Lecture 22

Numerical integration

1 Introduction

The derivative of any elementary function can be calculated explicitly as an elementary function. However, the anti-derivative of an elementary function may not be expressible as an elementary function. Therefore situations arise where the value of a definite integral

$$I=\int_a^b f(x)\,dx \tag{1}$$

must be either approximated analytically or calculated numerically, even if *f* is an elementary function. If *f* is a numerical function, or if we only have samples of *f*, then numerical integration is our only option.

There are two cases we will consider.

Case 1. We are given *n* samples of the function (x_i, y_i), $i=1,2,\ldots,n$. We cannot evaluate $f(x)$ for any other values of *x*.

Case 2. We can evaluate $f(x)$ for any value of *x*.

In Case 1 our options are somewhat limited while in Case 2 we have the freedom (and the burden) of deciding how many and which values of *x* to evaluate $f(x)$ at. We briefly deal with Case 1 in the next section and then devote the rest of the lecture to Case 2.

2 Integration of sampled data

Fig. 1 illustrates our problem in Case 1. We have *n* samples (x_i, y_i), $x_1=a<x_2<\cdots<x_n=b$. They may be uniformly spaced, with $x_{i+1}=x_i+h$, or non-uniformly spaced. A reasonable approach is to estimate $f(x)$ by interpolating these data and integrate that interpolation. In our study of interpolation techniques we learned that cubic splines provide the smoothest possible function which passes through *n* given points. A cubic function can be explicitly integrated. Therefore an attractive option is to interpolate our data with cubic splines and use the integral of the spline interpolation as an estimate of the integral of the underlying function $f(x)$. Scilab

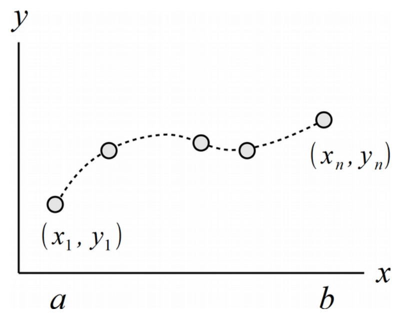

Fig. 1: Integration of sampled data by integrating an interpolated function.

provides the `intsplin` function to do just this. Its usage is

```
I = intsplin(x,y);
```

where `x,y` are the arrays of x and y samples, the integration is from x_1 to x_n, and I is the estimate of the integral. As an example, the following integral can be calculated exactly

$$I=\int_0^3 e^{-x}\sin(\pi x)dx=\frac{\pi}{\pi^2+1}(1+e^{-3})=0.3034152\ldots \qquad (2)$$

Eleven samples of $f(x)$ (Fig. 2) passed to `intsplin` estimated I with an error of less than 1%.

```
deff('y=f(x)','y=exp(-x).*sin(%pi*x)');
n = 11;
x = linspace(0,3,n);
y = f(x);
I = intsplin(x,y);
disp('I = '+string(I));

 I = 0.3057043
```

If no additional information is available, this is typically about as good as we can do with sampled data, especially if the sampling is non-uniform. However, suppose we know from the physics of the problem that $f(x)$ has to be of the form $a\,e^{-bx}\sin(cx)$. Then the best approach would be to estimate *a,b,c* using a least-squares fit and integrate $a\,e^{-bx}\sin(cx)$.

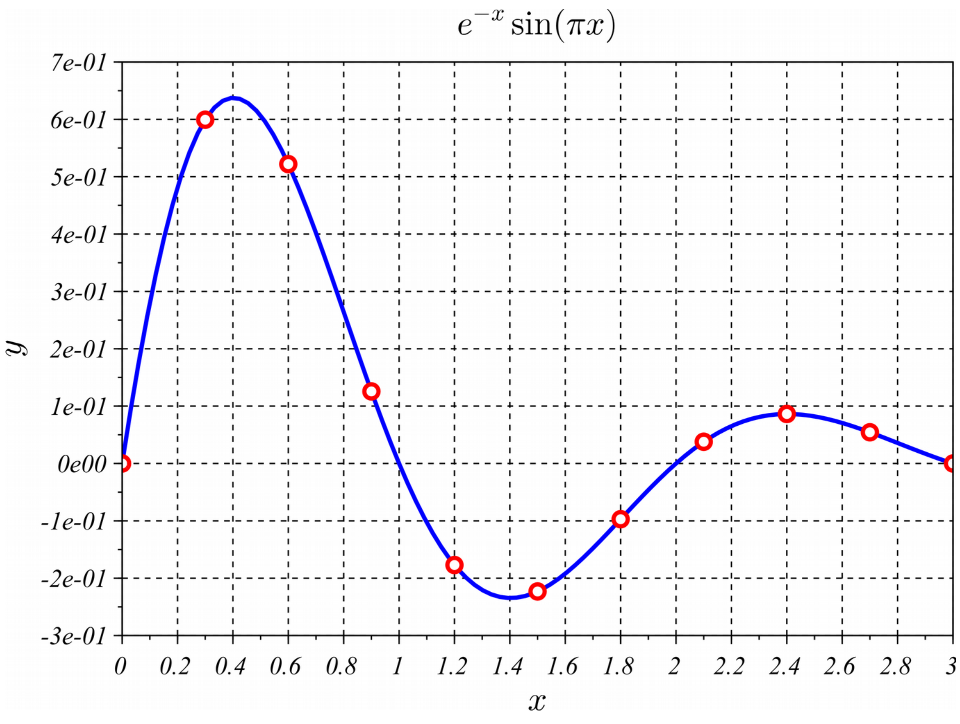

Fig. 2: Eleven samples of $f(x)=e^{-x}\sin(\pi x)$ *over* $0\le x\le 3$ *used with the* `intsplin` *function to estimate* $I=\int_0^3 f(x)dx$.

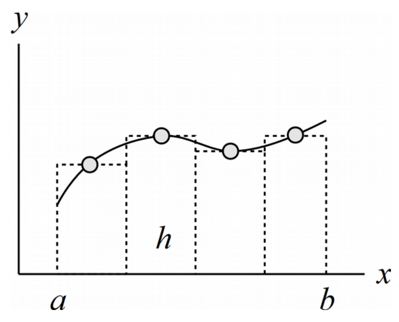

Fig. 3 Integration using the midpoint rule.

3 Midpoint (rectangle) rule

Now we turn to Case 2 where we can evaluate $f(x)$ as desired. The advantage we have over Case 1 is that we can increase the number of sample points and use the change in the value of I to estimate the error in our calculation.

The midpoint rule can be thought of as integration of a nearest-neighbor interpolation (Fig. 3). We divide the interval $a \le x \le b$ into n sub-intervals of width $h=(b-a)/n$. This width h is our *step size*. We sample $f(x)$ at the midpoint of each interval. Treating the function as a constant, the integral over one interval is $f(x)h$, the area of a rectangle of height $f(x)$ and width h. Adding the contributions of all n intervals we have

$$I \approx \sum_{i=1}^{n} f(a+[i-1/2]h)h \text{ where } h=\frac{b-a}{n} \tag{3}$$

The midpoint rule is conceptually simple. In it is nothing more than a Riemann sum such as is typically used in calculus textbooks to define a definite integral. It has the advantage that $f(x)$ is not evaluated at $x=a,b$, so it can be applied to functions which are singular at one or both endpoints, such as

$$\int_0^1 \frac{dx}{\sqrt{x}} \tag{4}$$

4 Trapezoid rule

The *trapezoid rule* approximates $f(x)$ using linear interpolation (Fig. 4). The integral is then the sum of areas of trapezoids. If the left and right heights of a trapezoid are $f(x_i)$ and $f(x_i+h)$ then the trapezoid's area is

$$I=\frac{h}{2}[f(x_i)+f(x_i+h)] \tag{5}$$

(the average height times the width). Adding up all these areas we have

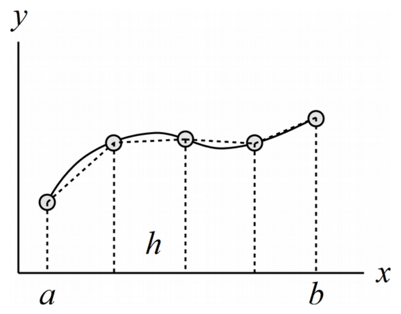

Fig. 4: Integration using the trapezoid rule.

$$I(h)=\frac{h}{2}[f(a)+f(a+h)]+\frac{h}{2}[f(a+h)+f(a+2h)]+\frac{h}{2}[f(a+2h)+f(a+3h)] \\ +\cdots+\frac{h}{2}[f(a+[n-1]h)+f(b)] \tag{6}$$

Notice that except for $f(a), f(b)$, all the function values appear twice in the sum. Therefore

$$I(h)=h\left[\frac{1}{2}f(a)+f(a+h)+f(a+2h)+\cdots+f(a+[n-1]h)+\frac{1}{2}f(b)\right] \tag{7}$$

or

$$I(h)=h\left[\frac{f(a)+f(b)}{2}+\sum_{i=1}^{n-1}f(a+ih)\right] \tag{8}$$

is the formula for integration by the trapezoid method.

It is highly desirable to have some idea of the accuracy of a numerical integration. A reasonable error estimate can be obtained by comparing one integration to a second with twice as many samples (half the step size)

$$\epsilon_{est}=|I(h/2)-I(h)|$$

In order to achieve a desired tolerance then we can iteratively halve *h* until this error estimate is less than the tolerance.

Suppose $a=0, b=1$ and our first iteration of the trapezoid rule uses $h=1$. Then we will need samples of *f* at $x=0, 1$. As shown in Fig. 5, if for the second iteration we take $h=1/2$ we will need samples at $x=0/2, 1/2, 2/2$. But we will have already sampled $x=0=0/2, 1=2/2$. If the third iteration has $h=1/4$ we will need samples at $x=0/4, 1/4, 2/4, 3/4, 4/4$, but we will have already sampled at $x=0/4=0/2=0, 2/4=1/2, 4/4=2/2=1$. In fact at each iteration only the odd-numbered samples are new. If at the fourth iteration we take $h=1/8$, we only need to sample at $x=1/8, 3/8, 5/8, 7/8$. In other words, if we break the sum in (8) into even and odd samples

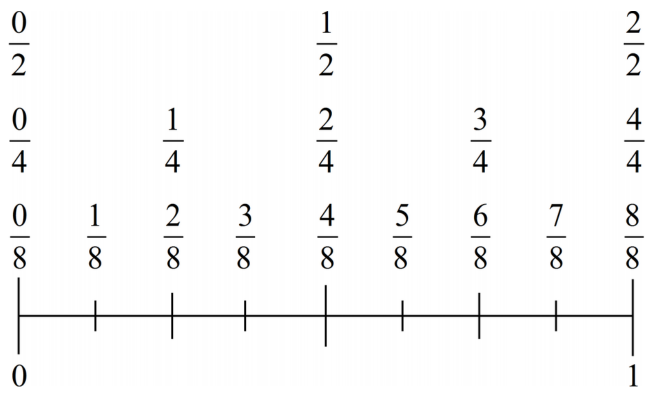

Fig. 5: Repeating the trapezoid rule at 1/2 the step size results in all "even" samples being identical to samples from the previous calculation.

$$\sum_{i=1}^{n-1} f(a+ih)=\sum_{\substack{i=1\\ \text{even}}}^{n-1} f(a+ih)+\sum_{\substack{i=1\\ \text{odd}}}^{n-1} f(a+ih) \tag{9}$$

the sum of even samples has already been calculated in the previous iteration. We only need to multiply the previous iteration value by 1/2 (since h is being halved) and add in the new (odd) samples

$$I(h)=\frac{1}{2}I(2h)+h\sum_{\substack{i=1\\ \text{odd}}}^{n-1} f(a+ih) \tag{10}$$

Our complete algorithm then reads

```
Algorithm for trapezoid-rule integration with error estimate
  $n=1,\ h=(b-a),\ I=\frac{h}{2}[f(a)+f(b)]$
  repeat until converged
    $I_{old}=I,\ n\leftarrow 2n,\ h=(b-a)/n$
    $I=\frac{1}{2}I_{old}+h\sum_{\substack{i=1\\ \text{odd}}}^{n-1} f(a+ih)$
    converged if $|I-I_{old}|<tol$
```

A Scilab implementation of this is given in the Appendix as function `intTrapezoid`. The trapezoid rule is second-order accurate, that is, the error varies as h^2. Halving the step size reduces the error by a factor of 1/4.

Example 1. $I=\int_0^3 e^{-x}\sin(\pi x)\,dx$ estimated with `intTrapezoid` with desired error of no more than 10^{-3}.

```
deff('y=f(x)','y=exp(-x)*sin(%pi*x)');
Iex = (%pi/(%pi^2+1))*(1+exp(-3)); //exact value
I = intTrapezoid(f,0,3,1e-3);
disp([Iex,I]);

0.3034152    0.3032642
```

The integration is indeed accurate to three decimal places. This required 129 function evaluations.

5 Simpson's rule

Simpson's rule integrates a quadratic interpolation of groups of three sampled function values. Suppose we want to estimate

$$I=\int_a^b f(x)\,dx \tag{11}$$

using $h=(b-a)/2$ and the three samples $y_1=f(a), y_2=f(a+h), y_3=f(a+2h=b)$. Let $x=a+th$. Then $a\le x\le b$ corresponds to $0\le t\le 2$ and $dx=h\,dt$ so that

$$I=h\int_0^2 f(a+th)\,dt \tag{12}$$

Representing $f(a+th)$ by the Lagrange interpolating polynomial through the points $(t=0,y_1),(t=1,y_2),(t=2,y_3)$ and integrating we find

$$I=h\int_0^2\left[\frac{1}{2}y_3t(t-1)-y_2t(t-2)+\frac{1}{2}y_1(t-1)(t-2)\right]dt=\frac{h}{3}(y_1+4y_2+y_3) \tag{13}$$

To apply this result in general (Fig. 6) we arrange our samples $x_1,x_2,x_3,x_4,\ldots$ into adjacent groups of three

$$(x_1,x_2,x_3),(x_3,x_4,x_5),(x_5,x_6,x_7),\ldots \tag{14}$$

(this only works if we have an odd number of samples, which implies an even number of intervals). We then apply (13)

$$I=\frac{h}{3}\left[(y_1+4y_2+y_3)+(y_3+4y_4+y_5)+(y_5+4y_6+y_7)+\cdots+(y_{n-2}+4y_{n-1}+y_n)\right] \tag{15}$$

Notice that samples at group boundaries, such as y_3 and y_5, appear twice in the summation. Therefore

$$I=\frac{h}{3}\left[y_1+4y_2+2y_3+4y_4+2y_5+4y_6+2y_7+\cdots+2y_{n-2}+4y_{n-1}+y_n\right] \tag{16}$$

Simpson's rule is fourth-order accurate (error varies as h^4). Simpson's rule applied to the eleven

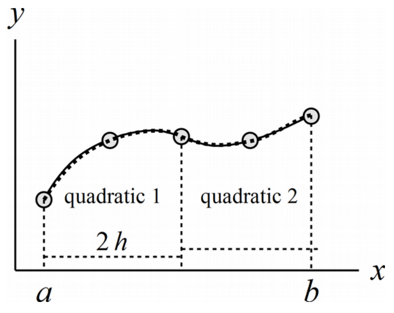

Fig. 6: Simpson's rule integration.

samples of Fig. 2 gave the estimate $I=0.3044273$.

Rules based on cubic interpolation of four-point groups (Simpson's 3/8 rule), quartic interpolation of five-point groups (Boole's rule), and even high-order interpolations exist. Together these are referred to as *Newton-Cotes formulas*. These formulas were of considerable interest in the past when calculations had to be done by hand.

An interesting way to view Simpson's rule is as follows. Suppose we have two samples of a function: $y_1=f(a)$ and $y_3=f(b)$. The trapezoid rule with $h=(b-a)$ gives

$$I_1=\frac{b-a}{2}[y_1+y_3] \tag{17}$$

Suppose we add a sample between the other two: $y_2=f((a+b)/2)$. The trapezoid rule with $h=(b-a)/2$ applied to these three samples gives us

$$I_2=\frac{b-a}{4}[y_1+2y_2+y_3] \tag{18}$$

Now

$$\begin{aligned}\frac{4I_2-I_1}{4-1}&=\frac{b-a}{3}\left[y_1+2y_2+y_3-\frac{1}{2}(y_1+y_3)\right]=\frac{b-a}{3}\left[\frac{1}{2}y_1+2y_2+\frac{1}{2}y_3\right]\\&=\frac{b-a}{6}[y_1+4y_2+y_3]\end{aligned} \tag{19}$$

is Simpson's rule with $h=(b-a)/2$. We see that if we have one trapezoid-rule estimate I_1 using a step size $2h$ and a second I_2 using step size *h*, then Simpson's rule with step size *h* can be calculated as

$$I=\frac{4I_2-I_1}{3} \tag{20}$$

Simpson's rule can be thought of as a weighted combination of trapezoid rules with different step sizes. This idea is generalized by *Romberg integration*.

6 Romberg integration

Neglecting round-off error, the trapezoid rule would (in principle) produce an exact result in the limit $h \to 0$. Let's call the exact result I_0. For arbitrary *h* let's denote the trapezoid-rule estimate by $I(h)$. Then $I_0 = I(0)$. For small but finite *h*, $I(h)$ will equal the exact result plus some error, and the error will be a function of the step size *h*. We can write

$$I(h) = I_0 + a h^2 + b h^4 + \cdots \tag{21}$$

(It can be shown that the error is an even function of *h* and therefore involves only even powers.) Romberg integration is a technique that allows us to subtract off the error terms $a h^2, b h^4, \ldots$.

Applying the trapezoid rule with step size $h/2$ we get

$$I(h/2) = I_0 + a h^2/4 + b h^4/16 + \cdots \tag{22}$$

We don't know the value of the coefficient *a*, so we don't know the first error terms in (21) and (22). However, we do know that for any value of *a*

$$4(a h^2/4) = a h^2 \tag{23}$$

This allows us to write

$$\frac{4 I(h/2) - I(h)}{4-1} = I_0 - b h^4/4 + \cdots \tag{24}$$

We have just removed the h^2 error term! Two second-order accurate trapezoid-rule calculations have been combined to produce a fourth-order accurate result. In fact, as we saw above, this is just Simpson's rule.

Now run the trapezoid rule with step size $h/4$ to get

$$I(h/4) = I_0 + a h^2/16 + b h^4/256 + \cdots \tag{25}$$

Once again the h^2 error term is $1/4$ the value of the previous iteration, and we can calculate

$$\frac{4 I(h/4) - I(h/2)}{4-1} = I_0 - b h^4/64 + \cdots \tag{26}$$

Now we have two results, (24) and (26), that are fourth-order accurate (both are Simpson's rule calculations). Furthermore, notice that although we don't know the value of the coefficient *b*, we do know that

$$16(b h^4/64) = b h^4/4 \tag{27}$$

Therefore

$$\frac{4^2(I_0 - b h^4/64) - (I_0 - b h^4/4)}{4^2 - 1} = I_0 + \cdots \tag{28}$$

and we have eliminated both the h^2 and h^4 error terms! This result is sixth-order accurate. We can continue on in this manner to produce a result accurate to as high an order as we wish.

Here is a useful notation that will allow us to easily code Romberg integration. Define

$$R(j,1) = I\left(\frac{b-a}{2^{j-1}}\right) \tag{29}$$

so that $R(1,1) = I(b-a)$, $R(2,1) = I((b-a)/2)$, $R(3,1) = I((b-a)/4)$ and so on. Stack these into a one-dimensional array

$$\begin{bmatrix} R(1,1) \\ R(2,1) \\ R(3,1) \end{bmatrix} \tag{30}$$

Now calculate

$$R(2,2)=\frac{4R(2,1)-R(1,1)}{4-1} \tag{31}$$

and

$$R(3,2)=\frac{4R(3,1)-R(2,1)}{4-1} \tag{32}$$

Just as for (24) and (26), $R(2,2)$ and $R(3,2)$ will be fourth-order accurate results, lacking the h^2 error term. Place these in the second column of our array

$$\begin{bmatrix} R(1,1) & 0 \\ R(2,1) & R(2,2) \\ R(3,1) & R(3,2) \end{bmatrix} \tag{33}$$

Now calculate

$$R(3,3)=\frac{4^2 R(3,2)-R(2,2)}{4^2-1} \tag{34}$$

As for (28) this will be sixth-order accurate, lacking both the h^2 and h^4 terms. Place this in the third column of our array

$$\begin{bmatrix} R(1,1) & 0 & 0 \\ R(2,1) & R(2,2) & 0 \\ R(3,1) & R(3,2) & R(3,3) \end{bmatrix} \tag{35}$$

The relation between an element in the k^{th} column and the elements in the previous column is

$$R(j,k)=\frac{4^{(k-1)}R(j,k-1)-R(j-1,k-1)}{4^{(k-1)}-1} \tag{36}$$

Suppose we want an eight-order accurate result. Calculate $R(4,1)=I((b-a)/8)$ and then use formula (36) to calculate $R(4,2), R(4,3), R(4,4)$ to obtain

$$\begin{bmatrix} R(1,1) & 0 & 0 & 0 \\ R(2,1) & R(2,2) & 0 & 0 \\ R(3,1) & R(3,2) & R(3,3) & 0 \\ R(4,1) & R(4,2) & R(4,3) & R(4,4) \end{bmatrix} \tag{37}$$

Our eight-order accurate estimate is $R(4,4)$. We can continue to add rows in this manner as many times as desired. The difference $R(4,4)-R(3,3)$ provides an error estimate. Adding these calculations to the trapezoid-rule algorithm results in

Algorithm for Romberg integration with error estimate

```
$n=1\,,\ h=(b-a)\,,\ I=\frac{h}{2}[f(a)+f(b)]\,,\ j=1\,,\ R(j,1)=I$
repeat until converged
    $I_{old}=I\,,\ n\leftarrow 2n\,,\ h=(b-a)/n$
    $I=\frac{1}{2}I_{old}+h\sum_{\substack{i=1\\ \text{odd}}}^{n-1} f(a+ih)$
    $j\leftarrow j+1\,,\ R(j,1)=I$
    for k=2,3,...,j
        $R(j,k)=\frac{w\,R(j,k-1)-R(j-1,k-1)}{w-1}\,,\ w=4^{k-1}$
converged if $|R(j,j)-R(j-1,j-1)|<tol$
```

A Scilab implementation of this appears in the Appendix as `intRomberg`.

Example 2. $I=\int_0^3 e^{-x}\sin(\pi x)dx$ estimated with `intTrapezoid` and `intRomberg` with desired error of no more than 10^{-6}.

```
global nCalls

function y = f(x)
  global nCalls
  nCalls = nCalls+1;
  y=exp(-x)*sin(%pi*x);
endfunction

Iex = (%pi/(%pi^2+1))*(1+exp(-3)); //exact result

nCalls = 0;
I = intTrapezoid(f,0,3,1e-6);
disp([Iex,I,nCalls]);

nCalls = 0;
I = intRomberg(f,0,3,1e-6);
disp([Iex,I,nCalls]);
```

```
    0.3034152    0.3034151    4097.
    0.3034152    0.3034152    65.
```

4097 function calls were required by the trapezoid rule while only 65 were needed for Romberg integration. The trapezoid rule error was $1.5\cdot10^{-7}$ while the Romberg integration error was $7.2\cdot10^{-11}$.

7 Gaussian quadrature

"Quadrature" is an historic term used to describe integration. So far we've assumed $f(x)$ is uniformly sampled along the *x* axis. The idea behind Gaussian quadrature is to consider arbitrarily placed *x* samples. To see why this is a good idea consider the following function

$$f(x)=a+b\,x^2+c\,x^4 \tag{38}$$

and its integral

$$I=\int_{-1}^{1} f(x)\,dx=2\left(a+\frac{1}{3}b+\frac{1}{5}c\right) \tag{39}$$

(Note: the integral of an odd power of *x* over $-1\le x\le 1$ vanishes, hence we don't bother to include odd powers in $f(x)$.) Suppose we are allowed to estimate *I* using three samples of $f(x)$. We could use Simpson's rule to get

$$I_{Simpson}=\frac{1}{3}[f(-1)+4f(0)+f(1)]=\frac{1}{3}[(a+b+c)+4a+(a+b+c)]=2\left(a+\frac{1}{3}b+\frac{1}{3}c\right) \tag{40}$$

The *a* and *b* terms are correct but the *c* term is not. This is not surprising since Simpson's rule interpolates the three samples with a quadratic. This is exact for a quadratic function, but the presence of the x^4 results in error. Now consider the following combination of three $f(x)$ samples

$$\begin{aligned}I_{Gauss}&=\frac{1}{9}\left[5f\left(-\sqrt{\frac{3}{5}}\right)+8f(0)+5f\left(\sqrt{\frac{3}{5}}\right)\right]=\frac{1}{9}\left[\left(5a+3b+\frac{9}{5}c\right)+(8a)+\left(5a+3b+\frac{9}{5}c\right)\right]\\&=2\left(a+\frac{1}{3}b+\frac{1}{5}c\right)\end{aligned} \tag{41}$$

This result is exact, $I_{Gauss}=I$, even though it required only three samples. It turns out that if you properly choose the *n* sample points x_i and corresponding *weights* w_i you can make

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

for $f(x)$ an arbitrary polynomial of order $2n-1$. The x_i turn out to be roots of certain polynomials, and the formulas for the x_i and w_i are fairly involved.

8 The intg (Scilab) and quadgk (Matlab) functions

Scilab and Matlab provide state-of-the-art numerical integration functions. Both are based on a version of Guassian quadrature called Gauss–Kronrod quadrature. In Scilab $I=\int_a^b f(x)\,dx$ is estimated using

```
    I = intg(a,b,f);
```

The default tolerance is very small. To specify the tolerance use

```
    [I,err] = intg(a,b,f,tol);
```

Optional output variable `err` is an estimate of the absolute error. In Matlab the corresponding function is

```
I = quadgk(f,a,b);
```

To use `intg` in the console you first need to define a function $f(x)$. For example

```
-->deff('y=f(x)','y=exp(-x)*sin(%pi*x)');
-->I = intg(0,3,f)
 I  =
 0.3034152
```

Example 3. $I=\int_0^3 e^{-x}\sin(\pi x)\,dx$ was estimated in Scilab using

```
deff('y=f(x)','y=exp(-x)*sin(%pi*x)');
I = intg(0,3,f,1e-6)
I  =
    0.3034152
```

Only 21 function calls were required to produce a result with error of “0,” i.e., the exact result and the `intg` result were equal to within double precision accuracy.

The `integrate` function convenient allows you to skip the `deff` statement as it accepts the function and variable of integration as string arguments

```
-->I = integrate('exp(-x)*sin(%pi*x)','x',0,3)
 I  =
    0.3034152
```

9 Improper integrals

An integral is improper if the integrand has a singularity within the integration interval. For example

$$\int_0^1 \frac{\sin x}{x}\,dx \tag{42}$$

Here the integrand in undefined at $x=0$ where it has a $0/0$ form. In fact $\lim_{x\to 0}\frac{\sin x}{x}=1$, so one could “define away” the problem with

$$f(x)=\begin{cases}\frac{\sin x}{x} & x\neq 0\\ 1 & x=0\end{cases} \tag{43}$$

On the other hand the integrand of

$$\int_0^1 \frac{1}{\sqrt{1-x^2}}\,dx=\frac{\pi}{2} \tag{44}$$

becomes infinite as $x\to 1$. In either case a solution would be an integration technique that avoids evaluating the function at the endpoints. The midpoint rule is a simple example of this type of so-called “open” integration formula. The Gauss–Kronrod quadrature method used by `intg` and `quadgk` is also an open formula and will work for functions with singularities at one or both endpoints. For example

```
-->integrate('sin(x)/x','x',0,1)
 ans  =

    0.9460831

-->integrate('1/sqrt(1-x^2)','x',0,1)
 ans  =

    1.5707963
```

For a singularity at $x=c$, $a<c<b$, we can break the integral into two

$$\int_a^b f(x)\,dx=\int_a^c f(x)\,dx+\int_c^b f(x)\,dx$$

Another type of improper integral is one with an infinite limit of integration, such as

$$\int_0^\infty x^3 e^{-x}\,dx=6 \tag{45}$$

One way to treat an integral of this type is by using a change of variable such as

$$x=-\ln(1-u) \tag{46}$$

For $u=0\,,x=0$ and as $u\to 1\,,x\to\infty$. The differential is

$$dx=\frac{du}{1-u} \tag{47}$$

Conveniently $e^{-x}=1-u$, so that $e^{-x}\,dx=du$, and the integral becomes

$$\int_0^1 (-\ln(1-u))^3\,du \tag{48}$$

This is also improper because as $u\to 1\,,x=-\ln(1-u)\to\infty$ but an open integration formula can evaluate it

```
-->integrate('(-log(1-u))^3','u',0,1)
 ans  =

    6.
```

10 References

1. http://en.wikipedia.org/wiki/Numerical_integration

11 Appendix – Scilab code

11.1 Trapezoid rule

```
//////////////////////////////////////////////////////////////////////
// intTrapezoid.sci
// 2014-11-15, Scott Hudson, for pedagogic purposes
// Trapezoid rule estimation of integral of f(x) from a to b
// Estimated error is <= tol
//////////////////////////////////////////////////////////////////////
function I=intTrapezoid(f, a, b, tol)
  n = 1;
  h = (b-a);
  I = (h/2)*(f(a)+f(b));
  converged = 0;
  while (~converged)
    Iold = I;
    n = 2*n;
    h = (b-a)/n;
    I = 0;
    for i=1:2:n-1 //i=1,3,5,... odd values
      I = I+f(a+i*h);
    end
    I = 0.5*Iold+h*I;
    if (abs(I-Iold)<=tol)
      converged = 1;
    end
  end
endfunction
```

11.2 Romberg integration

```
//////////////////////////////////////////////////////////////////////
// intRomberg.sci
// 2014-11-15, Scott Hudson, for pedagogic purposes
// Romberg integration of f(x) from a to b
// Estimated error is <= tol
//////////////////////////////////////////////////////////////////////
function I=intRomberg(f, a, b, tol)
  n = 1;
  h = (b-a);
  I = (h/2)*(f(a)+f(b));
  j = 1;
  R(j,j) = I;
  converged = 0;
  while (~converged)
    Iold = I;
    n = 2*n;
    h = (b-a)/n;
    I = 0;
    for i=1:2:n-1 //i=1,3,5,... odd values
      I = I+f(a+i*h);
    end
    I = 0.5*Iold+h*I;
    j = j+1;
    R(j,1) = I;
    for k=2:j
      w = 4^(k-1);
      R(j,k) = (w*R(j,k-1)-R(j-1,k-1))/(w-1);
    end
    if (abs(R(j,j)-R(j-1,j-1))<=tol)
      converged = 1;
    end
  end
  I = R(j,j);
endfunction
```