xy$$

$$y = Ai(x)$$

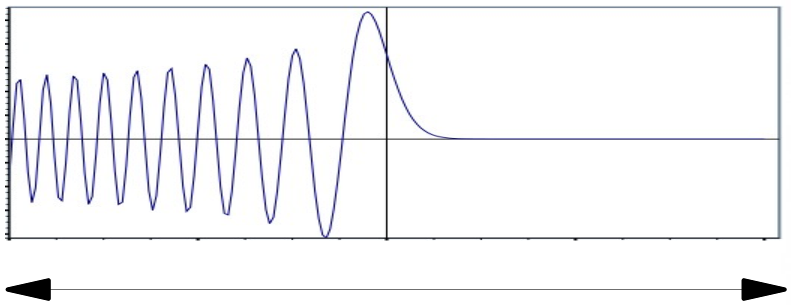
- Consider integrating *Ai(x)* from -20 to 20.
  - Mesh points on left must be closely spaced.
  - Mesh points on right don't need close spacing.
- Integration needs different *h* values over different intervals for low error.
- Matlab quad: “Adaptive Simpson Quadrature”.

Adaptive quadrature algorithm

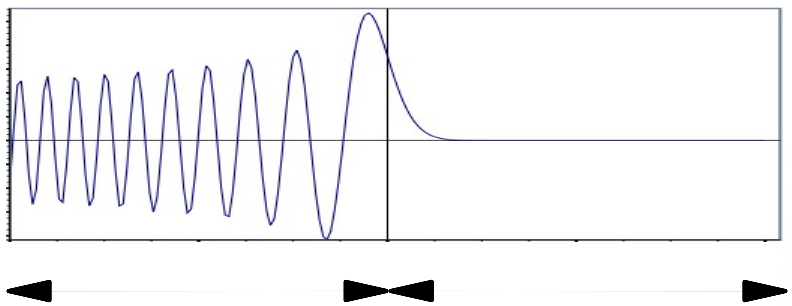
- Get error estimate for integral over full interval
  - Get error estimate by doing integral twice: once with h, then with $h/2$.
- If error estimate too high, then sub-divide interval.
- Do integral over each sub-interval separately, and get error estimates over sub-intervals.
- If error estimate too high in one or more sub-intervals, divide them again and recurse.

Example

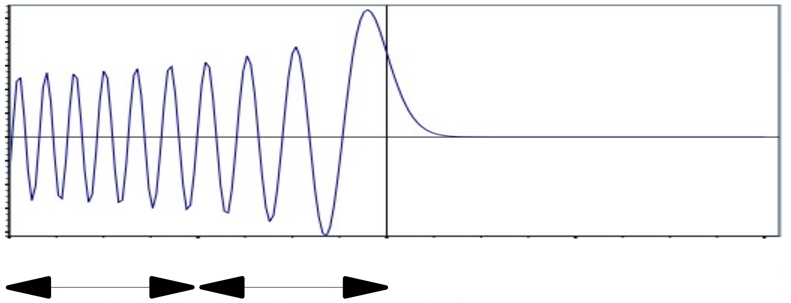
- First attempt
  - Do quad twice with different h to estimate error



- Second attempt
  - Do quad twice with different h to estimate error



- Third attempt
  - Do quad twice with different h to estimate error



Library you should know: QUADPACK

- Another public-domain numerical analysis package available on www.netlib.org.
- Numerical integration library (Fortran).
- Bindings to Octave, NumPy/SciPy, etc.
- Clones for C/C++, etc.
- Similar integration routines for C++ also available in GSL (Gnu Scientific Library).

Numerical Integration: Summary

- 1D integration:
  - Endpoint & trapezoidal rules
  - Newton-Cotes (evenly spaced x axis points)
  - Gaussian-Legendre quadrature (choose sample points and weights using Legendre polynomials).
  - Clenshaw-Curtis
- Comparison of methods