Numerical Integration of Functions

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

Reference:

1. *Applied Numerical Methods with MATLAB for Engineers*, Chapter 20 & Teaching material

Chapter Objectives

- Understanding how Richardson extrapolation provides a means to create a more accurate integral estimate by combining two less accurate estimates
- Understanding how Gauss quadrature provides superior integral estimates by picking optimal abscissas at which to evaluate the function
- Knowing how to use MATLAB's built-in functions `quad` and `quadl` to integrate functions

Richardson Extrapolation (1/2)

- ***Richard extrapolation*** methods use two estimates of an integral to compute a third, more accurate approximation.
- 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, respectively, an improved $O(h^4)$ estimate may be formed using:

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

- For the special case where the interval is halved ($h_2 = h_1/2$), this becomes:

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

Richardson Extrapolation (2/2)

- For the cases where there are two $O(h^4)$ estimates and the interval is halved ($h_m=h_l/2$), 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 ($h_m=h_l/2$), an improved $O(h^8)$ estimate may be formed using:

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

Richardson Extrapolation: An Example (1/2)

Richardson Extrapolation

Problem Statement. Use Richardson extrapolation to evaluate the integral of $f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$ from $a = 0$ to $b = 0.8$.

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

Segments	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 20.1

Richardson Extrapolation: An Example (2/2)

Richardson extrapolation can be used to combine these results to obtain improved estimates of the integral. For example, the estimates for one and two segments can be combined to yield

$$I = \frac{4}{3}(1.0688) - \frac{1}{3}(0.1728) = 1.367467$$

The error of the improved integral is $E_t = 1.640533 - 1.367467 = 0.273067 (\varepsilon_t = 16.6\%)$, which is superior to the estimates upon which it was based.

In the same manner, the estimates for two and four segments can be combined to give

$$I = \frac{4}{3}(1.4848) - \frac{1}{3}(1.0688) = 1.623467$$

which represents an error of $E_t = 1.640533 - 1.623467 = 0.017067 \, (\varepsilon_t = 1.0\%)$.

Example 20.1

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, respectively, and $i_{j,k}$ is the new approximation. k is the level of integration and j is used to determine which approximation is more accurate

Romberg Algorithm Iterations

- The chart below shows the process by which lower level integrations are combined to produce more accurate estimates:

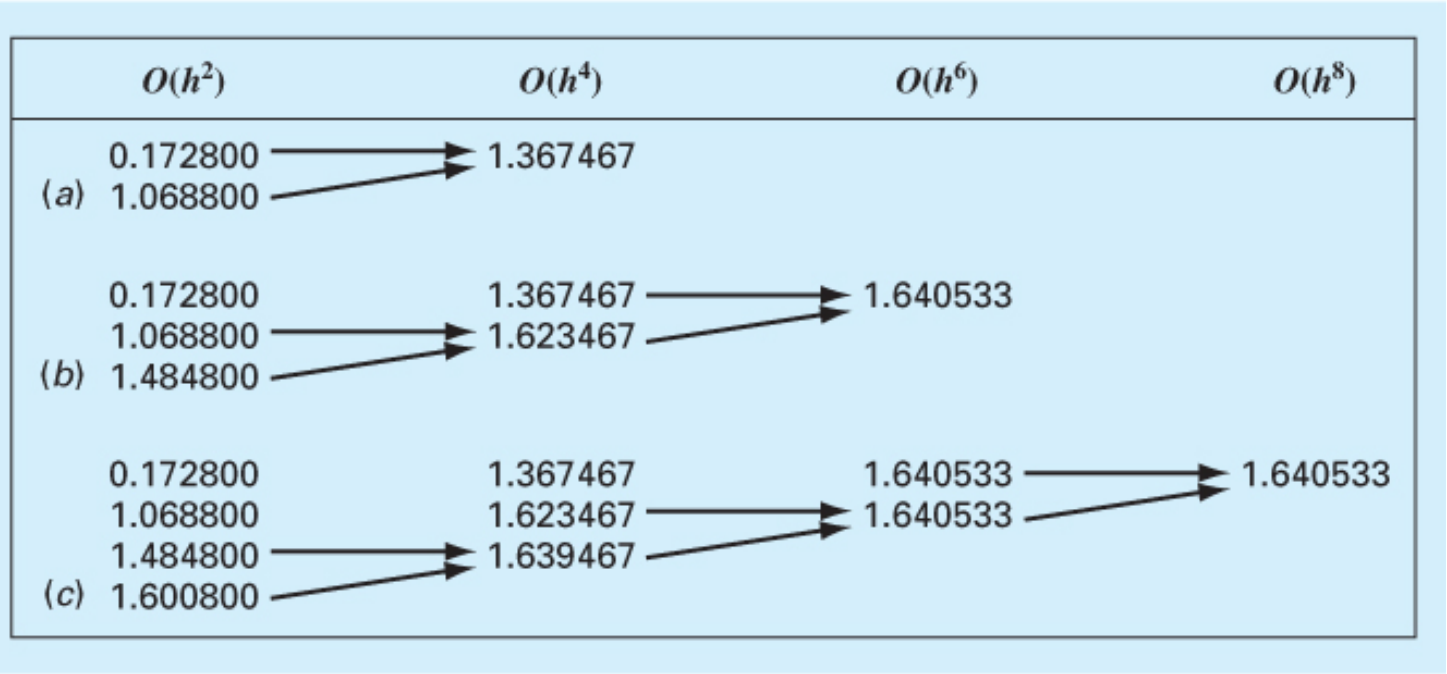

FIGURE 20.1
Graphical depiction of the sequence of integral estimates generated using Romberg integration. (*a*) First iteration. (*b*) Second iteration. (*c*) Third iteration.

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);
```

FIGURE 20.2
M-file to implement Romberg integration.

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

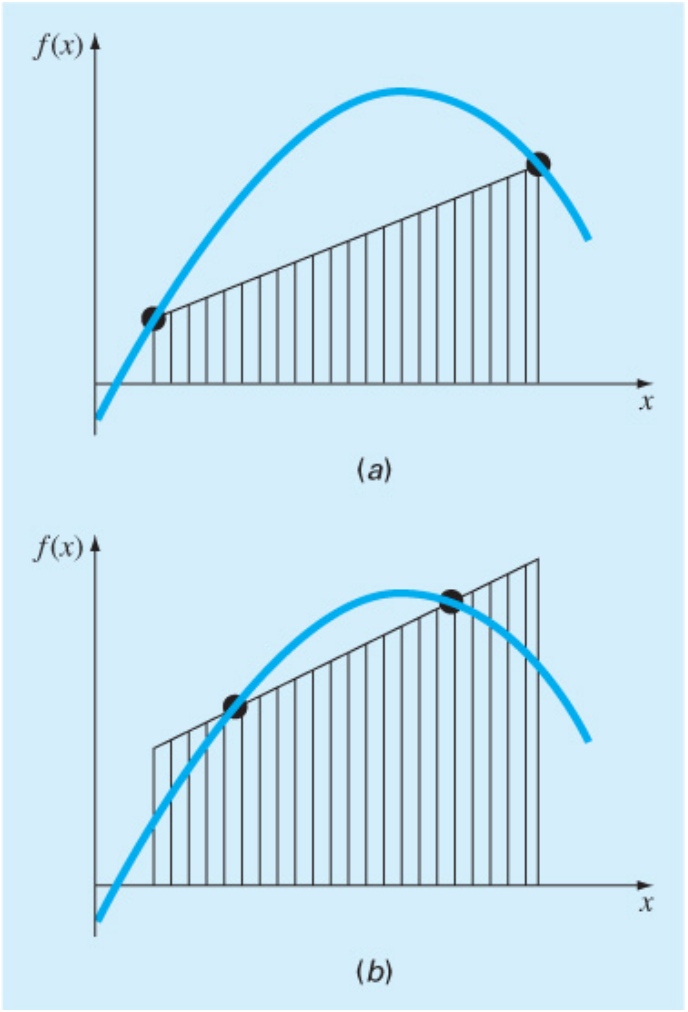

FIGURE 20.3
(*a*) Graphical depiction of the trapezoidal rule as the area under the straight line joining fixed end points. (*b*) An improved integral estimate obtained by taking the area under the straight line passing through two intermediate points. By positioning these points wisely, the positive and negative errors are better balanced, and an improved integral estimate results.

Gauss-Legendre Formulas (1/2)

- The *Gauss-Legendre* formulas seem to optimize estimates to integrals for functions over intervals from -1 to 1
- Integrals over other intervals require a change in variables to set the limits from -1 to 1
- The integral estimates are of the form:

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

  - Where the c_i and x_i are calculated to ensure that the method exactly integrates up to $(2n\text{-}1)^{th}$ order polynomials over the interval from -1 to 1

Gauss-Legendre Formulas (2/2)

TABLE 20.1 Weighting factors and function arguments used in Gauss-Legendre formulas.

Points	Weighting Factors	Function Arguments	Truncation Error
1	$c_0 = 2$	$x_0 = 0.0$	$\cong f^{(2)}(\xi)$
2	$c_0 = 1$ $c_1 = 1$	$x_0 = -1/\sqrt{3}$ $x_1 = 1/\sqrt{3}$	$\cong f^{(4)}(\xi)$
3	$c_0 = 5/9$ $c_1 = 8/9$ $c_2 = 5/9$	$x_0 = -\sqrt{3/5}$ $x_1 = 0.0$ $x_2 = \sqrt{3/5}$	$\cong f^{(6)}(\xi)$
4	$c_0 = (18 - \sqrt{30})/36$ $c_1 = (18 + \sqrt{30})/36$ $c_2 = (18 + \sqrt{30})/36$ $c_3 = (18 - \sqrt{30})/36$	$x_0 = -\sqrt{525 + 70\sqrt{30}}/35$ $x_1 = -\sqrt{525 - 70\sqrt{30}}/35$ $x_2 = \sqrt{525 - 70\sqrt{30}}/35$ $x_3 = \sqrt{525 + 70\sqrt{30}}/35$	$\cong f^{(8)}(\xi)$
5	$c_0 = (322 - 13\sqrt{70})/900$ $c_1 = (322 + 13\sqrt{70})/900$ $c_2 = 128/225$ $c_3 = (322 + 13\sqrt{70})/900$ $c_4 = (322 - 13\sqrt{70})/900$	$x_0 = -\sqrt{245 + 14\sqrt{70}}/21$ $x_1 = -\sqrt{245 - 14\sqrt{70}}/21$ $x_2 = 0.0$ $x_3 = \sqrt{245 - 14\sqrt{70}}/21$ $x_4 = \sqrt{245 + 14\sqrt{70}}/21$	$\cong f^{(10)}(\xi)$
6	$c_0 = 0.171324492379170$ $c_1 = 0.360761573048139$ $c_2 = 0.467913934572691$ $c_3 = 0.467913934572691$ $c_4 = 0.360761573048131$ $c_5 = 0.171324492379170$	$x_0 = -0.932469514203152$ $x_1 = -0.661209386466265$ $x_2 = -0.238619186083197$ $x_3 = 0.238619186083197$ $x_4 = 0.661209386466265$ $x_5 = 0.932469514203152$	$\cong f^{(12)}(\xi)$

Gauss-Legendre: An Example

Two-Point Gauss-Legendre Formula

Problem Statement. Use Eq. (20.17) to evaluate the integral of

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

between the limits $x = 0$ to 0.8. The exact value of the integral is 1.640533.

Solution. Before integrating the function, we must perform a change of variable so that the limits are from -1 to $+1$. To do this, we substitute $a = 0$ and $b = 0.8$ into Eqs. (20.22) and (20.23) to yield

$$x = 0.4 + 0.4x_d \qquad \text{and} \qquad dx = 0.4dx_d$$

Both of these can be substituted into the original equation to yield

$$\int_0^{0.8} (0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5)\,dx$$

$$= \int_{-1}^{1} [0.2 + 25(0.4 + 0.4x_d) - 200(0.4 + 0.4x_d)^2 + 675(0.4 + 0.4x_d)^3 - 900(0.4 + 0.4x_d)^4 + 400(0.4 + 0.4x_d)^5]0.4dx_d$$

Therefore, the right-hand side is in the form that is suitable for evaluation using Gauss quadrature. The transformed function can be evaluated at $x_d = -1/\sqrt{3}$ as 0.516741 and at $x_d = 1/\sqrt{3}$ as 1.305837. Therefore, the integral according to Eq. (20.17) is $0.516741 + 1.305837 = 1.822578$, which represents a percent relative error of -11.1%. This result is comparable in magnitude to a four-segment application of the trapezoidal rule or a single application of Simpson's 1/3 and 3/8 rules. This latter result is to be expected because Simpson's rules are also third-order accurate. However, because of the clever choice of base points, Gauss quadrature attains this accuracy on the basis of only two function evaluations.

Example 20.3

Adaptive Quadrature

- Methods such as Simpson's 1/3 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 in MATLAB

- MATLAB has two built-in functions for implementing adaptive quadrature:
  - `quad`: uses adaptive Simpson quadrature; possibly more efficient for low accuracies or nonsmooth functions
  - `quadl`: uses Lobatto quadrature; possibly more efficient for high accuracies and smooth functions
- *q* = quad(*fun*, *a*, *b*, *tol*, *trace*, *p1*, *p2*, …)
  - *fun* : function to be integrates
  - *a*, *b*: integration bounds
  - *tol*: desired absolute tolerance (default: 10-6)
  - *trace*: flag to display details or not
  - *p1*, *p2*, …: extra parameters for `fun`
  - `quadl` has the same arguments