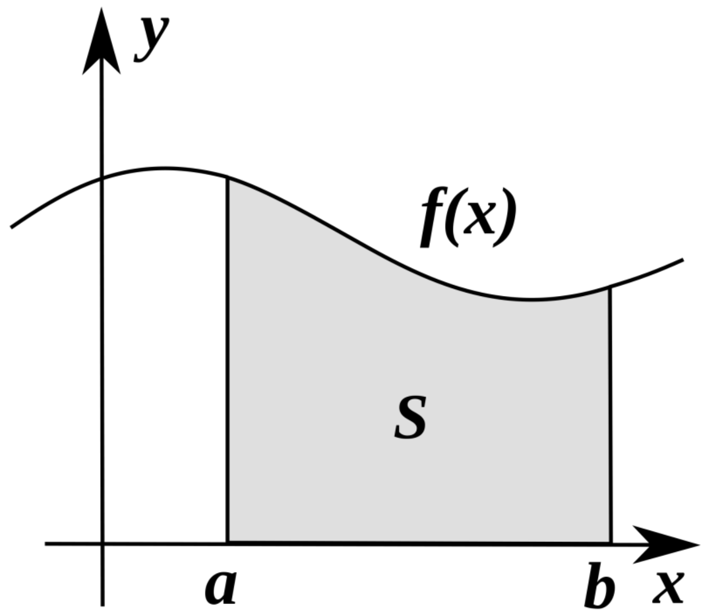
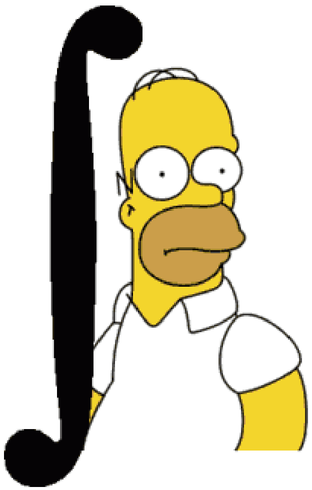
Techniques for Numerical Integration

$$\int_a^b f(x)\,dx$$



Lauren Donohoe

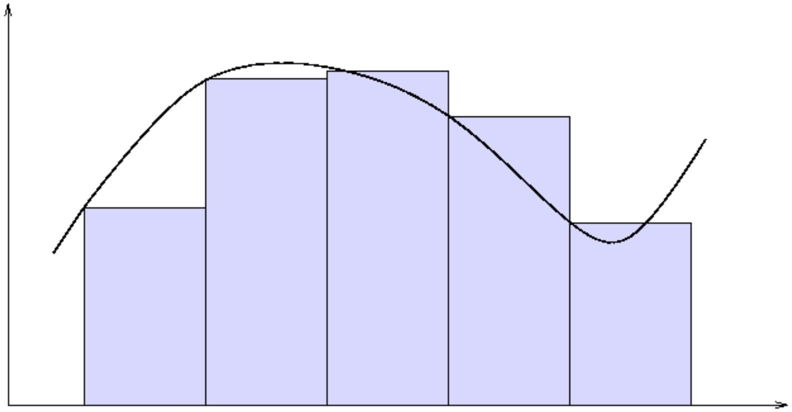
Numerical Integration Techniques



- Trapezoid Rule
- Simpson's Rule(s)
- Romberg Integration
- Gaussian Quadrature
  - Gauss-Lobatto Quadrature
  - Gauss-Kronrod Quadrature

MATLAB Comparison

- trapz()
- simps()
- quad()
- romberg()
- quadl()
- quadgk()
- integral()



Working with Singularities

The Integral – The Basics

What you already know

Indefinite Integral:

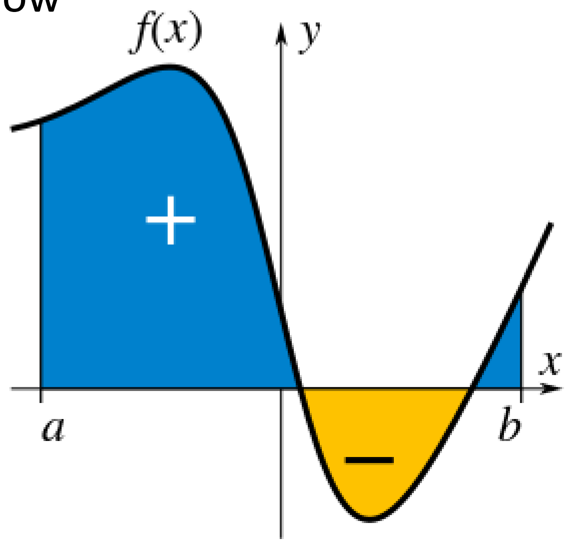
$$F(x) = \int f(x)\,dx.$$

Definite Integral:

On the closed interval from a to b, or [a,b]

$$\int_a^b f(x)\,dx$$

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



Integral - Area Under the Curve

The Fundamental Theorem of Calculus:

If f(x) is a continuous, real-valued function defined on the closed interval [a,b], and F(x) is defined for all x in [a,b], then F(x) is differentiable on (a,b)

$$F(x) = \int_a^x f(t)\,dt.$$

$$F'(x) = f(x)$$

The Interpolation Polynomial

The simple approach to Numerical Integration

Let $p_n(x)$ be the polynomial to interpolate f(x) at $x_0, x_1, \ldots, x_n$ where

$$a \leq x_0 < x_1 < \cdots < x_n = b.$$

Then use this interpolation polynomial to compute f(x) by using

$$\int_a^b p_n(x)dx \approx \int_a^b f(x)dx.$$

Where $a = x_0$ and $b = x_n$.

Then taking the form

$$I_n(f) = \sum_{k=0}^{n} c_k f(x_k) \approx \int_a^b f(x)dx,$$

the function $I_n(f)$ takes the ***exact* value of the integral for polynomials of degree n or less**

$$\sum_{k=0}^{n} c_k x_k^j = \int_a^b x^j dx, \quad j = 0, \ldots, n.$$

Represented as a linear system

→

$$\begin{pmatrix} 1 & 1 & \cdots & \cdots & 1 \\ x_0 & x_1 & \cdots & \cdots & x_n \\ x_0^2 & x_1^2 & \cdots & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^n & x_1^n & \cdots & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \int_a^b dx \\ \int_a^b x dx \\ \vdots \\ \int_a^b x^n dx \end{pmatrix}$$

The Interpolation Polynomial - Applied

The first example

If we let [a,b] = [0,1],
x_k = kh where h = 1/n

$$\begin{pmatrix} 1 & 1 & \cdots & \cdots & 1 \\ x_0 & x_1 & \cdots & \cdots & x_n \\ x_0^2 & x_1^2 & \cdots & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^n & x_1^n & \cdots & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \int_a^b dx \\ \int_a^b x dx \\ \vdots \\ \int_a^b x^n dx \end{pmatrix}$$

For n = 1
then X_0 = 0 and x_1 = 1,

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} \int_a^b dx \\ \int_a^b x dx \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}$$

Has solution C_0 = C_1 = ½, and plugging this back into the equation for the polynomial,

$$I_1(f) = \frac{1}{2}[f(0) + f(1)]$$

Or more generally:

$$I_1(f) = \frac{b-a}{2}[f(a) + f(b)]$$

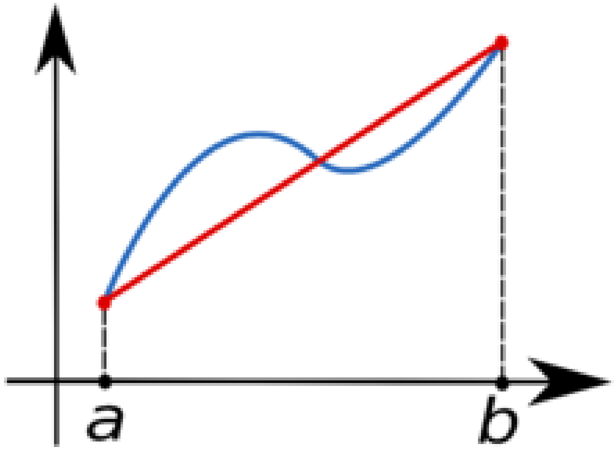
The Trapezoid Rule

A technique for approximating the definite integral

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

The trapezoid rule approximates the area under the curve as a trapezoid with upper corners on the curve, and determines the value for the interval using the area of the trapezoid formed.

(only ONE trapezoid, for now)



Q = trapz(Y) returns the approximate integral of Y using the trapezoid method (by default, with unit spacing)

The Interpolation Polynomial - Applied

The second example

If we continue to let [a,b] = [0,1], x_k = kh where h = 1/n

$$\begin{pmatrix} 1 & 1 & \cdots & \cdots & 1 \\ x_0 & x_1 & \cdots & \cdots & x_n \\ x_0^2 & x_1^2 & \cdots & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0^n & x_1^n & \cdots & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \int_a^b dx \\ \int_a^b x dx \\ \vdots \\ \int_a^b x^n dx \end{pmatrix}$$

For n = 2

then X_0 = 0, x_1 = ½ and x_2 = 1,

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1/2 & 1 \\ 0 & 1/4 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \end{pmatrix}$$

Has solution C_0 = C_2 = 1/6, and C_1 = 2/3 and plugging this into the polynomial,

$$I_2(f) = \frac{1}{6}[f(0) + 4f(1/2) + f(1)]$$

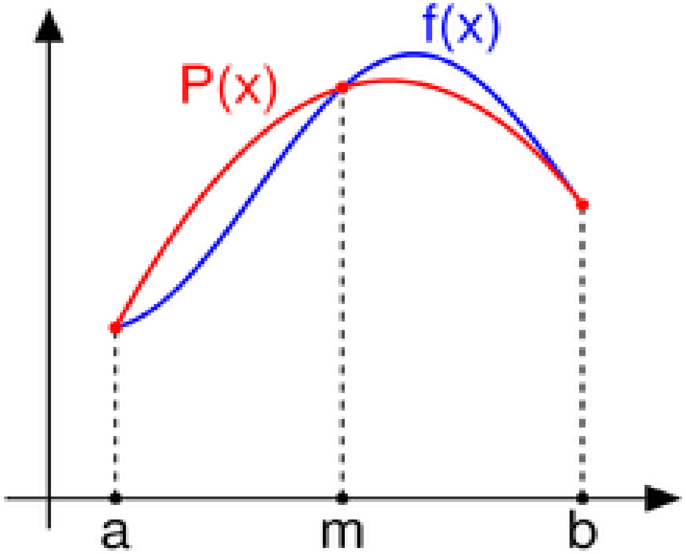
Or more generally:

$$I_2(f) = \frac{(b-a)}{6}[f(a) + 4f([a+b]/2) + f(b)]$$

Simpson's Rule

A *better* technique for approximating the definite integral

$$\int_a^b f(x)\,dx \approx \tfrac{b-a}{6}\left[f(a) + 4f\left(\tfrac{a+b}{2}\right) + f(b)\right].$$



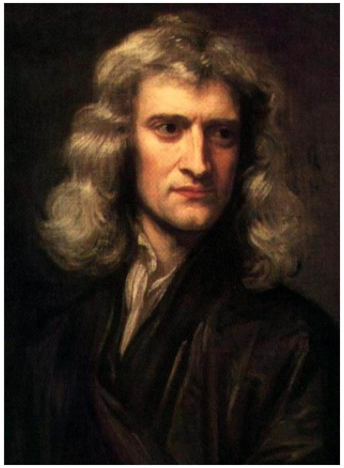
Simpson's rule approximates the area under the curve using **quadratic** interpolation
[Parabolic arcs rather than straight lines]

Simpson 3/8 Rule (n = 3)

$$\int_a^b f(x)\,dx \approx \tfrac{(b-a)}{8}\left[f(a) + 3f\left(\tfrac{2a+b}{3}\right) + 3f\left(\tfrac{a+2b}{3}\right) + f(b)\right]$$

Simpson's 3/8 rule approximates the area under the curve using **cubic** interpolation rather than quadratic interpolation

Newton-Cotes & Error Formulas



Recap: Interpolation Formula is used to approximate integrals in numerical analysis

n = 1 – Trapeziod rule
n = 2 – Simpson's rule
n = 3 – Simpson's 3/8 Rule

n = 4, 5, 6,… Newton – Cotes Formula of order n
(Guaranteed exact for degree n or less)

$x_0, x_1, \dots, x_n$
are **evenly spaced**

For unevenly spaced points, Gaussian Quadrature is necessary.

Error Formulas:

Trapezoid Rule

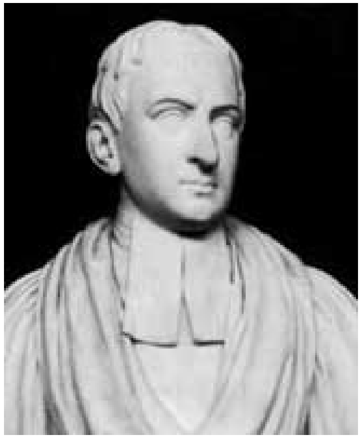
$$\int_a^b f(x)dx = I_1(f) - \frac{(b-a)^2}{12} f''(\xi_2)$$

for some $\xi_2 \in (a, b)$

Simpson's Rule

$$\int_a^b f(x)dx = I_2(f) - \frac{(b-a)h^4}{180} f^{(4)}(\xi_4)$$

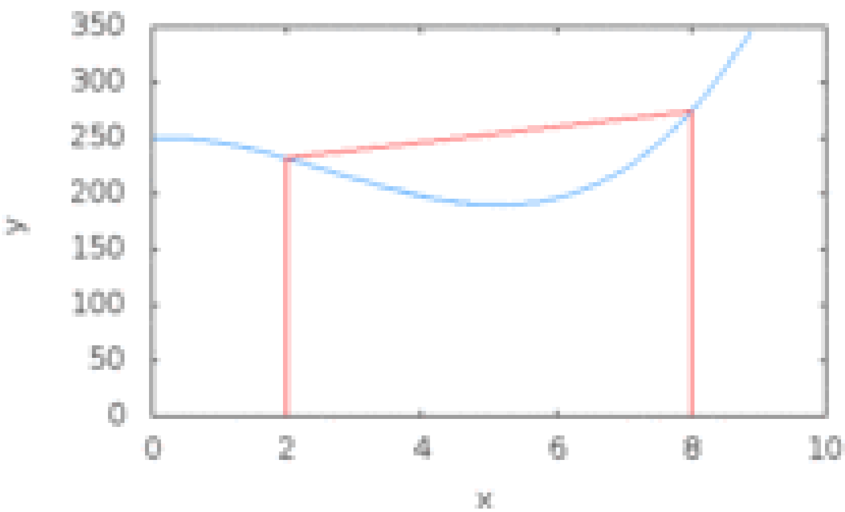
where $h = (b-a)/2$ and $\xi_4 \in (a, b)$



Composite Formulas

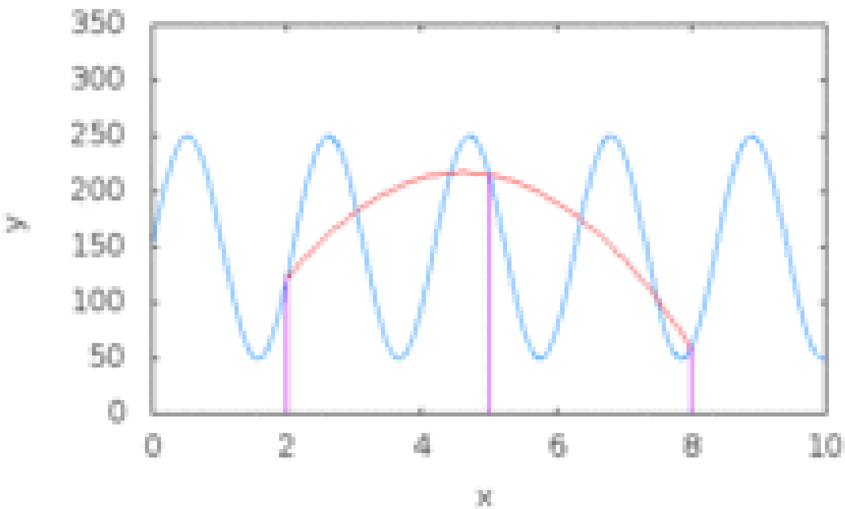
A *MUCH better* technique for approximating the definite integral

As n increases, the different Newton-Coates formulas help us to approximate the value of the integral of more complex curves, represented by higher order polynomials.



“Composite” =
Break the integral up into “smaller” integrals and sum the parts…

$$\int_a^b f(x)dx = \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x)dx$$



In general, the more “parts”, the better the approximation.

Composite Trapezoid Rule

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

For notation simplicity using spacing h = $x_{k+1} - x_k$ = (b-a)

$$T_0(h) = \int_{x_k}^{x_{k+1}} f(x)dx = \frac{h}{2}\sum_{k=0}^{n-1}[f(x_k) + f(x_{k+1})]$$

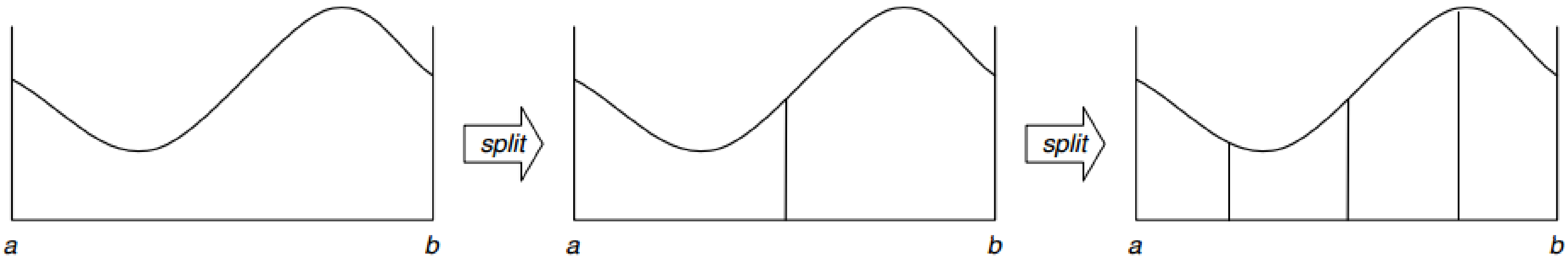
$$\begin{aligned}
T_0(h) &= \frac{h}{2}[\sum_{k=0}^{n-1} f(x_k) + \sum_{k=0}^{n-1} f(x_{k+1})] \\
&= \frac{h}{2}[\sum_{k=0}^{n-1} f(x_k) + \sum_{k=1}^{n} f(x_k)] \\
&= \frac{h}{2}[f(x_0) + f(x_n)] + h\sum_{k=1}^{n-1} f(x_k)
\end{aligned}$$

Therefore, to halve the interval size, midpoints:L $x_{k+1/2} = [x_k + x_{k+1}]/2$

$$\begin{aligned}
T_0(h/2) &= \frac{h}{4}[f(x_0) + f(x_n)] + \frac{h}{2}\sum_{k=0}^{n-1} f(x_k) + \frac{h}{2}\sum_{k=0}^{n-1} f(x_{k+1/2}) \\
&= \frac{1}{2}T_0(h) + \frac{h}{2}\sum_{k=0}^{n-1} f(x_{k+1/2})
\end{aligned}$$

Q = trapz(X,Y) returns the approximate integral of Y using the trapezoid method with spacing X

Adaptive Simpson's Rule



"Composite" =
Break the integral up into "smaller" integrals and sum the parts…

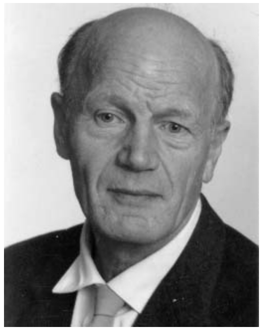
"Adaptive" =
Recursively splitting the integral in half and checking the error term compared to some desired maximum value

$$I = \int_a^b f(x)\, dx = S(a,b) + E(a,b)$$

Q = quad(fun,a,b,tol) returns the approximate integral of the function fun using "recursive adaptive composite Simpson's Rule" to within an error of tol (larger tolerance values means fewer evaluations and faster computation but a less accurate result

Romberg Integration

Combining everything up until this point…



The composite trapezoid rule for spacing h was

$$T_0(h) = \frac{h}{2}[f(x_0) + f(x_n)] + h\sum_{k=0}^{n-1} f(x_k)$$

And with half the interval size,
need the function evaluated at the midpoints

$$T_0(h/2) = \frac{1}{2}T_0(h) + \frac{h}{2}\sum_{k=0}^{n-1} f(x_{k+1/2})$$

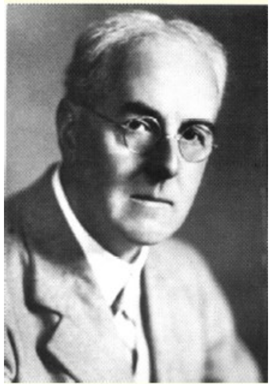
To(h) is needed in order to determine To(h/2) …
It follows that in order to compute To(h/2^k) we need To(h). To(h/2), … , To(h/2^k)

Following the same process to determine the composite Simpson's rule has the result

$$T_1(h) = (4T_0(h/2) - T_0(h))/3$$

Similarly, To(h/4) and To(h/2) are needed to form T_1(h/2), and so forth…

Then again in the same way, T_1(h) and T_1(h/2) can be used to determine T_2(h)…



[This technique of using multiple low order approximations to obtain a higher order approximation is called Richardson Extrapolation.]

Romberg Integration

Richardson Extrapolation + Trapezoid Rule = Romberg Integration

Such that finally, the general form

$$T_k(h) = (4^k T_{k-1}(h/2) - T_{k-1}(h))/(4^k - 1)$$

Which can be used with the table

$$\begin{array}{ccccccccc}
T_0(h) & \nearrow & T_1(h) & \nearrow & T_2(h) & \nearrow & T_3(h) & \nearrow & T_4(h) \\
T_0(h/2) & \nearrow & T_1(h/2) & \nearrow & T_2(h/2) & \nearrow & T_3(h/2) & & \\
T_0(h/4) & \nearrow & T_1(h/4) & \nearrow & T_2(h/4) & & & & \\
T_0(h/8) & \nearrow & T_1(h/8) & & & & & & \\
T_0(h/16) & & & & & & & &
\end{array}$$

in order to form the flow of the algorithm.

The rows $\rightarrow$

The columns $\downarrow$

The diagonals $\nearrow$

ALL converge to the exact value of the integral

Stopping Criterion for some tolerance ϵ $|T_{k-1}(h) - T_k(h)| \leq \epsilon$

Gaussian Quadrature

A *slightly different* technique for approximating the definite integral

"Quadrature" is a numerical analysis technique where a definite integral is approximated using a weighted sum of function values at specified points within the domain of integration

The n-point Gaussian Quadrature rule
yields exact results for polynomials of degree (2n-1) or less as long as a "suitable choice" of points x_i and weights w_i are used for i = 1,2,…,n

The domain is conventionally used as the closed interval [-1,1]

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

How is this different?
These "specified points" **DO NOT** have to be evenly spaced (as they did for Trapezoid, Simpson's, and Romberg)

Gaussian Quadrature

… using a "suitable choice" of points x_i and weights w_i

Gaussian Quadrature will produce accurate results if the function f(x) is well approximated by a polynomial function within the domain …

[This method is not well suited for functions with singularities…]

If f(x) can be written as

$$f(x) = w(x)g(x)$$

where g(x) can be well approximated using a polynomial and w(x) is known, then alternative points and weights that depend on the **weighing function** give better results

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

and the evaluation points x_i are the roots (zeros) the specific polynomial used to approximate the function, a polynomial belonging to a family of orthogonal polynomials called the *orthogonal polynomial sequence*

Gaussian Quadrature

Weighing Functions

Quadrature Type	Weighing Function w(x)	Orthogonal Polynomials
Gauss-Legendre Quadrature	1	Legendre Polynomials
Gauss-Jacobi Quadrature	$(1-x)^{\alpha}(1+x)^{\beta}, \quad \alpha, \beta > -1$	Jacobi Polynomials
Chebyshev-Gauss Quadrature	$\frac{1}{\sqrt{1-x^2}}$	Chebyshev Polynomials (first kind)
Chebyshev-Gauss Quadrature	$\sqrt{1-x^2}$	Chebyshev Polynomials (second kind)
Gauss-Laguerre Quadrature	e^{-x}	Laguerre Polynomials
Gauss-Laguerre Quadrature	$x^{\alpha}e^{-x}, \quad \alpha > -1$	Generalized Laguerre Polynomials
Gauss-Hermite Quadrature	e^{-x^2}	Hermite Polynomials

Gauss – Lobatto Quadrature

An Extension of Gaussian Quadrate

How is Gauss-Lobatto different than Gaussian Quadrature?

- The integration points INCLUDE the endpoints of the integration interval
- Accurate for polynomials up to degree 2n-3

The Lobatto Quadrature of the function f(x) on the interval [-1,1] is

$$\int_{-1}^{1} f(x)\,dx = \frac{2}{n(n-1)}[f(1)+f(-1)] + \sum_{i=2}^{n-1} w_i f(x_i) + R_n$$

with weights

$$w_i = \frac{2}{n(n-1)[P_{n-1}(x_i)]^2}, \qquad x_i \neq \pm 1$$

and remainder

$$R_n = \frac{-n(n-1)^3 2^{2n-1}[(n-2)!]^4}{(2n-1)[(2n-2)!]^3} f^{(2n-2)}(\xi), \qquad -1 < \xi < 1.$$

q = quadl(fun,a,b) approximates the integral of the function fun from a to b, to within an error of 10^{-6} using adaptive Lobatto quadrature. (Limits a and b must be finite.)

Gauss – Kronrod Quadrature

Another Extension of Gaussian Quadrate

Remember:
Gaussian Quadrature of order n is accurate for polynomials up to degree 2n-1

Gauss-Kronrod Rules:

The interval [a,b] is subdivided such that the new evaluation points of these subintervals never coincide with the original evaluation points except at zero and odd numbers

→

Adding n+1 points to an n-point Quadrature, in this manner makes the **resulting rule of order 3n+1.** This allows for computation of much higher-order estimates using function values of lower-order estimates

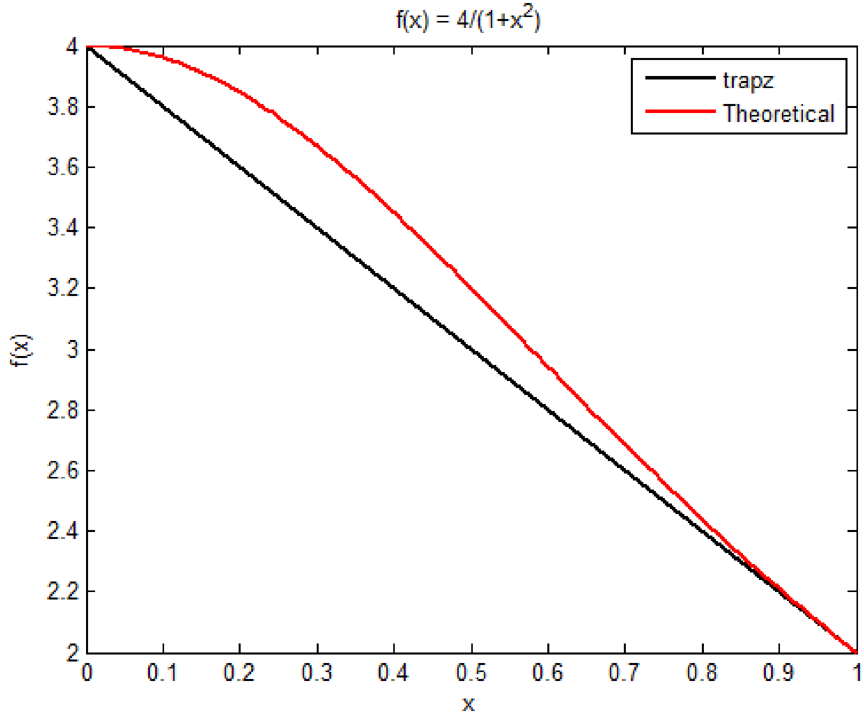
q = quadgk(fun,a,b) approximates the integral of the function fun from a to b using high-order adaptive quadrature with default error tolerances. (Limits a and b can be infinite or complex.)

MATLAB Comparison - Code

```
% function 4/(1+x^2)from 0 to 1 (integral is pi)
myfun = @(x) 4./(1+x.^2);
```

```
%% TRAPEZOID RULE
n = 2;
points_t = linspace(0,1,n);
interp_t = feval(myfun,points_t);

figure1 = figure('Color',[1 1 1]);
plot(points_t,interp_t,'Color','k','LineWidth',2)
title('f(x) = 4/(1+x^2)')
xlabel('x')
ylabel('f(x)')

tic
int_t = trapz(interp_t);
T_t = toc;
disp(['Trapezoid Rule: ',num2str(int_t)])
errort = abs(int_t-pi);
disp(['With Error: ',num2str(errort)])
disp(['Time elapsed: ',num2str(T_t),' seconds'])
disp(' ')
```

```
%% SIMPSON'S RULE
n = 3;
points_s =  linspace(0,1,n);
interp_s = feval(myfun,points_s);

hold on;
plot(points_s,interp_s,'Color','b','LineWidth',2)

tic
int_s = simps(interp_s)/2;
T_s = toc;
```




```
int_q = quad(myfun,0,1,0.1);
int_r = romberg(myfun,0,1,0.1);
int_r2 = romberg(myfun,0,1,1e-14);
int_l = quadl(myfun,0,1);
int_gk = quadgk(myfun,0,1);
int_i = integral(myfun,0,1);
```

MATLAB Comparison - Results

	Integral Value	Error	Time Elapsed (seconds)	MATLAB Function
Trapezoid Rule	3	0.14159	0.02266	trapz()
Simpson's Rule	3.1333	0.0082593	0.030717	simps() *
Adaptive Composite Simpson Quadrature	3.14159525048309	2.5969e-06	0.023679	quad()
Romberg Integration (with tolerance 0.1)	3.141592502458707	1.5113e-07	0.001437	romberg() *
Romberg Integration (with tolerance 1e-14)	3.141592653589793	0	0.014495	romberg() *
Gauss-Lobatto Quadrature	3.141592707032192	5.3442e-08	0.02867	quadl()
Gauss-Kronrod Quadrature	3.141592653589793	0	0.067964	quadgk()
MATLAB's Integral Function	3.141592653589793	0	0.089876	integral()

π = 3.1415926535897932384626… ↑ ↑ Zero to double precision

Differences in MATLAB Functions

Which function should I use to perform numerical integration?

- quad() is more efficient for low accuracy with non-smooth scalar-valued functions
- quadl() is more efficient for higher accuracy with smooth scalar-valued functions
- quadv() & integral() perform vectorized quadrature for a vector-valued function
- quadgk() is the most efficient for high accuracy if the function is oscillatory
- quadgk() & integral() supports infinite limits of integration
- quadgk() & integral() can handle moderate singularities at the endpoints
- integral() automatically supports mixed relative (digits) and absolute (when I = 0) error control
- integral() uses a higher order method than quadl() so it is usually more accurate on smooth problems
- integral() is more reliable than quad() because it starts with a much finer initial mesh than quad() and is more conservative in error control

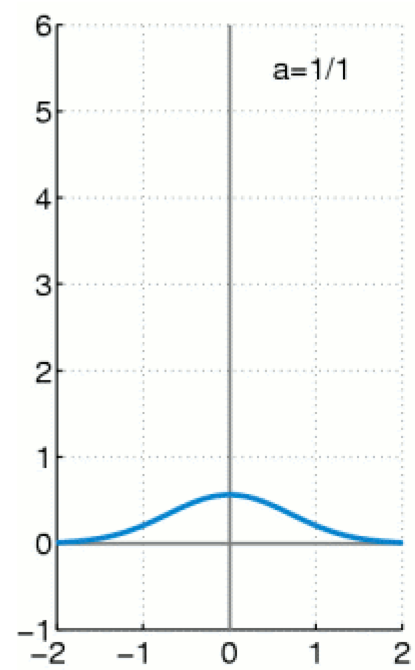
Handling Singularities in MATLAB

- quadgk() & integral() can handle moderate singularities at the endpoints
- quad() is more efficient for low accuracy with non-smooth scalar-valued functions

"If there is a singularity within the domain of the function, the sum of the intervals over multiple subintervals can be used with the singularities at endpoints"

The Dirac-Delta Function

$$\delta_a(x) = \frac{1}{a\sqrt{\pi}} e^{-x^2/a^2}$$




```
a = 1e-7;
myfun = @(x) (1/(a*sqrt(pi)))*exp((-x.^2)/(a^2));
q = quadgk(myfun,-1,1)
q = quadgk(myfun,-1,0) + quadgk(myfun,0,1)
```

	Without Split	With Split
quad()	a = 1e-20 *	a = 1e-21
quadgk()	a = 1e-4	a = 1e-7
integral()	a = 1e-4	a = 1e-7

* Warning: Minimum step size reached; singularity possible.

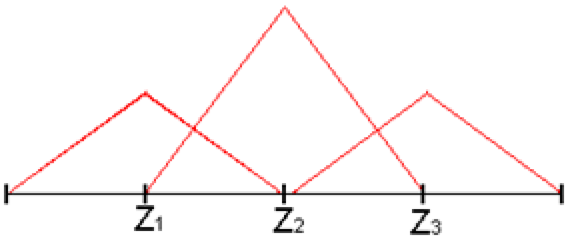
Pocklington's Integral Equation

Using MATLAB to evaluate Pocklington's Integral Equation

$$\frac{1}{j\omega\varepsilon_0}\int_{-L/2}^{L/2} I(z')\left[\frac{\partial^2}{\partial z^2}+k^2\right]G(z,z')dz' = -E_z^i(z)$$

Using piecewise triangular sub-domain functions

$$f_n(z)=\begin{cases}\dfrac{\Delta-|z-z_n|}{\Delta}; & z_n-\Delta<z<z_n+\Delta\\ 0; & \text{otherwise}\end{cases}$$



And point-matching (or collocation) weighing functions

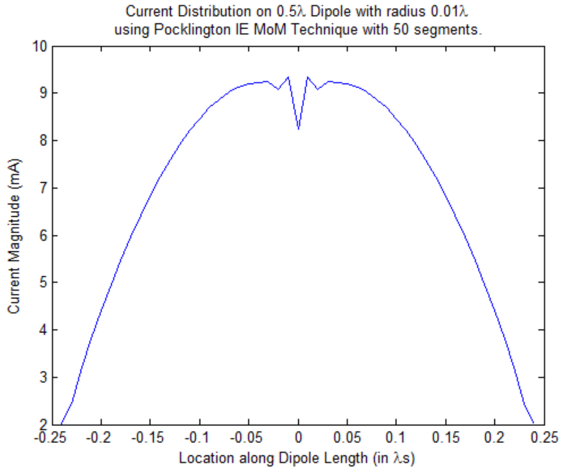
$$w_m(z)=\delta(z-z_m)$$

The kernel of Pocklington's I.E. has a **singularity** at the middle segment of the dipole

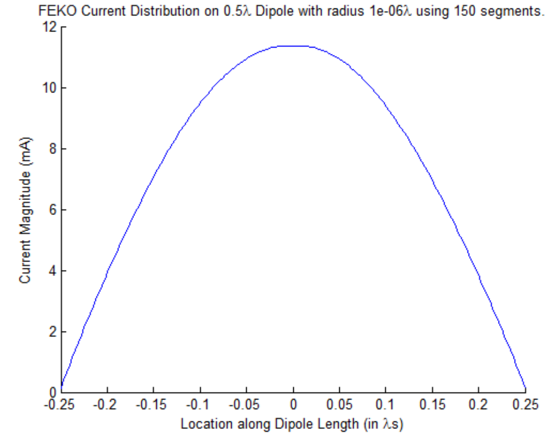
$$K(z_m,z')=\frac{1}{4\pi j\omega\varepsilon_0}\left[\frac{e^{-jkR}}{R^5}\left[(1+jkR)(2R^2-3a^2)+k^2a^2R^2\right]\right]$$

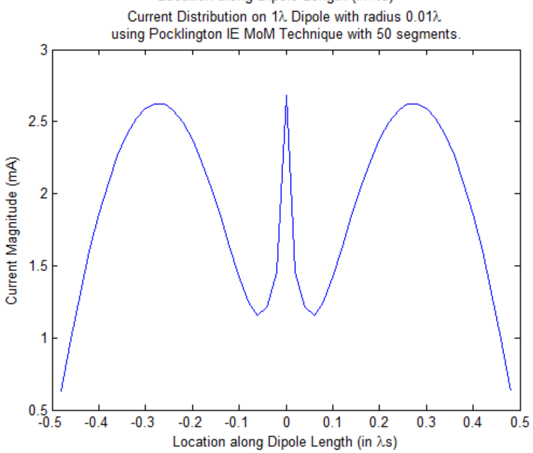
$$R=\sqrt{(z-z')^2+a^2}$$

Pocklington's I.E.

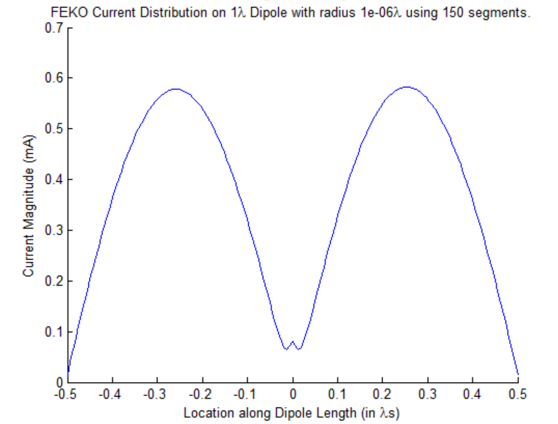


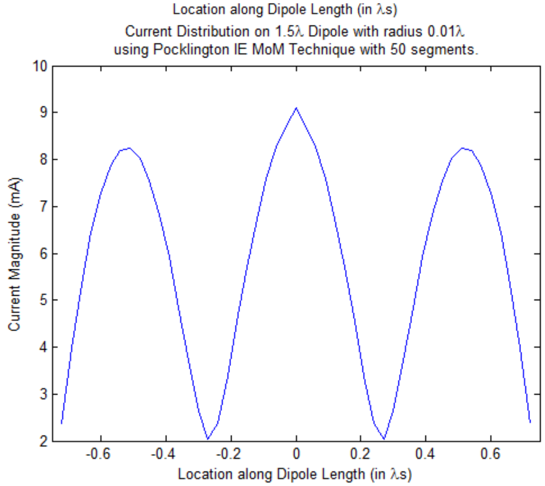
- FEKO →



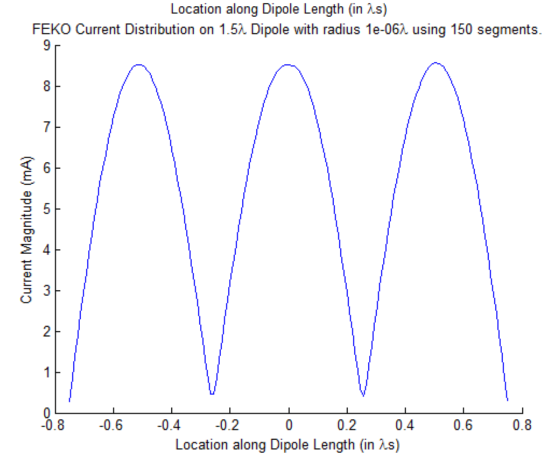


- Singularity on the center segment
- Splitting the integral does not help
- Comparison of impedances (center segment)





- ← MATLAB



References

http://www.mathstat.dal.ca/~tkolokol/classes/1500/romberg.pdf

http://en.wikipedia.org/wiki/Integral#Fundamental_theorem_of_calculus_2

http://en.wikipedia.org/wiki/Polynomial_interpolation

http://en.wikipedia.org/wiki/Simpson%27s_rule

http://en.wikipedia.org/wiki/Newton%E2%80%93Cotes_formulas

http://www.cse.psu.edu/~barlow/cse451/classnotes.html

Advanced Mathematics and Mechanics Applications Using MATLAB, Third Edition
By David Halpern, Howard B. Wilson, Louis H. Turcotte

Advanced Engineering Mathematics with MATLAB, Third Edition
By Dean G. Duffy

http://en.wikipedia.org/wiki/Trapezoidal_rule

http://www.mathworks.com/matlabcentral/fileexchange/25754-simpsons-rule-for-numerical-integration/content/simps.m

http://www.mathworks.com/help/matlab/

http://ezekiel.vancouver.wsu.edu/~cs330/lectures/integration/simpsons.pdf