Quadrature

Fabio Cannizzo

Definition of Integral

Definition

$$I = \int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=0}^{n-1} f(x_i)\Delta x, \quad x_i \in [a + i\Delta x, a + (i+1)\Delta x], \quad \Delta x = \frac{b-a}{n}$$

symbolically computed via the fundemantal theorem of calculus

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

where $F(x)$ is a fuction such that

$$\frac{dF(x)}{dx} = f(x)$$

Why Numerical

- It may just be more convenient
- We may not know the function f(x) in closed form
- In some cases a primitive F(x) in closed form may not exist. Example:

$$\int e^{-x^2} dx$$

Polynomial Integration

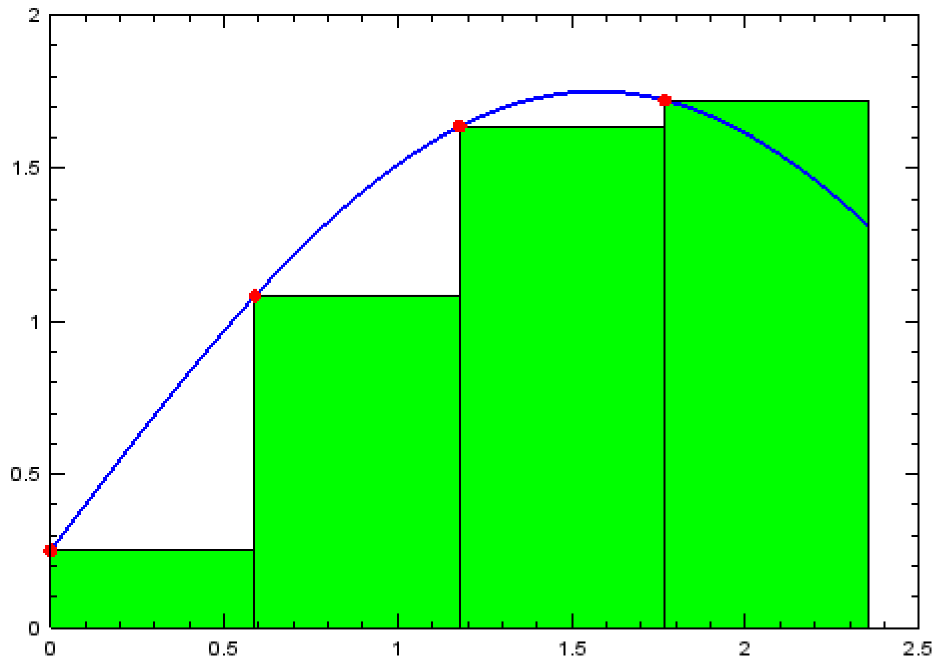
- A simple approach is to replace the function being integrated with the approximating polynomial of degree n

$$\int_a^b f(x)dx \approx \int_a^b p^n(x)dx$$

- For greater accuracy, we could use high order polynomials, but they oscillate! Instead we break the integral domain is *m* sub-intervals (**composite**) and use different low order polynomial on each sub-interval. If we have n+1 points:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} p_i^n(x)dx$$

Rectangle Rule: Example



$$\int_0^{\frac{3}{4}\pi} \frac{1}{4} + \frac{3}{2}\sin(x)dx = \frac{3}{16}\pi + \left(\frac{\sqrt{2}}{2} + 1\right)\frac{3}{2} \approx 3.1497$$

```
>> RectangleDemo(@myfun, 0, 0.75*pi(), 4)
ans =  2.7628
```

Rectangle Rule

- We simply use the definition of integral
- We can choose the position of the x in the intervals $[x_i, x_{i+1}]$ as $x = x_i$

$$I = \int_a^b f(x)dx \approx f(a)(b-a) \qquad \text{(SINGLE INTERVAL)}$$

$$I = \int_a^b f(x)dx \approx h \sum_{i=0}^{m-1} f(a+ih), \quad h = \frac{b-a}{m} \qquad \text{(COMPOSITE INTERVAL)}$$

- Note that this is the same as constructing an interpolation **piecewise constant**, then compute its integral

Rectangle Rule: Source Code

```
function I = Rectangle( f, a, b, m )

   h = (b-a)/m;

   yi = f( a+[0:m-1]*h );

   I=h*sum(yi);
```

Rectangle Rule: Convergence

On one single interval [a,b], the error is proportional to

$E = O(b-a)^2$

Subdiving [a,b] in m sub-intervals (composite), each of size $h = \dfrac{b-a}{m}$

on each sub-interval we have an error $E_i = O(h^2)$

and the total error is proportional to m times the error on a single sub-interval

$E = O(mE_i) = O(mh^2) = O(h) = O(\dfrac{1}{m})$

- This means if we double the number of sub-intervals *m*, i.e. if we half the size of the sub-interval *h*, the error reduce by a factor 2

Rectangle Rule Summary

- Very simple to implement
- Robust
- Computation cost: n evaluations of f(x)
- Composite convergence: $O(m^{-1})$
- If f(x) is increasing, the rectangle rule underestimates, otherwise it overestimates

Mid-Point Rule

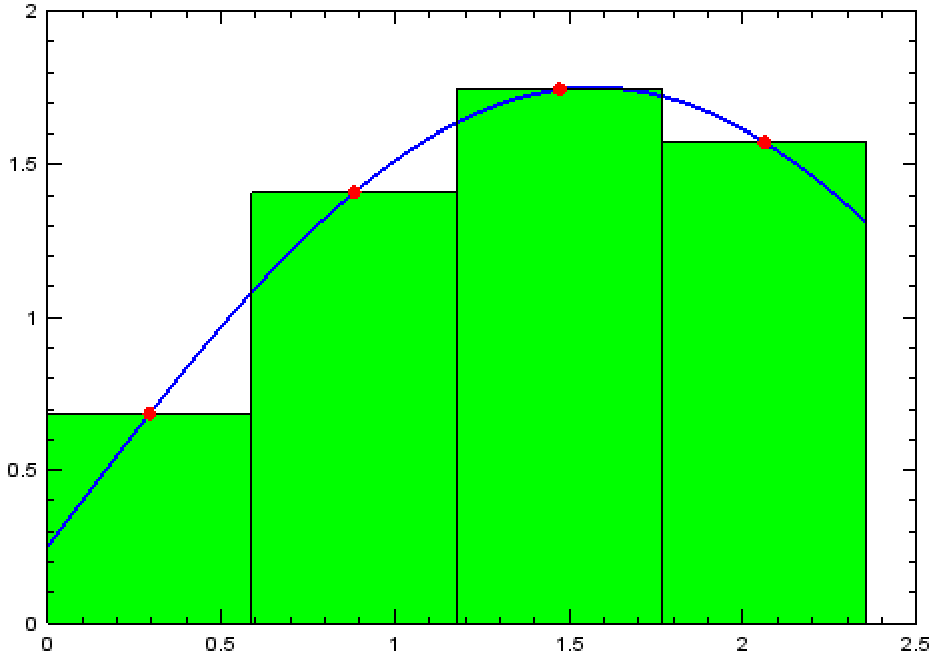
- Similar to the rectangle rule, but we take the mid point of the interval $[x_i, x_{i+1}]$

$$I = \int_a^b f(x)dx \approx f\left(\frac{a+b}{2}\right)(b-a) \qquad \text{(SINGLE INTERVAL)}$$

$$I = \int_a^b f(x)dx \approx h\sum_{i=0}^{m-1} f\left(a + \left(i + \frac{1}{2}\right)h\right), \quad h = \frac{b-a}{m} \qquad \text{(COMPOSITE INTERVAL)}$$

- Note that this is the same as constructing an interpolation **piecewise constant**, then compute its integral

Mid-Point Rule: Example



$$\int_0^{\frac{3}{4}\pi} \frac{1}{4} + \frac{3}{2}\sin(x)dx = \frac{3}{16}\pi + \left(\frac{\sqrt{2}}{2} + 1\right)\frac{3}{2} \approx 3.1497$$

```
>> RectangleDemo(@myfun, 0, 0.75*pi(), 4)
ans =  3.1871
```

Mid-Point Rule: Source Code

```
function I = MidPoint( f, a, b, m )

   h = (b-a)/m;

   yi = f( a+([0:m-1]+0.5)*h );

   I=h*sum(yi);
```

Mid-Point Rule: Convergence

On one **single** interval [a,b]

$$E = O((b-a)^3)$$

Composite interval:
If we break [a,b] in equally spaced sub-intervals of size h=(b-a)/m

$$E = O(h^2) = O(m^{-2})$$

This means if we double m, we expect E to reduce by a factor 4

Mid-Point Rule Summary

- Very simple to implement
- Robust
- Computation cost: n evaluations of f(x)
- Composite convergence: $O(m^{-2})$
- Superior to the rectangle rule at roughly the same cost

Trapezoid Rule

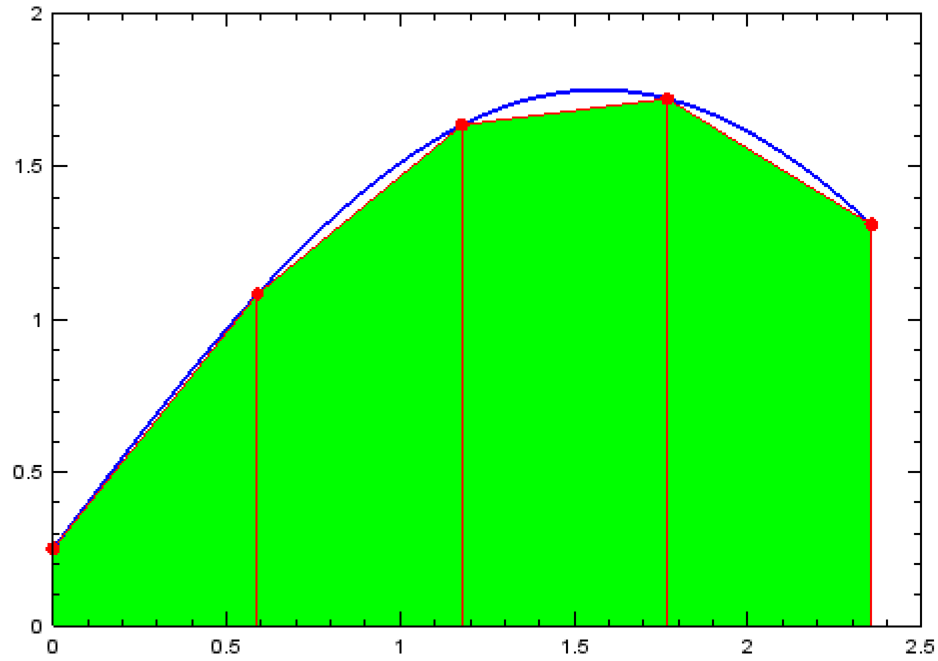
- Instead of choosing one point in $[x_i, x_{i+1}]$, we can use the average of $f(x_i)$ and $f(x_{i+1})$

$$I = \int_a^b f(x)dx \approx \left[\frac{1}{2}f(a) + \frac{1}{2}f(b)\right](b-a) \qquad \text{(SINGLE INTERVAL)}$$

$$I = \int_a^b f(x)dx \approx h\sum_{i=0}^{m-1} \frac{1}{2}f(x_i) + \frac{1}{2}f(x_{i+1}), \quad h = \frac{b-a}{m} \qquad \text{(COMPOSITE INTERVAL)}$$

- Note that this is the same as constructing an interpolation **piecewise linear**, then compute its integral
- In other words, we are summing the areas of the trapezoids delimited by $f(x_i)$ and $f(x_{i+1})$

Trapezoid Rule: Example



$$\int_0^{\frac{3}{4}\pi} \frac{1}{4} + \frac{3}{2}\sin(x)dx = \frac{3}{16}\pi + \left(\frac{\sqrt{2}}{2}+1\right)\frac{3}{2} \approx 3.1497$$

```
>> TrapezoidDemo(@myfun, 0, 0.75*pi(), 4)
ans =  3.0752
```

Trapezoid Rule: Source Code

- We transform the formula a little bit to reduce the number of algebraic operations

$$\int_a^b f(x)dx \approx h \sum_{i=0}^{m-1} \frac{1}{2} f(x_i) + \frac{1}{2} f(x_{i+1}) = h \left[\frac{f(x_0) + f(x_m)}{2} + \sum_{i=1}^{m-1} f(x_i) \right]$$

```
function I = Trapezoid( f, a, b, m )

   h = (b-a)/m;

   yi = f( a+[0:m]*h );

   I = h * ( 0.5 *(yi(1)+yi(m+1)) + sum(yi(2:m)) );
```

Trapezoid Rule: Convergence

On one **single** interval [a,b]

$$E = O((b-a)^3)$$

Composite interval:
If we break [a,b] in equally spaced sub-intervals of size h=(b-a)/m

$$E = O(h^2) = O(m^{-2})$$

This means if we double m, we expect E to reduce by a factor 4

Trapezoid Rule Summary

- Very simple to implement
- Robust
- Computation cost: n+1 evaluations of f(x)
- Composite convergence: $O(m^{-2})$
  - Same as the Mid-Point rule. This is not because the trapezoid rule is poor, it is because the Mid-Point rule, thanks to symmetry, does very well. The intuition is that the mid point is not that different from the average of the two points
- Error
  - if the integrand is concave up, the trapezoidal rule overestimates the true value.
  - Similarly, a concave-down function yields an underestimate.
  - If the interval of the integral being approximated includes an inflection point, the error is harder to identify

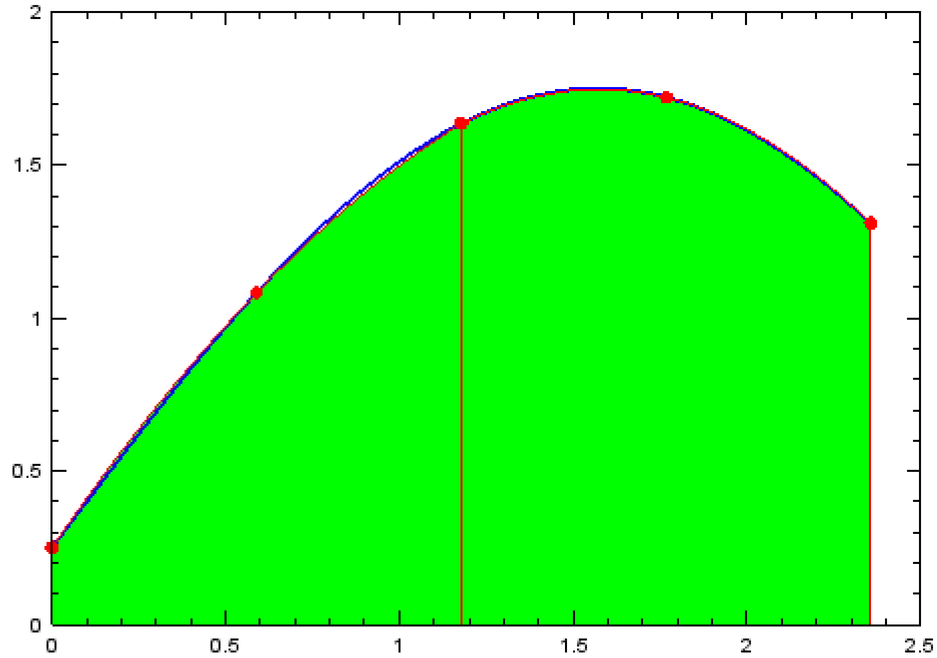
Simpson Rule

- Given an even number of intervals, on every pair of intervals we approximate the function with a parabola
- I.e. we are defining an interpolation scheme piecewise quadratic

$$I = \int_a^b f(x)dx \approx \left[\frac{1}{6}f(a) + \frac{4}{6}f\left(\frac{a+b}{2}\right) + \frac{1}{6}f(b)\right](b-a) \qquad \text{(SINGLE)}$$

$$I = \int_a^b f(x)dx \approx h\sum_{i=0}^{m-1} \frac{1}{6}f(x_i) + \frac{4}{6}f\left(\frac{x_i + x_{i+1}}{2}\right) + \frac{1}{6}f(x_{i+1}), \quad h = \frac{b-a}{m} \qquad \text{(COMPOSITE)}$$

Simpson Rule: Example



$$\int_0^{\frac{3}{4}\pi} \frac{1}{4} + \frac{3}{2}\sin(x)dx = \frac{3}{16}\pi + \left(\frac{\sqrt{2}}{2} + 1\right)\frac{3}{2} \approx 3.1497$$

```
>> SimpsonDemo(@myfun, 0, 0.75*pi(), 4)

ans =  3.1515
```

Simpson Rule: Source Code

We manipulate a bit the composite formula to reduce the number of arithmetic operations

$$I \approx h \sum_{i=0}^{m-1} \frac{1}{6} f(x_i) + \frac{4}{6} f\left(\frac{x_i + x_{i+1}}{2}\right) + \frac{1}{6} f(x_{i+1}), \quad h = \frac{b-a}{m}$$

introducing a new set of points $z_j = j\dfrac{h}{2} \implies x_i = z_{2i}$

$$I \approx \frac{h}{6} \sum_{j=0}^{2(m-1)} f(z_j) + 4f(z_{j+1}) + f(z_{j+2}) = \frac{h}{6}\left(f(z_0) + f(z_{2m}) + 2\sum_{j=1}^{m-1} f(z_{2j}) + 4\sum_{j=0}^{m-1} f(z_{2j+1})\right)$$

it is convenient to refer to the number of points, not intervals, so we define $n = 2m, \Delta = \dfrac{b-a}{n}$

$$I \approx \frac{\Delta}{3}\left(f(z_0) + f(z_n) + 2\sum_{j=1}^{n/2-1} f(z_{2j}) + 4\sum_{j=0}^{n/2-1} f(z_{2j+1})\right)$$

```
function I = Simpson( f, a, b, m )
   assert(mod(m,2)==0); # requires an even number of points
   h = (b-a)/m;
   yi = f( a+[0:m]*h );
   I=h*( yi(1)+yi(m+1) + 4*sum(yi(2:2:m)) + 2*sum(yi(3:2:m-1)) )/3;
```

Simpson Rule: Convergence

On one **single** interval [a,b]

$$E = O((b-a)^5)$$

Composite interval:
If we break [a,b] in equally spaced sub-intervals of size h=(b-a)/m

$$E = O(h^4) = O(m^{-4})$$

This means if we double m, we expect E to reduce by a factor 16

Simpson Rule Summary

- Very simple to implement
- Robust
- Computation cost: n+1 evaluations of f(x)
- Composite convergence: $O(m^{-4})$
- Simpson is exact for polynomial of order 3 or lower (error is $O(h^4)$)

Richardson Extrapolation

- We saw Richardson for finite differences. The same idea can be applied to integration to obtain an higher order scheme
- We know that Simpson rule is O(h^4)
- We use it twice, once with h and once with h/2. The Taylor expansion of the integral are

$$I = S(h) = ah^4 + O(h^5)$$

$$I = S(2h) + 16ah^4 + O(h^5)$$

we combine linearly the equations above to cancel the term inh^4

$$I = \frac{16S(h) - S(2h)}{15} + O(h^5)$$

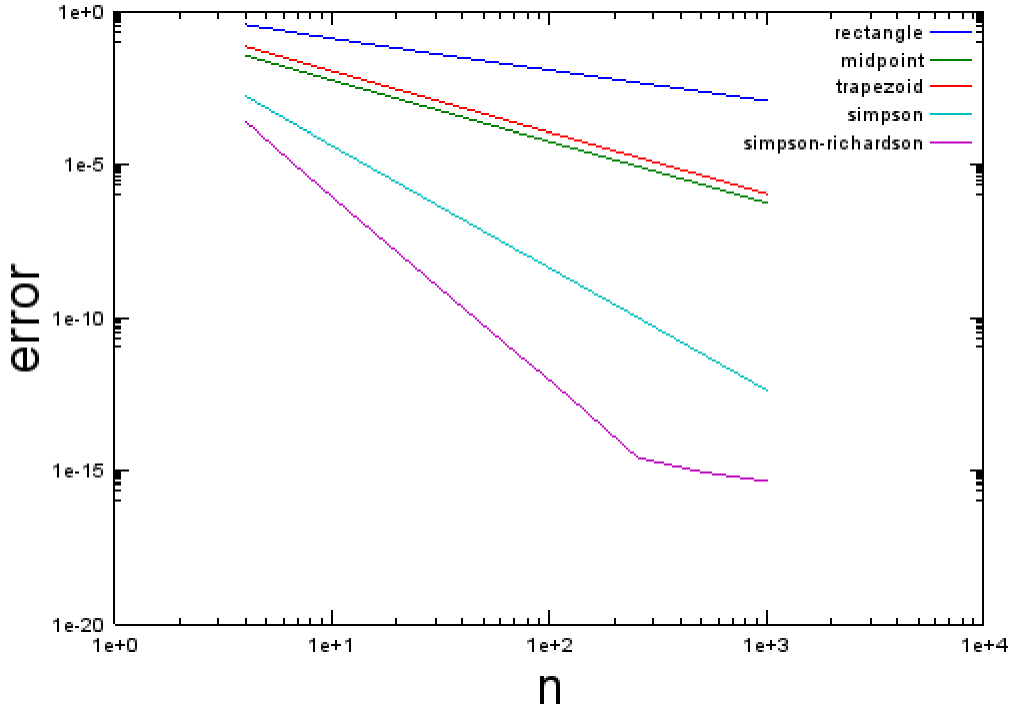
so we obtain an approximation of order $O(h^5)$

example:

```
>> s2=Simpson(@myfun, 0, 0.75*pi(), 2)
s2 =  3.1824
>> s4=Simpson(@myfun, 0, 0.75*pi(), 4)
s4 =  3.1515
>> (16*s4-s2)/15
ans =  3.1494
```

Note that if we did the 2 calculations in the same call, the points necessary to compute S(2h) are a subset of those necessary to compute S(h), hence there are no extra function evaluations with respect to just computing S(h)

Convergence Comparison



The weird scarlet segment connects just two points, the second of which is at machine precision

$$I = \int_0^{\frac{3}{4}\pi} \frac{1}{4} + \frac{3}{2}\sin(x)dx = \frac{3}{16}\pi + \left(\frac{\sqrt{2}}{2} + 1\right)\frac{3}{2} \approx 3.1497$$

```
>> ConvergenceTest( @myfun, 0, 0.75*pi(), I )
```

Newton Cotes Integration

- Newton Cotes integration means approximating the function f(x) in [a,b] with a polynomial p(x)

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx$$

- passing by a number of points equally spaced in [a,b]

$$\text{let} \quad h = \frac{b-a}{n}, \quad x = \{a, a+h, a+2h, \ldots, b\}$$

- The formula are said to be **closed** if the extreme points *a* and *b* are used, **open** if they are not
- If the formula is closed (*a* and *b* included) such polynomial can be written in Lagrange form as

$$p_n(x) = \sum_{i=0}^{n} l_i(x) f(x_i), \quad \text{where } l_i(x) = \frac{\prod_{j=0, j\neq i}^{n}(x - x_j)}{\prod_{j=0, j\neq i}^{n}(x_i - x_j)}, \quad \text{note } l_i(x_j) = \left\{\begin{smallmatrix} 1, & i=j \\ 0, & i\neq j \end{smallmatrix}\right.$$

- In the open case the indices go from 1 to n-1 and we obtain a polynomial of degree n-2

Newton Cotes Formulas

- Newton Cotes formulas can be simply obtained integrating the $l_i(x)$

$$\int_a^b f(x)dx \approx \int_a^b p_n(x)dx = \sum_{i=0}^{n} f(x_i)\int_a^b l_i(x)dx = \sum_{i=0}^{n} W_i f(x_i)dx, \quad \text{where } W_i = \int_a^b l_i(x)dx$$

- Changing the integration domain for $l_i(x)$ to [0,1], and distinguishing between open and close formula, we obtain

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{n} A_i f(x_i), \quad \text{where } A_i = \frac{\displaystyle\int_0^1 \prod_{j=0,j\neq i}^{n} (nx-j)\,dx}{\displaystyle\prod_{j=0,j\neq i}^{n} (i-j)} \qquad \text{CLOSED}$$

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=1}^{n-1} B_i f(x_i), \quad \text{where } B_i = \frac{\displaystyle\int_0^1 \prod_{j=1,j\neq i}^{n-1} (nx-j)\,dx}{\displaystyle\prod_{j=1,j\neq i}^{n-1} (i-j)} \qquad \text{OPEN}$$

Newton Cotes

- Mid-Point, Trapezoid and Simpson belong to the family of Newton Cotes formula, which use equally spaced points in [a,b]
- We distinguish between OPEN and CLOSED type, depending on if a and b are included (semi-closed formulas also exists)
- Note the given Newton Cotes formulas refer to a **single** interval (not composite).
- n is the number of sub-intervals in a single interval, e.g. for Simpson n=2 (do not confuse with m, in the composite formulas)

Newton Cotes Error

- The error below refers to **Δ=b-a**, i.e. they are for one single interval (not for the composite formula).
- **n** is the number of sub-intervals (e.g. in Simpson or MidPoint n=2, in Trapezoid n=1)
- For the composite formula the order of convergence is expressed with respect to **h=(b-a)/m**, and it decreases by 1
- Note that, even number of sub-interval leads to superior convergence

SINGLE INTERVAL		COMPOSITE FORMULA	
CLOSED, n even:	$O\left(\Delta^{n+3}\right)$	CLOSED, n even:	$O\left(h^{n+2}\right)$
CLOSED, n odd:	$O\left(\Delta^{n+2}\right)$	CLOSED, n odd:	$O\left(h^{n+1}\right)$
OPEN, n even:	$O\left(\Delta^{n+1}\right)$	OPEN, n even:	$O\left(h^{n}\right)$
OPEN, n odd:	$O\left(\Delta^{n}\right)$	OPEN, n odd:	$O\left(h^{n-1}\right)$

Newton Cotes Example: Mid-Point Rule

- Focusing on one single interval [a,b] we have: Δ=b-a
  - there are 3 points x_0=a, x_1=(a+b)/2, x_2=b, i.e. 2 sub-intervals (so n=2)
  - points at the boundary are not used, so the formula is of type *OPEN*
  - Error is $O(\Delta^{n+1})$, e.g. $O(\Delta^3)$
  - Formula is:

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=1}^{1} B_i f(x_i) = (b-a)B_1 f(x_1)$$

$$\text{where } B_i = f(x_i), \quad \text{where } B_i = \frac{\displaystyle\int_0^1 \prod_{j=1,j\neq i}^{1} (2x-j)\,dx}{\displaystyle\prod_{j=1,j\neq i}^{1} (i-j)} = \frac{\int_0^1 dx}{1} = 1$$

$$\text{hence } \int_a^b f(x)dx \approx (b-a)f(x_1)$$

Newton Cotes Example: Simpson Rule

- Focusing on one single interval [a,b] we have: Δ=b-a
  - there are 3 points x_0=a, x_1=(a+b)/2, x_2=b, i.e. 2 sub-intervals (so n=2).
  - points at the boundary are used, so the formula is of type *CLOSED*
  - Error is $O(\Delta^{n+3})$, e.g. $O(\Delta^5)$

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{2} A_i f(x_i) = (b-a)[A_0 f(x_0) + A_1 f(x_1) + A_2 f(x_2)]$$

where

$$A_i = \frac{\int_0^1 \prod_{j=0,j\neq 0}^{2}(2x-j)\,dx}{\prod_{j=0,j\neq 0}^{2}(i-j)} = \frac{\int_0^1 (2x-1)(2x-2)\,dx}{(0-1)(0-2)} = \frac{1}{6}$$

$$A_i = \frac{\int_0^1 \prod_{j=0,j\neq 1}^{2}(2x-j)\,dx}{\prod_{j=0,j\neq 1}^{2}(i-j)} = \frac{\int_0^1 (2x-0)(2x-2)\,dx}{(1-0)(1-2)} = \frac{4}{6}$$

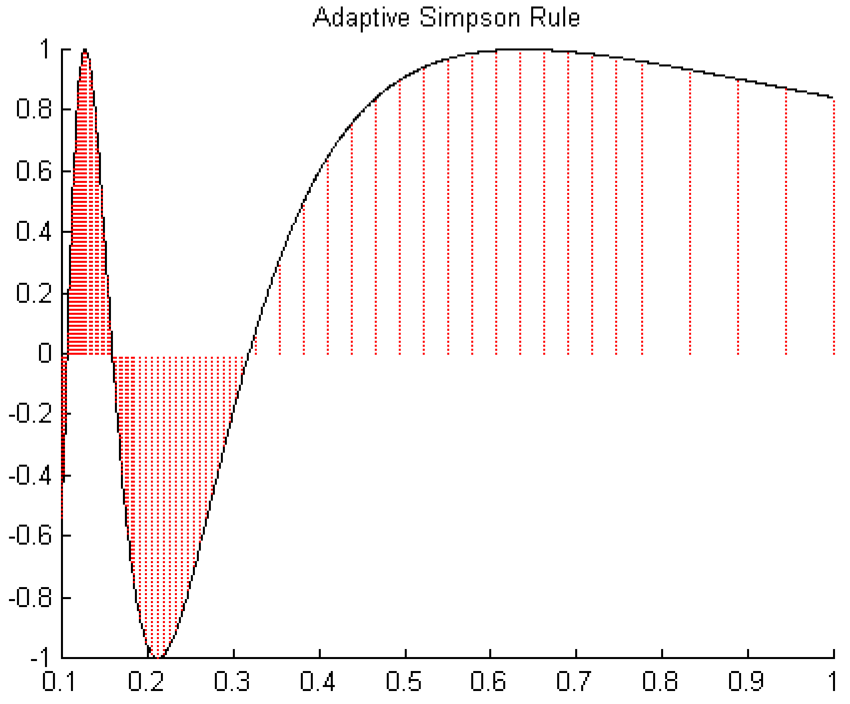
$$A_i = \frac{\int_0^1 \prod_{j=0,j\neq 2}^{2}(2x-j)\,dx}{\prod_{j=0,j\neq 2}^{2}(i-j)} = \frac{\int_0^1 (2x-0)(2x-1)\,dx}{(2-0)(2-1)} = \frac{1}{6}$$

Adaptive Simpson Rule

- At iteration 1 we start with Simpson with exactly 3 points in [a,b]: {a,c,b}, where c=(b+a)/2
- We compute Simpson in [a,c] and [c,b], adding a central point in each interval
- We check if S(a,b) – [S(a,c)+S(c,b)] is below a certain tolerance. If yes we stop, otherwise we keep bisecting and we now repeat the process individually on [a,c] and [c,b]
- The idea is that we add point only where it is necessary, i.e. where the error is larger

Adaptive Simpson Rule

- We observe that the density of point is not uniform



Adaptive Simpson Rule

- We can use Richardson extrapolation
- Simpson error terms are of order $O(h^4)$ and we showed that we can combine Simpson approximations with step size h and 2h to obtain a new approximation of order $O(h^5)$:
  - Compute $S_1=S[a,b]$, error $O(h^4)$
  - Compute $S_2=[S(a,c)+S(c,b)]$, error $O(h^4)$
  - Compute $S_3=(16\ S_2+S_1)/15$, error $O(h^5)$
- Because the error of S_3 is much smaller than the error of S_2, we assume it is negligible and (S_3-S_2) is just the error of S_2
- We obtain the stopping criteria:

  $|S_2-S_3| < \varepsilon \Rightarrow |S_2 - S_1| < 15\ \varepsilon$

  where ε is the accuracy criteria for the portion of interval [a,b]
  - if we stop, we return S_3

```
>> [I,n]=AdaptiveSimpson(@myfun,0,0.75*pi(),1e-8)
I =  3.14970879432736
n =  121
```

Gaussian Integration

- In Newton Cotes formula, we fix the position of the points and we compute weights for each point
- If we choose also the position of the points arbitrarily, we have more degrees of freedom to minimize the error
- An n-point Gaussian quadrature rule is constructed to yield an exact result for polynomials of degree $2n - 1$ or less by a suitable choice of the points x_i and weights w_i for $i = 1,...,n$
- This is way more accurate than Newton Cotes!
- The domain of integration for such a rule is, without loss of generality, conventionally taken as $[-1, 1]$
- The catch is that high order is not always good. In order to obtain high accuracy, the function must behave like a polynomial (i.e. very smooth)

Gaussian Integration

- If we can write f(x) as a product of 2 functions w(x)g(x), where g(x) behaves like a polynomial, then

$$\int_a^b f(x)dx = \int_a^b w(x)g(x)dx \approx \sum_i A_i g(x_i)$$

- .e. the weights A_i are chosen so that if g(x) was indeed a polynomial, the formula would be exact

$$\sum_{i=1}^{n} A_i x_i^j = \int_a^b w(x) x_i^j dx \quad j = 0 \ldots 2n-1$$

Non linear system of 2n equations in 2n unknowns Ai, xi

- Newton Cotes are exact if f(x) is a polynomial, here we can achieve exact results also for a polynomial multiplied by some weight function, which can be chosen to improve smoothness of f(x), but if not chosen properly could turn a smooth f(x) into a non-smooth g(x)!

Gaussian Integration

- For some special weight functions, the points x_i are the roots of some special polynomials and the values of the weights A_i are tabulated
- Once we get xi and Ai from the tables, we simply apply the formula:

$$\int_a^b f(x)dx = \int_a^b w(x)g(x)dx \approx \sum_i A_i g(x_i)$$

Note that g(x) is not the same as f(x)

Gaussian Integration

$W(x)$	$[a,b]$	Polynomial
1	$[-1,1]$	Legendre
$\frac{1}{\sqrt{1-x^2}}$	$(-1,1)$	Chebishev 1
$\sqrt{1-x^2}$	$[-1,1]$	Chebishev 2
$\frac{1}{\sqrt{x}}$	$[0,1]$	Related to Legendre
$\sqrt{x}$	$[0,1]$	Related to Legendre
$\frac{x}{\sqrt{1-x}}$	$[0,1]$	Related to Chebishev 1
$x^{\alpha}e^{-x}$	$[0,\infty]$	Laguerre
e^{-x^2}	$[-\infty,\infty]$	Hermite
$(1-x^{\alpha})(1+x^{\beta})$	$(-1,1)$	Jacobi

Polynomial roots (x_i) and weights (A_i) can be found tabulated

Gauss-Legendre Roots and Weights Table Example

n = 2

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

n = 3

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

n = 4

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

Gaussian Integration: Example

$$I = \int_0^{\frac{3}{4}\pi} \frac{1}{4} + \frac{3}{2}\sin(x)dx = \frac{3}{16}\pi + \left(\frac{\sqrt{2}}{2} + 1\right)\frac{3}{2} \approx 3.1497$$

- We can use Gauss Legendre weight function w(x)=1
- We use 4 points (same cost as previous methods, so we can compare)
- Gauss Legendre requires the interval to be [0,1], so we need to transform the integration domain, then we apply the formula

$$\text{let } x = \frac{3\pi}{8}(y+1) \implies dx = \frac{3\pi}{8}dy$$

$$\int_0^{\frac{3}{4}\pi} \frac{3}{2} + \frac{1}{4}sin(x)dx = \int_{-1}^{1} \underbrace{(1)}_{w(x)} \underbrace{\left[\frac{1}{4} + \frac{3}{2}sin\left(\frac{3\pi}{8}(y+1)\right)\frac{3\pi}{8}\right]}_{g(x)} dy \approx \sum_{i=1}^{4} w_i\, g(y_i) \approx 3.1497$$

```
>> f=@(x)(1/4+3/2*sin(3*pi/8*(x+1)))*3*pi/8;
>> w=[0.6521451548625461, 0.6521451548625461, 0.3478548451374538, 0.3478548451374538];
>> y=[-0.3399810435848563, 0.3399810435848563, -0.8611363115940526, 0.8611363115940526];
>> w*f(y)'
ans =  3.1497
```

Gaussian Integration: Example

- Could we choose another weight function instead of Legendre?
- Yes, but in this case Legendre is an obvious choice because
  - It works on finite domain [-1,1] and our integral is on a finite domain
  - No other weight function appears explicitly as a factor of f(x), hence I should multiply and divide f(x) by the weight function to make it appear
  - With Legendre I obtain a function g(x) infinitely continuous (you can verify that)
- Remember the goal is to make g(x) smooth. With the choice of Legendre I achieve that with minimal effort (just transform the integral to [-1,1]. So there is no reason to use another function
- For example, Chebishev2 would have a g(x) less smooth

$$\int_0^{\frac{3}{4}\pi} \frac{3}{2} + \frac{1}{4} sin(x) dx = \int_{-1}^{1} \underbrace{(\sqrt{1-x^2})}_{w(x)} \underbrace{\left[\frac{1}{4} + \frac{3}{2} sin\left(\frac{3\pi}{8}(y+1)\right) \frac{3\pi}{8}\right] \frac{1}{\sqrt{1-x^2}}}_{g(x)} dy$$

Comparison with Newton Cotes

- Suppose we use Gauss-Legendre with 3 points
- Convergence on single interval
  - This is expected to be exact for a polynomial of order 3x2-1=5, i.e. $O((b-a)^7)$
  - Simpson also uses 3 points, but it is only exact for a polynomial up to order 3, i.e. $O((b-a)^5)$
- When we transform to **composite**, if we divide [a,b] in m sub-intervals,
  - Simpson requires roughly 2n evaluations, because at the boundary of every interval it uses common points
  - Gauss instead requires 3n evaluations
- Computation cost is O(n) in both cases, but accuracy is $O(h^6)$ vs $O(h^4)$

Warnings

- Same issues discussed with interpolation: high order schemes can give bad surprises with non-smooth functions
- Beware discontinuities and singularities!
  - Non adaptive method will lead to very poor accuracy (they rely on f(x) to ne smooth)
  - Adaptive method will keep bisecting, becoming very expensive, and with poor accuracy (the interval could become smaller than machine precision). There needs to be a guard to avoid bisecting too much.
- Identify discontinuities and break the integral
- Fast and frequent changes of slope can be as bad as discontinuities

Advanced Methods

- Most commonly used methods are adaptive
- They are combination of more basic methods
- A key performance requirement of adaptive method is to be able to reuse previously computed points (e.g. in Simpson, at every bisection, we only need 2 new extra points)

Multiple Integrals

- Very expensive: computational cost grows exponentially with number of dimensions
- If I need n points for a 1-D integral, I may need n^3 for a 3-D integral
- When possible we should reduce the order of the integral
  - e.g. if we start from a double integral and we can solve the inner integral analytically, we are left with a 1-D integral
- A good technique for high dimensionality is Monte Carlo integration

Further Readings

- Prof Amos Rons' lecture notes
  http://pages.cs.wisc.edu/~amos/412/lecture-notes/lecture17.pdf
  http://pages.cs.wisc.edu/~amos/412/lecture-notes/lecture18.pdf
  http://pages.cs.wisc.edu/~amos/412/lecture-notes/lecture19.pdf
- Prof Saltzman's lecture notes
  http://www.dirac.org/numerical/gaussian_quadrature/gaussian.pdf
- Numerical Recipes in C++