Mahidol University
Faculty of Information and Communication Technology

Numerical Integration of Functions

Rawesak Tanawongsuwan, Ph.D.
rawesak.tan@mahidol.ac.th

This slide is part of teaching materials for ITCS122 Numerical Methods
Semester 2/2023, Calendar year 2024

Introduction

- Previously, functions to be integrated numerically will typically be of two forms: a table of values or a function.
  - For tabulated information, we are limited by the number of points that are given.
  - For the case where the function is available, we can generate as many values of $f(x)$ as needed to hopefully reach acceptable accuracy.
- Previous algorithms (for example, the Simpson’s 1/3 rule) might be acceptable to use in integration.
- However, there are more efficient methods which take an advantage on the ability to generate function values to develop efficient schemes for numerical integration

Richardson Extrapolation (1)

Use two estimates of an integral to compute a third, more accurate approximation

The estimate and the error associated with the composite trapezoidal rule

$$I = I(h) + E(h)$$

I = the exact value of the integral

$I(h)$ = the approximation from the composite trapezoidal rule

$E(h)$ = the truncation error

Recall that the error of the composite trapezoidal rule

$$E \cong -\frac{(b-a)^3}{12n^2}\bar{f}'' = -\frac{b-a}{12}h^2\bar{f}''$$

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

Richardson Extrapolation (2)

If we make two different estimates using step sizes of h_1 and h_2

$$I(h_1) + E(h_1) = I(h_2) + E(h_2)$$

And assume that $\bar{f}''$ is constant regardless of step size

Ratio of the two errors

$$\frac{E(h_1)}{E(h_2)} \cong \frac{h_1^2}{h_2^2}$$

$$E(h_1) \cong E(h_2)\left(\frac{h_1}{h_2}\right)^2$$

Richardson Extrapolation (3)

$$I(h_1) + E(h_2)\left(\frac{h_1}{h_2}\right)^2 = I(h_2) + E(h_2)$$

$$E(h_2) = \frac{I(h_1) - I(h_2)}{1 - \left(\frac{h_1}{h_2}\right)^2}$$

$$I = I(h_2) + E(h_2) = I(h_2) + \frac{1}{\left(\frac{h_1}{h_2}\right)^2 - 1}[I(h_2) - I(h_1)]$$

It can be shown (Ralston and Rabinowitz, 1978) that the error of this estimate is $O(h^4)$.

It means that if two $O(h^2)$ estimates $I(h_1)$ and $I(h_2)$ are calculated for an integral using step sizes of h_1 and h_2, an improved $O(h^4)$ estimate can be obtained.

Combining integrals to obtain improved estimates

- For the special case where the interval is halved ($h_2 = \frac{h_1}{2}$), this becomes

$$I = \frac{4}{3} I(h_2) - \frac{1}{3} I(h_1)$$

- For the cases where there are two $O(h^4)$ estimates and the interval is halved ($h_m = \frac{h_l}{2}$), an improved $O(h^6)$ estimate may be formed using :

$$I = \frac{16}{15} I_m - \frac{1}{15} I_l$$

- For the cases where there are two $O(h^6)$ estimates and the interval is halved ($h_m = \frac{h_l}{2}$), an improved $O(h^8)$ estimate may be formed using :

$$I = \frac{64}{63} I_m - \frac{1}{63} I_l$$

Note that I_m and I_l are the more and less accurate estimates

Example : Richardson Extrapolation (1)

Let $f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$

Use Richardson extrapolation to compute $\int_0^{0.8} f(x)dx$

The exact value of the integration $= 1.640533$

Single and composite applications of the trapezoidal rule can be used to evaluate the integral

Segments	$\boldsymbol{h}$	**Integral**	ε_t
1	0.8	0.1728	89.5%
2	0.4	1.0688	34.9%
4	0.2	1.4848	9.5%

Example : Richardson Extrapolation (2)

Segments	$\boldsymbol{h}$	**Integral**	ε_t
1	0.8	0.1728	89.5%
2	0.4	1.0688	34.9%
4	0.2	1.4848	9.5%

Richardson extrapolation can be used to combine these results to obtain improved estimates of the integral.

The estimates for 1 and 2 segments can be combined to yield

$I =$

$E_t =$

$\varepsilon_t =$

The estimates for 2 and 4 segments can be combined to yield

$I =$

$E_t =$

$\varepsilon_t =$

Example : Richardson Extrapolation (3)

Combine the $O(h^4)$ estimates to compute an integral with $O(h^6)$

$$I = \frac{16}{15} I_m - \frac{1}{15} I_l$$

$$=$$

The exact value of the integration $= \ 1.640533$

The Romberg Integration Algorithm

Note that the weighting factors for the Richardson extrapolation add up to 1 and that as accuracy increases, the approximation using the smaller step size is given greater weight.

In general,

$$I_{j,k} = \frac{4^{k-1} I_{j+1,k-1} - I_{j,k-1}}{4^{k+1} - 1}$$

where,

- $I_{j+1,k-1}$ and $I_{j,k-1}$ are the more and less accurate integrals
- $I_{j,k}$ is the new approximation
- k is the level of integration
  - $k = 1$ corresponds to the original trapezoidal rule estimates
  - $k = 2$ corresponds to the $O(h^4)$ estimates
  - $k = 3$ corresponds to the $O(h^6)$ estimates
- j is used to determine which approximation is more accurate
  - $j + 1$ represents the more accurate
  - j represents the less accurate

The Romberg Integration Algorithm (2)

$$I_{j,k} = \frac{4^{k-1} I_{j+1,k-1} - I_{j,k-1}}{4^{k+1} - 1}$$

For $k = 2$ and $j = 1$

$$I_{1,2} = \frac{4I_{2,1} - I_{1,1}}{3}$$

which is equivalent to

$$I = \frac{4}{3} I(h_2) - \frac{1}{3} I(h_1)$$

Romberg Algorithm Iterations

Segments	$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$
1	0.172800	1.367467		
2	1.068800			
1	0.172800	1.367467	1.640533	
2	1.068800	1.623467		
4	1.484800			
1	0.172800	1.367467	1.640533	1.640533
2	1.068800	1.623467	1.640533	
4	1.484800	1.639467		
8	1.600800			

Romberg Algorithm Iterations

Segments	$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$
1	0.172800	1.367467		
2	1.068800			
1	0.172800	1.367467	1.640533	
2	1.068800	1.623467		
4	1.484800			
	$I_{1,1}$	$I_{1,2}$	$I_{1,3}$	$I_{1,4}$
1	0.172800	1.367467	1.640533	1.640533
2	1.068800	1.623467	1.640533	
4	1.484800	1.639467	$I_{2,3}$	
8	1.600800			

Stopping criterion

- A termination criterion is required to assess the accuracy of the results.

$$|\varepsilon_a| = \left|\frac{I_{1,k} - I_{2,k-1}}{I_{1,k}}\right| \times 100\%$$

where $\varepsilon_a =$ an estimate of the percent relative error

We compare the new estimate with a previous value.

When the change between the old and new values is below a prespecified error criterion (or threshold) ε_s, the computation is terminated.

Estimates of the percent relative error

Segments	$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$
1	0.172800	1.367467		
2	1.068800			
1	0.172800	1.367467	1.640533	
2	1.068800	1.623467		
4	1.484800			
1	0.172800	1.367467	1.640533	1.640533
2	1.068800	1.623467	1.640533	
4	1.484800	1.639467		
8	1.600800			

$$|\varepsilon_a| = \left|\frac{1.367467 - 1.068800}{1.367467}\right| \times 100\% = 21.8\%$$

$$|\varepsilon_a| = \left|\frac{1.640533 - 1.623467}{1.640533}\right| \times 100\% = 1.0\%$$

$$|\varepsilon_a| = \left|\frac{1.640533 - 1.640533}{1.640533}\right| \times 100\% = 0.0\%$$

MATLAB Code for Romberg

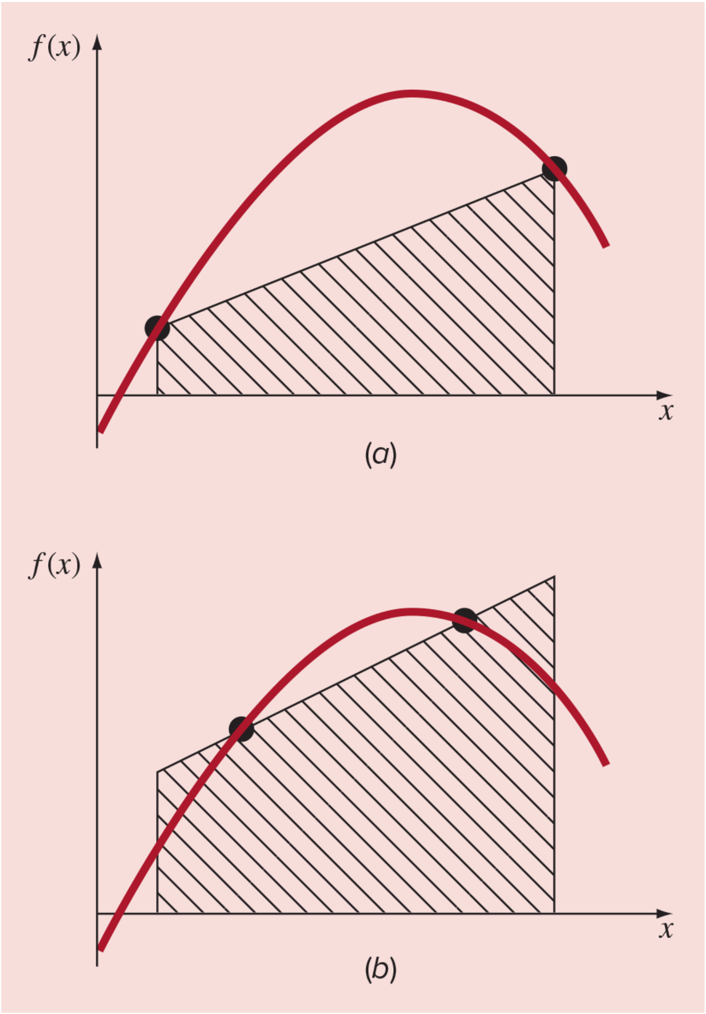
```
function [q,ea,iter] = romberg(func,a,b,es,maxit,varargin)
% romberg: Romberg integration quadrature
%  q = romberg(func,a,b,es,maxit,p1,p2,...):
%          Romberg integration.
% input:
%  func = name of function to be integrated
%  a, b = integration limits
%  es = desired relative error (default = 0.000001%)
%  maxit = maximum allowable iterations (default = 30)
%  p1,p2,... = additional parameters used by func
% output:
%  q = integral estimate
%  ea = approximate relative error (%)
%  iter = number of iterations

if nargin<3,error('at least 3 input arguments required'),end
if nargin<4|isempty(es), es=0.000001;end
if nargin<5|isempty(maxit), maxit=50;end
n = 1;
I(1,1) = trap(func,a,b,n,varargin{:});
iter = 0;
while iter<maxit
  iter = iter+1;
  n = 2^iter;
  I(iter+1,1) = trap(func,a,b,n,varargin{:});
  for k = 2:iter+1
    j = 2+iter-k;
    I(j,k) = (4^(k-1)*I(j+1,k-1)-I(j,k-1))/(4^(k-1)-1);
  end
  ea = abs((I(1,iter+1)-I(2,iter))/I(1,iter+1))*100;
  if ea<=es, break; end
end
q = I(1,iter+1);
```

Gauss Quadrature

- Gauss quadrature describes a class of techniques for evaluating the area under a straight line by joining any two points on a curve rather than simply choosing the endpoints.

- The key is to choose the line that balances the positive and negative errors.



Gauss-Legendre Formulas

- The integral estimates are of the form:

$$I \cong c_0 f(x_0) + c_1 f(x_1) + \cdots + c_{n-1} f(x_{n-1})$$

$$\cong \sum_{i=0}^{n-1} c_i f(x_i)$$

where the c_i (weighting coefficient) and x_i must be determined.

- Linear combination of function evaluation $\rightarrow$ this technique is not appropriate for cases where the function is unknown.

Two-Point Gauss-Legendre

$$I \cong c_0 f(x_0) + c_1 f(x_1)$$

We need four equations to determine c_0, c_1, x_0, x_1.

Assume the integrals of $y =$ constant, $y = x, y = x^2$, and $y = x^3$ are computed exactly.

The Gauss-Legendre formulas seem to optimize estimates to integrals for functions over intervals from −1 to 1.

Integrals over other interval from a to b require a change in variables to set the limits from -1 to 1. Or simply shift $[a\ b] \rightarrow [-1\ 1]$

Determining the Constants

$$\int_{-1}^{1} 1dx = c_0 f(x_0) + c_1 f(x_1) = x\Big|_{-1}^{1} = 1 - (-1) = 2$$

$$\int_{-1}^{1} xdx = c_0 f(x_0) + c_1 f(x_1) = \frac{x^2}{2}\Big|_{-1}^{1} = \frac{1}{2} - \frac{(-1)^2}{2} = 0$$

$$\int_{-1}^{1} x^2dx = c_0 f(x_0) + c_1 f(x_1) = \frac{x^3}{3}\Big|_{-1}^{1} = \frac{1}{3} - \frac{(-1)^3}{3} = \frac{2}{3}$$

$$\int_{-1}^{1} x^3dx = c_0 f(x_0) + c_1 f(x_1) = \frac{x^4}{4}\Big|_{-1}^{1} = \frac{1}{4} - \frac{(-1)^4}{4} = 0$$

Constants for Gauss-Legendre Formulas

Points	Weighting Factors	Function Arguments	Truncation Error
1	$c_0 = 2$	$x_0 = 0.0$	$\cong f^{(2)}(\xi)$
2	$c_0 = 1$	$x_0 = -1/\sqrt{3}$	$\cong f^{(4)}(\xi)$
	$c_1 = 1$	$x_1 = 1/\sqrt{3}$	
3	$c_0 = 5/9$	$x_0 = -\sqrt{3/5}$	$\cong f^{(6)}(\xi)$
	$c_1 = 8/9$	$x_1 = 0.0$	
	$c_2 = 5/9$	$x_2 = \sqrt{3/5}$	
4	$c_0 = (18 - \sqrt{30})/36$	$x_0 = -\sqrt{525 + 70\sqrt{30}}/35$	$\cong f^{(8)}(\xi)$
	$c_1 = (18 + \sqrt{30})/36$	$x_1 = -\sqrt{525 - 70\sqrt{30}}/35$	
	$c_2 = (18 + \sqrt{30})/36$	$x_2 = \sqrt{525 - 70\sqrt{30}}/35$	
	$c_3 = (18 - \sqrt{30})/36$	$x_3 = \sqrt{525 + 70\sqrt{30}}/35$	

Shifting the Limits of Integration

$$\int_a^b f(x)dx \Rightarrow \int_{-1}^{1} ???\,dx_d$$

Assume x_d is linearly related to x :

$$x = a_1 + a_2 x_d$$

$$a_1 = \frac{b+a}{2}, \qquad a_2 = \frac{b-a}{2}$$

$$a_1 + a_2(1) = b$$
$$a_1 + a_2(-1) = a$$

$$x = \frac{b+a}{2} + \frac{b-a}{2} x_d, \qquad dx = \frac{b-a}{2} dx_d$$

Solve for a_1 & a_2 for any specific problem

Example : Use two-point Gauss-Legendre Quadrature to estimate the integral below:

$$\int_{0}^{0.8} (0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5)dx$$

Shifting the limits of the integration $x =$ and $dx =$

$$I = \int_{0}^{0.8} (0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5)dx$$

$I_{true} = 1.640533$

A percent relative error $=$

Example : Use three-point Gauss-Legendre Quadrature to estimate the integral below:

$$\int_{0}^{0.8} (0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5)dx$$

Shifting the limits of the integration $x =$ and $dx =$

$$I = \int_{0}^{0.8} (0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5)dx$$

$I_{true} = 1.640533$

A percent relative error =

Example : Use three-point Gauss-Legendre Quadrature to estimate the integral below:

$$I = \int_{1}^{2} \left(2x + \frac{3}{x}\right)^{2} dx$$

$x =$ and $dx =$

$$I = \int_{1}^{2} \left(2x + \frac{3}{x}\right)^{2} dx$$

$I_{true} = 25.8333$

A percent relative error =

Adaptive quadrature

- Methods such as Simpson's ⅓ rule has a disadvantage in that it uses equally spaced points - if a function has regions of abrupt changes, small steps must be used over the *entire domain* to achieve a certain accuracy.

- *Adaptive quadrature* methods for integrating functions automatically adjust the step size so that small steps are taken in regions of sharp variations and larger steps are taken where the function changes gradually.

Adaptive quadrature using trapezoidal rule

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

Level 1

a --------------------------- b

$h_1 = b - a$
$I = I(h_1) + E(h_1)$

a ----------- c ----------- b

$h_2 = \frac{h_1}{2}$
$I = I(h_2) + E(h_2)$

We can prove that $E(h_1) = 4E(h_2)$, which can imply that

$$E(h_2) = \frac{1}{3}[I(h_2) - I(h_1)]$$

If $|E(h_2)| \leq tolerance$

$$I = I(h_2) + E(h_2)$$

else /* go to next level */

a ----------- c + c ----------- b

Adaptive quadrature using Simpson’s 1/3 rule

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

Level 1

a ------------- c -------------- b

$$h_1 = b - a$$
$$I = I(h_1) + E(h_1)$$

a ----- d ----- c ----- e ------ b

$$h_2 = \frac{h_1}{2}$$
$$I = I(h_2) + E(h_2)$$

We can prove that $E(h_1) = 16E(h_2)$, which can imply that

$$E(h_2) = \frac{1}{15}[I(h_2) - I(h_1)]$$

If $|E(h_2)| \leq tolerance$

$$I = I(h_2) + E(h_2)$$

else /* go to next level */

a -----d----- c + c -----e----- b

Adaptive quadrature (Matlab code, Simpson’s 1/3 rule)

```
function q = quadadapt(f,a,b,tol,varargin)

% Evaluates definite integral of f(x) from a
to b

if nargin < 4 | isempty(tol),tol = 1.e-6;end

c = (a + b)/2;

fa = feval(f,a,varargin{:});

fc = feval(f,c,varargin{:});

fb = feval(f,b,varargin{:});

q = quadstep(f, a, b, tol, fa, fc, fb,
varargin{:});

end
```

```
function q = quadstep(f,a,b,tol,fa,fc,fb,varargin)

% Recursive subfunction used by quadadapt.

h = b - a; c = (a + b)/2;

fd = feval(f,(a+c)/2,varargin{:});

fe = feval(f,(c+b)/2,varargin{:});

q1 = h/6 * (fa + 4*fc + fb);

q2 = h/12 * (fa + 4*fd + 2*fc + 4*fe + fb);

if abs(q2 - q1) <= tol

q = q2 + (q2 - q1)/15;

else

qa = quadstep(f, a, c, tol/2, fa, fd, fc, varargin{:});

qb = quadstep(f, c, b, tol/2, fc, fe, fb, varargin{:});

q = qa + qb;

end

end
```

Adaptive quadrature (Matlab code, Simpson’s 1/3 rule)

This is the version of code from textbook

```
function q = quadadapt(f,a,b,tol,varargin)
% Evaluates definite integral of f(x) from a
to b
if nargin < 4 | isempty(tol),tol = 1.e-6;end
c = (a + b)/2;
fa = feval(f,a,varargin{:});
fc = feval(f,c,varargin{:});
fb = feval(f,b,varargin{:});
q = quadstep(f, a, b, tol, fa, fc, fb,
varargin{:});
end
```

```
function q = quadstep(f,a,b,tol,fa,fc,fb,varargin)
% Recursive subfunction used by quadadapt.
h = b - a; c = (a + b)/2;
fd = feval(f,(a+c)/2,varargin{:});
fe = feval(f,(c+b)/2,varargin{:});
q1 = h/6 * (fa + 4*fc + fb);
q2 = h/12 * (fa + 4*fd + 2*fc + 4*fe + fb);
if abs(q2 - q1) <= tol
q = q2 + (q2 - q1)/15;
else
qa = quadstep(f, a, c, tol, fa, fd, fc, varargin{:});
qb = quadstep(f, c, b, tol, fc, fe, fb, varargin{:});
q = qa + qb;
end
end
```

The explanation why the tolerance parameter is not divided by 2 is given in the original document of page 6 https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/quad.pdf

Adaptive Quadrature in MATLAB

MATLAB has a built-in function for implementing adaptive quadrature:

```
q = integral(fun, a, b)
```

fun : function to be integrates.

a, b: integration bounds.