Applied Mathematics, 2014, 5, 1412-1426
Published Online June 2014 in SciRes. http://www.scirp.org/journal/am
http://dx.doi.org/10.4236/am.2014.510133

Behavior of the Numerical Integration Error

Tchavdar Marinov[*], Joe Omojola, Quintel Washington, LaQunia Banks

Department of Natural Sciences, Southern University at New Orleans, New Orleans, USA
Email: [*]tmarinov@suno.edu

Received 11 February 2014; revised 11 March 2014; accepted 18 March 2014

Abstract

In this work, we consider different numerical methods for the approximation of definite integrals. The three basic methods used here are the Midpoint, the Trapezoidal, and Simpson's rules. We trace the behavior of the error when we refine the mesh and show that Richardson's extrapolation improves the rate of convergence of the basic methods when the integrands are sufficiently differentiable many times. However, Richardson's extrapolation does not work when we approximate improper integrals or even proper integrals from functions without smooth derivatives. In order to save computational resources, we construct an adaptive recursive procedure. We also show that there is a lower limit to the error during computations with floating point arithmetic.

Keywords

Numerical Integration, Algorithms with Automatic Result Verification, Roundoff Error

1. Introduction

Suppose $f(x)$ is a real function of the real variable x, defined for all $x \in [a,b]$. The definite integral from the function $f(x)$ from a to b is defined as the limit

$$I = \int_a^b f(x)\,\mathrm{d}x = \lim_{\max_k \Delta_k \to 0} \sum_{k=1}^{n} f(x_k)\Delta_k, \tag{1}$$

where $a = x_0 < x_1 < \cdots < x_n = b$ and $\Delta_k = x_k - x_{k-1}, k = 1,2,\cdots,n.$ In the case when the anti-derivative $F(x)$ of $f(x)$ is known, we can calculate the definite integral using the fundamental theorem of Calculus:

$$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a). \tag{2}$$

[*]Corresponding author.

The anti-derivatives of many functions cannot be expressed as elementary functions. Examples include

$$e^{-x^2}, \sin x^2, \cos x^2, \frac{\sin x}{x}, \frac{\cos x}{x}, \frac{1}{\ln x}. \tag{3}$$

In some cases, when the function is obtained as a result of numerical calculations or experimental observations, the function values are available only for a fixed, finite set of points $x_i \in [a,b]$ (see for example [1] and [2]). In those cases the Fundamental Theorem of Calculus cannot be applied and the values of the integral have to be approximated numerically.

It is clear that when using numerical calculations, due to the round-off error, we usually cannot obtain the exact value I of the integral. So, we say that the numerical value of the integral I is obtained when we obtain $\tilde{I}$ *approximate value* of I, such that $|I-\tilde{I}|<\varepsilon$, where $\varepsilon>0$ is the acceptable error. Even in some cases when the Fundamental Theorem of Calculus is applicable, we still have to work with approximate value of the integral. For example

$$I = \int_0^1 e^x \, dx = e - 1 = 1.7172\cdots \tag{4}$$

Since e is a transcendent number we cannot represent it as a finite decimal number. So, we have to use an approximation of the integral.

Usually, the approximate value of the integral is calculated using the so-called *formulas for numerical integration* or *quadrature formulas*. The general idea is to approximate the integral as a sum

$$I = \int_a^b f(x)\mathrm{d}x \approx \tilde{I} = \sum_{k=0}^{n} A_k f(x_k) \tag{5}$$

of the values of the function $f(x)$, calculated at some nodes x_k, multiplied by some coefficient A_k, called *weights* or *weight coefficients*. The nodes and the weight coefficients do not depend on the function $f(x)$.

Usually the nodes and the weight coefficients are calculated so that the formula is exact for the polynomials of highest possible degree. We say that the quadrature formula is *exact for all polynomials of degree* m if for any polynomial $P_m(x)$ of degree m

$$\int_a^b P_m(x)\mathrm{d}x = \sum_{k=0}^{n} A_k P_m(x_k). \tag{6}$$

In this work we consider so called Newton-Cotes formulas defined on a fixed set of nodes $x_i \in [a,b]$, $i = 0,1,2,\cdots,n$ (see [3] for details). In this case, in order to obtain the coefficients A_i, one has to solve the system of linear equations

$$\int_a^b x^l \, \mathrm{d}x = \sum_{k=0}^{n} A_k x_k^l, \qquad l = 0,1,2,\cdots,m. \tag{7}$$

The Error Term

The error term, $R(f)$, is defined as

$$R(f) = \int_a^b f(x)\mathrm{d}x - \sum_{k=0}^{n} A_k x_k^l. \tag{8}$$

If the function $f(x)$ has a continuous m derivative, the error term can be represented in the form (see for details [3] [4])

$$R(f) = c(b-a)^{m+1} f^{(m)}(\xi), \tag{9}$$

where c is a constant and $\xi \in (a,b)$.

The degree of precision of a quadrature formula is defined as the positive integer m satisfying $R(P_k)=0$ for $k = 0,1,\cdots,m$, where P_k is a polynomial of degree k, but $R(P_{m+1}) \neq 0$ for some polynomial of degree $(m+1)$.

2. Midpoint Rule

The simplest formula, using only one node, is the *Midpoint Rule*. Suppose the node is the midpoint $x_0 = \frac{b-a}{2}$ of the interval $[a,b]$. By solving the Equation (7), we obtain the formula

$$\tilde{I} = (b-a) f\left(\frac{b+a}{2}\right) \tag{10}$$

If the function $f(x)$ has a continuous second derivative, the error term of the Midpoint Rule can be represented in the form

$$R(f) = \frac{(b-a)^3}{24} f''(\xi), \tag{11}$$

where $\xi \in (a,b)$.

2.1. Composite Midpoint Rule

In order to reduce the error term we partition the interval $[a,b]$ into N subintervals $[x_i, x_{i+1}]$, $i = 0,1,2,\cdots,N$. Assume $x_0 = a$, $x_N = b$, and $x_0 < x_1 < \cdots < x_N$ (see **Figure 1**). Suppose the length of each subinterval is the same, $x_{i+1} - x_i = h$. So, $h = \frac{b-a}{N}$ and $x_i = a + ih$. Then,

$$\int_a^b f(x)\mathrm{d}x \approx h\sum_{k=1}^{N} f(x_k - h/2). \tag{12}$$

If the function $f(x)$ has a continuous second derivative in $[a,b]$, the approximation error

$$R(f,N) = \int_a^b f(x)\mathrm{d}x - h\sum_{k=1}^{N} f(x_k - h/2) = I - I_N, \tag{13}$$

is given as

$$R(f,N) = \frac{(b-a)^3}{24N^2} f''(\xi) = cf''(\xi)h^2, \qquad \xi \in (a,b), \tag{14}$$

where $c = \frac{b-a}{24}$.

Suppose we have performed two calculations with different numbers of subintervals, N and $2N$. Then

$$R(f,N) = I - I_N = cf''(\xi)h^2, \tag{15}$$

and

$$R(f,2N) = I - I_{2N} = cf''(\xi')\frac{h^2}{4}. \tag{16}$$

If $f''(\xi) \approx f''(\xi')$, we obtain

$$E = \frac{R(f,N)}{R(f,2N)} = 4. \tag{17}$$

h
$x_0 = a$
x_1
x_2
x_3
$\cdots$
x_{N-3}
x_{N-2}
x_{N-1}
$x_N = b$

Figure 1. The mesh.

Therefore, if we double the number of subintervals, the error term is decreased four times. This is a practical way of confirming the power of h in the error term. In practice [5], the ratio is not exactly four, because f'' is usually not constant.

2.2. Numerical Experiments

We prepared a SAGE function MPloop for our numerical experiments with the Midpoint Rule. The code is given here:

```
sage: def MPloop(a,b,n):
...       h=(b-a)/n # the step
...       AI = 0
...       for i in [1..n]:
...           AI = AI + f(a+(i-0.5)*h)
...       AI = float(AI*h)
...       return AI
```

The variables *a* and *b* are the lower and upper limits of the integral, and the variable *n* is the number of subintervals.

The results obtained with the function MPloop for the integral

$$I = \int_{-1}^{2}\left(4x^3 + 2x\right)\mathrm{d}x = 18 \tag{18}$$

are shown in **Table 1**. The last column shows the ratio of the errors $\frac{R(f,N)}{R(f,2N)}$. Our numerical calculations confirm that if we double the number of subintervals, the error term is decreased four times. The error distribution for the same calculation is given in **Figure 2**. The red dots are the numerical error and the blue line is the graph of the function $y = \frac{-13.5}{N^2}$. Again, the numerical experiments confirm that the error is going to zero when N is going to infinity the same way as the function $y = \frac{-13.5}{N^2}$ is going to zero. One application of the Midpoint Rule can be found in [6].

3. Trapezoidal Rule

The formula based on two nodes, $x_0 = a$ and $x_1 = b$, and exact for all second order polynomials, can be calculated via the system of two Equations (7). The resulting formula

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{2}\left[f(a) + f(b)\right], \tag{19}$$

is called *Trapezoidal Rule*, because it produces exactly the area of the trapezoid with vertexes $(a,0)$, $(b,0)$, $(a, f(a))$, and $(b, f(b))$.

If the function $f(x)$ has a continuous second derivative, the error term of the Trapezoidal Rule can be represented in the form (see [3])

$$R(f) = -\frac{(b-a)^3}{12} f''(\xi), \tag{20}$$

where $\xi \in (a,b)$.

3.1. Composite Trapezoidal Rule

The composite formula for the Trapezoidal Rule is

$$\int_a^b f(x)\mathrm{d}x \approx h\left[\frac{f(a)+f(b)}{2} + f(x_1) + f(x_2) + \cdots + f(x_{N-1})\right]. \tag{21}$$

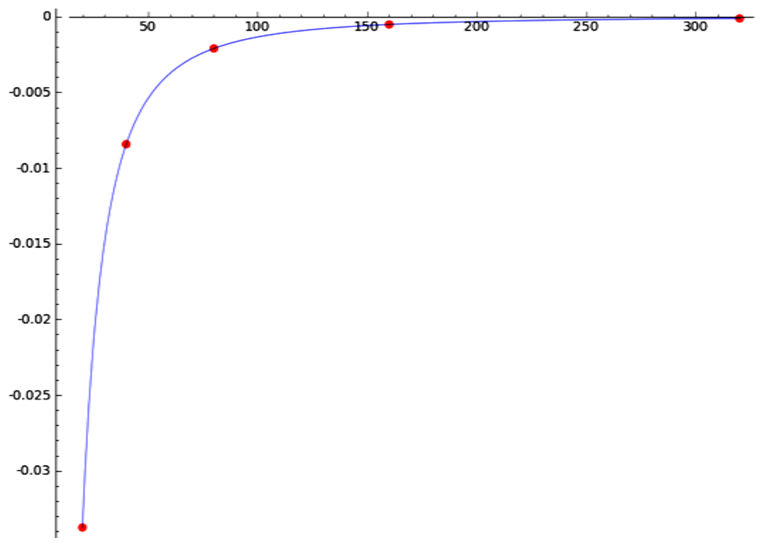

Figure 2. The error distribution for $\int_{-1}^{2}(4x^3+2x)\mathrm{d}x$ calculated with Midpoint Rule. The red dots are the errors and the blue curve is $y=-13.5(1/N)^2$.

Table 1. Numerical results for $\int_{-1}^{2}(4x^3+2x)\mathrm{d}x$ calculated with the Midpoint Rule.

N	$\tilde{I}$	$R(f,N)$	$R(f,N)/R(f,2N)$
20	17.96625	−0.03375000000000128	
40	17.99156	−0.00843750000000298	3.99999999999874
80	17.99789	−0.00210937500000341	3.99999999999495
160	17.99947	−0.00052734375000085	4.00000000000000
320	17.99986	−0.00013183593750198	3.99999999994610

The nodes x_k are defined in subsection 2.1 as given in **Figure 1**.

The error term is (see [5] for details)

$$R(f,N)=-\frac{(b-a)^3}{12N^2}f''(\xi)=cf''(\xi)h^2,\qquad \xi\in[a,b]. \tag{22}$$

where $c=-\frac{b-a}{12}$.

As with the Midpoint Rule, we obtain

$$E=\frac{R(f,N)}{R(f,2N)}\approx 4. \tag{23}$$

3.2. Numerical Experiments

We construct a SAGE function TPloop(a,b,n) for the composite Trapezoidal Rule:

```
sage: def TPloop(a,b,n):
...       h=(b-a)/n # the step
...       AI = f(a)
...       for i in [1..(n-1)]:
...           AI = AI + 2*f(a+i*h)
...       AI = AI +f(b)
...       AI = float(AI*h/2)
...       return AI
```

The variables a and b are the lower and upper limits of the integral, and the variable n is the number of subintervals.

The results obtained with function TRloop for the integral (18) are shown in **Table 2**. The last column shows the ratio of the errors $\frac{R(f,N)}{R(f,2N)}$. Our numerical calculations confirm that if we double the number of subintervals, the error term is decreased four times. The error distribution for the same calculation is shown graphically in **Figure 3**. The graph of the function $y=\frac{27}{N^2}$ is given in the same figure. The error points and the graph of the quadratic function are undistinguishable. This experiment confirms the second order of approximation of the numerical realization of the Trapezoidal Rule.

4. Simpson's Rule

Suppose we want to use three nodes $x_0=a$, $x_1=\frac{a+b}{2}$, and $x_2=b$. For this case the quadrature Formula (5) has three free (unknown) coefficients

$$\int_a^b f(x)\mathrm{d}x \approx A_0 f(a) + A_1 f\left(\frac{a+b}{2}\right) + A_2 f(b),$$

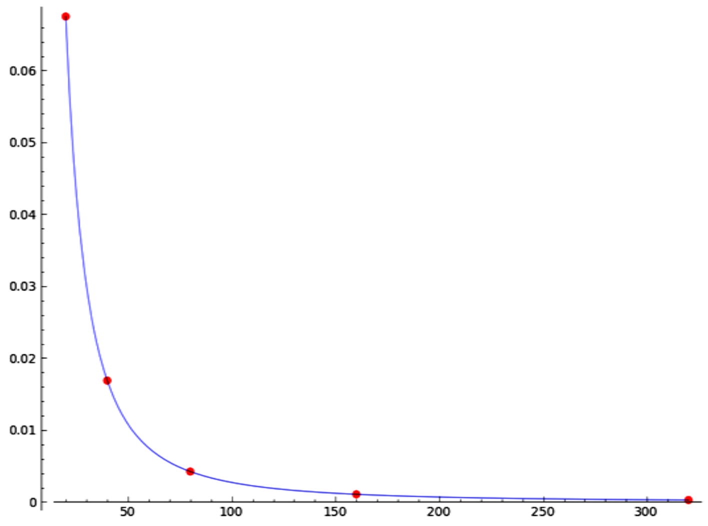

Figure 3. The error distribution for $\int_{-1}^{2}(4x^3+2x)\mathrm{d}x$ calculated with the Trapezoidal Rule. The red dots are the errors and the blue curve is $y=27(1/N)^2$.

Table 2. Numerical results for $\int_{-1}^{2}(4x^3+2x)\mathrm{d}x$ calculated with the Trapezoidal Rule.

N	$\tilde{I}$	$R(f,N)$	$R(f,N)/R(f,2N)$
20	18.0675	0.0675	
40	18.016875	0.016875	4.00000000000021
80	18.00421875	0.00421875	4.00000000000000
160	18.001054687499998	0.001054687499998	4.00000000000674
320	18.000263671874997	0.000263671874997	4.00000000004042

and we can make the formula exact for second order polynomials. In other words, the formula must be exact for all polynomials of zero degree and particularly for $P(x)=1$. So,

$$A_0 + A_1 + A_2 = \int_a^b 1\mathrm{d}x = b-a,$$

The formula must be exact for all first order polynomials and particularly for $P(x)=x$. Therefore,

$$aA_0 + \frac{a+b}{2}A_1 + bA_2 = \int_a^b x\mathrm{d}x = \frac{b^2-a^2}{2},$$

The formula must be exact for all second order polynomials and particularly for $P(x)=x^2$. Therefore,

$$a^2A_0 + \left(\frac{a+b}{2}\right)^2 A_1 + b^2A_2 = \int_a^b x^2\mathrm{d}x = \frac{b^3-a^3}{3}.$$

The last three equations form the system (7) for this case. The solution to the system is $A_0 = A_2 = \frac{b-a}{6}$, $A_3 = 4\frac{b-a}{6}$, and the *Simpson's Rule* is given by

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]+R(f). \tag{24}$$

The error term, under the condition that the fourth derivative $f^{(4)}(x)$ of the function $f(x)$ exists and is continuous in the interval $[a,b]$, is given by (see [5])

$$R(f) = -\left(\frac{b-a}{2}\right)^5 \frac{f^{(4)}(\xi)}{90}, \qquad \xi\in[a,b]. \tag{25}$$

4.1. Composite Simpson's Formula

One way to construct the composite Simpson's formula is to divide the interval $[a,b]$ into an even number of subintervals, say $N=2m$. Then the composite formula is

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x \approx \frac{h}{3}\{&f(a)+f(b)+2\left[f(x_2)+f(x_4)+\cdots+f(x_{2m-2})\right]\\ &+4\left[f(x_1)+f(x_3)+\cdots+f(x_{2m-1})\right]\},\end{aligned} \tag{26}$$

where $h = \frac{b-a}{N} = \frac{b-a}{2m}$. The error term in this case is

$$R(f,N) = -\left(\frac{b-a}{2}\right)^5 \frac{f^{(4)}(\xi)}{90m^4} = cf^{(4)}(\xi)h^4, \qquad \xi\in(a,b), \tag{27}$$

where $c = \frac{b-a}{180}$.

As with the two previous rules, we obtain

$$E = \frac{R(f,N)}{R(f,2N)} = 16. \tag{28}$$

So, if the number of subintervals is multiplied by 2, the numerical error decreases by a factor of $1/16$.

4.2. Numerical Experiments

We prepared a SAGE function Simpsons for our numerical experiments with Simpson's Rule. The code is given here:

```
sage: def Simpsons(a,b,n):
...       h=(b-a)/n # the step
...       AI = f(a)
...       for i in [1..n/2]:
...           k=2*(i-1)+1
...           AI = AI + 4*f(a+k*h)+2*f(a+(k+1)*h)
...
...       AI = AI - f(b)
...       AI = float(AI*h/3)
...       return AI
```

The variables a and b are the lower and upper limits of the integral, and the variable n is the number of subintervals.

The results obtained with function Simpsons for the integral

$$I = \int_{-1}^{2} \sin x dx = 0.9564491424152821 \tag{29}$$

are shown in **Table 3**. The last column shows the ratio of the errors $\frac{R(f,N)}{R(f,2N)}$. Our numerical calculations confirm that if we double the number of subintervals, the error term decreases sixteen times. The error distribution for the same calculation is given in **Figure 4**. The red dots are the numerical error and the blue line is the graph of the function $y = \frac{0.43}{N^4}$. Because the error approaches zero very rapidly, we use a logarithmic scale on the y-axis. The numerical experiments confirm that the numerical error's rate of approach to zero as N approaches infinity is the same as the rate of approach to zero of the function $y = \frac{0.43}{N^4}$.

5. Error Estimation from Numerical Results

How do we determine the number of subintervals such that the computed result meets some prescribed accuracy?

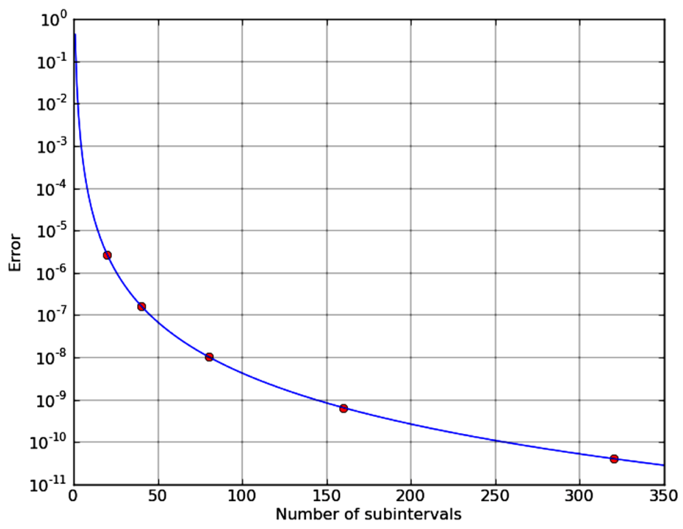

Figure 4. The error distribution for $\int_{-1}^{2} \sin x dx$ calculated with composite Simpson's Rule. The red dots are the errors and the blue curve is $y = 0.43 h^4$.

Table 3. Numerical results for $\int_{-1}^{2} \sin \mathrm{d}x$ calculated with Simpson's Rule.

N	$\tilde{I}$	$R(f,N)$	$R(f,N)/R(f,2N)$
20	0.9564518396509495	2.6972356673704567e−06	
40	0.9564493106537587	1.6823847659441782e−07	16.03221642260134
80	0.9564491529249056	1.0509623504795229e−08	16.00804029922248
160	0.9564491430720508	6.567686394731709e−10	16.00201786922341
320	0.9564491424563286	4.104649953262651e−11	16.00060046414255

According to the formulas for the error (14), (22), and (27), the error depends on a derivative of the integrand. If we know the corresponding derivative of the integrand, we can calculate the upper bound of the error. Usually, in practice we do not know the relevant derivative of the integrand and we cannot bound the error this way.

In practice, we frequently use the rule known as *Runge's principle*. The formulas for numerical integration (12), (21), and (26) are of the form

$$I = \int_a^b f(x)\,\mathrm{d}x = I_N + R(f,N), \tag{30}$$

where I_N is the approximated value of the integral calculated when the interval $[a,b]$ is divided into N subintervals, and

$$R(f,N) = \frac{cf^{(m)}(\xi)}{N^p}, \tag{31}$$

is the error term for the corresponding formula, c is a constant, m, p are integers, and $\xi \in [a,b]$.

Suppose we perform two calculations with different numbers of subintervals, N and $2N$:

$$I = I_N + \frac{cf^{(m)}(\xi')}{N^p} \quad \text{and} \quad I = I_{2N} + \frac{cf^{(m)}(\xi'')}{2^p N^p}. \tag{32}$$

Under the assumption $f^{(m)}(\xi') \approx f^{(m)}(\xi'')$ (see [5] for details), we can eliminate I from the above two equations, to obtain

$$R(f,2N) \approx r(2N) = \frac{I_{2N} - I_N}{2^p - 1}. \tag{33}$$

This approximation of the error term is known as *Runge's principle*. In addition, instead of I_{2N}, we can use

$$\tilde{I} = I_{2N} + r(2N) = I_{2N} + \frac{I_{2N} - I_N}{2^p - 1}, \tag{34}$$

as a better approximation of the integral. This technique is known as *Richardson's extrapolation*; it is used in many types of numerical calculations, especially when we are solving differential equations numerically.

In practice, usually the exact value of the integral I is unknown. So, we cannot use the Formulas (17), (23), and (28) to confirm the power of h in the error term. Substituting $R(f,N)$ with $r(N)$ from the Runge principle, we obtain

$$2^p = \frac{R(f,2N)}{R(f,4N)} \approx \frac{r(2N)}{r(4N)} = \frac{I_{2N} - I_N}{I_{4N} - I_{2N}}. \tag{35}$$

Next we will consider the Runge principle, Richardson's extrapolation, and the power of h estimation in the error term to the Midpoint, Trapezoidal, and Simpson's rules.

5.1. Midpoint Rule

A simple SAGE implementation of the Runge principle for estimating the error during the numerical calculations is given here:

```
 sage: def MPerror(a,b,eps):
 ...        Inew =(b-a)*f((a+b)/2 )
 ...        Err=10
 ...        n=1
 ...        while Err>eps :
 ...            Iold=Inew
 ...            n=n*2
 ...            h=(b-a)/n
 ...            Inew = 0
 ...            for i in [0..(n-1)]:
...               Inew=Inew+f(a+(i+0.5)*h)
...            Inew=float(Inew*h)
...            Err=abs(Inew-Iold)/3
...            Iextr=Inew+(Inew-Iold)/3
...        return (Inew, Iextr, n)
```

The variables a and b in the definition of the function MPerror contain the lower and upper limits of the integral. The variable eps is the satisfactory error. The function returns: the first variable, Inew, is the approximation of the integral with the Midpoint Rule; the second variable, Iextr, is the value of Richardson's extrapolation for the integral using Iold and Inew; the third variable n gives the number of subintervals for the final approximation.

To verify the theoretical results for the Midpoint method, we calculate the integral

$$\int_1^4 \sin x \mathrm{d}x \tag{36}$$

with a different number of subintervals, and approximate the value of the integral with Richardson's extrapolation. The results are given in Table 4. Our numerical experiments confirm the second order of approximation $\left(4=2^2\right)$ of the Midpoint Rule, and show that Richardson's approximation has a fourth order of approximation $\left(16=2^4\right)$ of the integral.

5.2. Trapezoidal Rule

When the number of subintervals N increases, the error in the composite Trapezoidal Rule reduces. On the other hand, when N increases, the number of the function values to be calculated also increases. We need to find a balance between the accuracy requirements and the cost (in operations or computing time) of the algorithm. This balance can be reached with the following algorithm:

We calculate the approximation of the integral I_0 by applying the Trapezoidal Rule over the whole interval $[a,b]$. Then we calculate the approximation I_1 by dividing the interval $[a,b]$ into two subintervals, and in each we apply the Trapezoidal Rule. In this partition the new node is only one in the middle of this interval. The other two nodes are involved in the calculation of I_0 and we do not need to calculate the value of the function on those nodes again. We calculate I_2 by introducing four subintervals. The new nodes are only two. So, we find $I_3, I_4, \cdots, I_k, \cdots$, such that the number of subintervals needed to calculate I_k is 2^k, but the number of new nodes of the mesh on which we calculate I_k is 2^{k-1}. The remaining $2^{k-1}+1$ nodes are involved in the calculation of I_{k-1}, and the values of the integrand are already calculated. The process continues until the following condition holds:

Table 4. Numerical error for the Midpoint Rule, Richardson's extrapolation, and the rates of convergence.

N	Midpoint error	rate	Richardson's extrap.	rate
8	0.007024577280901		−0.00011693442490234	
16	0.0017507392430902	4.01234	−7.20676951360e−06	16.22563
32	0.0004373481675656	4.00307	−4.4885760908108e−07	16.05580
64	0.00010931601997121	4.00076	−2.80292269394e−08	16.01391
128	2.7327691408896e-05	4.00019	−1.7514452110845e−09	16.00348

$$\frac{|I_k - I_{k-1}|}{3} < \varepsilon.$$

This algorithm is an example of the so-called *adaptive algorithms* the number of required subintervals depends on the integrand, but the step is uniform throughout the interval $[a,b]$. Sometimes we need to use a mesh with different steps depending on the behavior of the function under the integral.

A simple SAGE implementation of the Runge principle for estimating the error during the numerical calculations with Trapezoidal Rule is given here:

```
sage: def TRerror(a,b,eps):
...       h = b-a
...       Inew =h*(f(a)+f(b))/2
...       Err=10
...       n=1
...       while Err>eps :
...           Iold=Inew
...           Inew = 0
...           for i in [0..(n-1)]:
...                Inew=Inew+f(a+i*h+h/2)
...           Inew=float(Inew*h+Iold)/2
...           n=n*2
...           h=h/2
...           Err=abs(Inew-Iold)/3
...           Iextr=Inew+(Inew-Iold)/3
...           Iold=Inew
...       return (Inew, Iextr, n)
```

The variables a and b in the definition of the function TRerror contain the lower and upper limits of the integral. The variable eps is the satisfactory error. The function returns: the first variable, Inew, is the approximation of the integral with Trapezoidal Rule; the second variable, Iextr, is the value of Richardson's extrapolation for the integral using Iold and Inew; the third variable n gives the number of subintervals for the final approximation.

To verify the theoretical results for the Trapezoidal method, we calculate the integral (36) with a different number of subintervals and approximate the value of the integral with Richardson's extrapolation. The results are given in **Table 5**. Our numerical experiments confirm the second order of approximation $(4 = 2^2)$ of the Trapezoidal Rule and show that Richardson's approximation has forth order of approximation $(16 = 2^4)$ of the integral.

The fact that Richardson's extrapolation applied to the Trapezoidal Rule gives fourth order of approximation has a simple explanation. Since,

$$\begin{aligned} I_k + \frac{I_k - I_{k-1}}{3} = \frac{4I_k - I_{k-1}}{3} &= \frac{1}{3}\left\{4h\left[\frac{f(a)+f(b)}{2} + f(x_1) + f(x_2) + \cdots + f\left(x_{2^k-1}\right)\right]\right. \\ &\left. - 2h\left[\frac{f(a)+f(b)}{2} + f(x_2) + f(x_4) + \cdots + f\left(x_{2^k-2}\right)\right]\right\} \\ &= \frac{h}{3}\left\{f(a) + f(b) + 2\left[f(x_2) + f(x_4) + \cdots + f(x_{2m-2})\right]\right. \\ &\left. +4\left[f(x_1) + f(x_3) + \cdots + f(x_{2m-1})\right]\right\}, \end{aligned} \tag{37}$$

Table 5. Numerical error for the Trapezoidal Rule, Richardson's extrapolation, and the rates of convergence.

N	Trapezoidal error	Rate	Richardson's extrap.	Rate
8	−0.0140244567175		0.0001333996544001792	
16	−0.0034999397183	4.00705	8.232614749248413e−06	16.20380
32	−0.00087460023761	4.00175	5.129226172684298e−07	16.05040
64	−0.000218626035026	4.00043	3.203250420469317e−08	16.01256
128	−5.4655007526e−05	4.00010	2.0016397428435084e−09	16.00313

Richardson's extrapolation for the Trapezoidal Rule is nothing else but Simpson's formula. Our numerical experiments show that the error term for Richardson's extrapolation is the same $\left(16=2^4\right)$ as the error term for Simpson's formula.

5.3. Simpson's Rule

To verify the theoretical results for Simpson's method, we calculate the integral (36) with a different number of subintervals, and approximate the value of the integral with Richardson's extrapolation. The results are given in **Table 6**. Our numerical experiments confirm the fourth order of approximation $\left(16=2^4\right)$ of Simpson's rule, and show that Richardson's approximation has a sixth order of approximation $\left(64=2^6\right)$ of the integral. The SAGE code is given here:

```
sage: def SMPSerror(a,b,eps):
...       EI=float(integrate(f(x), (x,a,b)))
...       Inew =(b-a)*float(f(a)+4*f((a+b)/2 )+f(b))/6
...       Err=10
...       n=1
...       while Err>eps :
...           Iold=Inew
...           n=n*2
...           h=(b-a)/n
...           Inew = 0
...           for i in [0..(n-1)]:
...               Inew=Inew+f(a+i*h)+4*f(a+i*h+h/2)+f(a+i*h+h)
...           Inew=float(Inew*h)/6
...           Err=abs(Inew-Iold)/15
...           Iextr=Inew+(Inew-Iold)/15
...           Iold=Inew
...       return (Inew, Iextr, n)
```

6. Adaptive Procedures

According to [5]: "An adaptive quadrature routine is a numerical quadrature algorithm which uses one or two basic quadrature rules and which automatically determines the subinterval size so that the computed result meets some prescribed accuracy requirement.'' For our adaptive function we chose the Midpoint Rule, because it involves just one node to approximate the integral in the given subinterval. The SAGE function needs to be changed slightly if one wishes to use a different quadrature formula. The SAGE code for the function MPrecursion(a,b,eps) is given here:

```
sage: def MPrecursion(a,b,eps):
...       I11 = (b-a)*f(a+ (b-a)/4 )/2
...       I12 = (b-a)*( f(a+(b-a)/8)+f(a+3*(b-a)/8) )/4
...       err=(I12-I11)/3
...       if abs(err) < eps/2 :
...           I1 = I12
...       else:
...           I1 = MPrecursion(a, (a+b)/2, eps/2)
...       I21 = (b-a)*f(a+3*(b-a)/4)/2
...       I22 = (b-a)*( f(a+5*(b-a)/8)+f(a+7*(b-a)/8) )/4
...       err=(I22-I21)/3
...       if abs(err) < eps/2 :
...           I2 = I22
...       else:
...           I2 = MPrecursion((a+b)/2,b, eps/2)
...       I=I1+I2
...       return I
```

The parameters of the function MPrecursion are a and b-the lower and upper limits of the integral, and eps the accuracy. The function divides the interval into two subintervals, and, in each subinterval, approximates the integral twice using the Midpoint Rule for the full half interval and using the composite Midpoint Rule, which divides the half interval into two subintervals. If the error, according to Runge's principle, is less than eps, the value calculated with the composite rule is accepted as an approximation of the integral for this subinterval. If

Table 6. Numerical error for Simpson's rule, the Richardson's extrapolation, and the rates of convergence.

N	Simpson's error	Rate	Richardson's extrap.	Rate
4	0.0001333996544004		−7.579037909266617e−06	
8	8.23261474947e−06	16.2038	−1.1185456050277764e−07	67.7579
16	5.12922617490e−07	16.0504	−1.7235246563274131e−09	64.8987
32	3.20325042046e−08	16.0125	−2.6836755040449134e−11	64.2225
64	2.001638854665e−09	16.0031	−4.1877612488860905e−13	64.0837

the error is greater than the prescribed accuracy, the function calls itself with a half-length interval and halved accuracy requirements. Therefore, we have one recursive algorithm which uses different mesh sizes for different parts of the interval. If the integrand is smooth and slowly varying, a relatively small number of subintervals is used. In the parts of the interval where the integrand is changing rapidly, the mesh is relatively fine.

An example of the behavior of the function MPrecursion is given in **Figure 5**. The red dots on the x-axis are the nodes used for calculation of

$$\int_{0.04}^{2} \sin\left(\frac{1}{x}\right) \mathrm{d}x, \tag{38}$$

and the blue line is the graph of the integrand.

7. Can We Always Trust the Runge Principle?

As we have shown in the previous section, the Runge principle for estimating the error during the numerical calculations works well under the condition that the respective derivative of the integrand exists and this derivative is a continuous function. If the derivative in the error term does not exist, the Runge principle fails, and Richardson's extrapolation cannot be applied. Consider the following examples.

7.1. Improper Integrals

The formal definition of *improper integral* is:

If $f(x)\to\infty$ when $x\to a$ (or $f(x)\to\infty$ when $x\to b$), then

$$\int_a^b f(x)\mathrm{d}x = \lim_{c\to a}\int_c^b f(x)\mathrm{d}x, \quad \left(\text{or } \int_a^b f(x)\mathrm{d}x = \lim_{c\to b}\int_a^c f(x)\mathrm{d}x\right). \tag{39}$$

Examples:

$$\int_0^1 \frac{\mathrm{d}x}{\sqrt{x}} = 2, \quad \text{and} \quad \int_0^1 \frac{\mathrm{d}x}{\sqrt{1-x}} = 2. \tag{40}$$

Calculating improper integrals numerically is challenging. Usually, we need some *a priori* information about the integrand, because the techniques we apply depend on the properties of this function. Two main problems arise:

1) the integrand is not defined at one (or both) limits of the integral;

2) the integrand goes to infinity at one (or both) limits of the integral;

Because of 1) Trapezoidal and Simpson's rules cannot be used. The Midpoint Rule can be used, since we do not need the value of the function at the endpoints a and b.

The results from calculations of the first improper integral from (40) with the Midpoint Rule are given in **Table 7**. The ratio of the errors in this case is about 1.414. Compared to the ratio of 4 for the "good'' functions, we see that the convergence is very slow. We will have a similar convergence for a "good'' integrand when we approximate the integral with a formula with error term $R(f,N)=c\sqrt{h}=c\sqrt{1/N}$.

7.2. Singularity in the Derivative

The slow convergence of the improper integral is not a surprise, since the integrand goes to infinity at one of the endpoints of the interval. Consider the integral

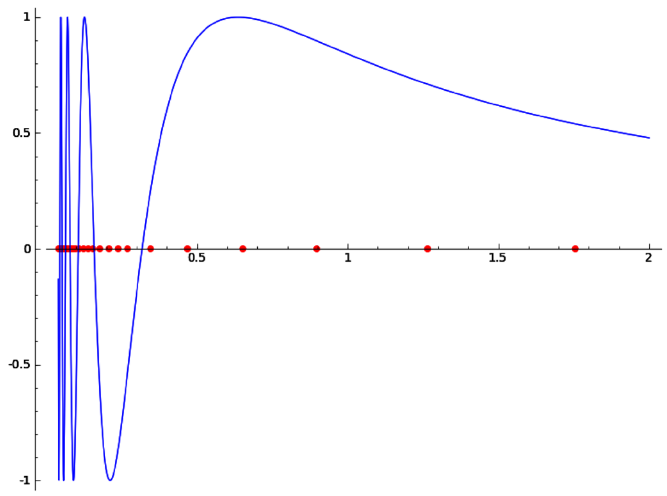

Figure 5. The mesh used from the adaptive algorithm to approximate.

Table 7. Numerical results for $\int_0^1 1/\sqrt{x}\,\mathrm{d}x$ calculated with the Midpoint Rule.

N	$\tilde{I}$	$R(f,N)$	$R(f,N)/R(f,2N)$
20	1.8647926204877276	−0.13520737951227235	
40	1.9043701466055494	−0.09562985339445063	1.4138616207490537
80	1.9323735308432972	−0.09562985339445063	1.4140891072230746
160	1.9521793771296476	−0.04782062287035238	1.4141695590215648
320	1.966185341295599	−0.033814658704401035	1.4141980047288913

$$\int_0^1 \sqrt{x}\,\mathrm{d}x = \frac{2}{3}. \tag{41}$$

The function $y=\sqrt{x}$ is a continuous and bounded function, but the second derivative of this function goes to infinity when $x \to 0$. So, we can expect problems with practical convergence of the numerical calculation of the integral. The results from the numerical calculations of (41) with the Midpoint Rule are given in **Table 8**. The ratio of the errors in this case is about 2.8. Comparing this ratio to 4, for the “good’’ functions, we see that the convergence is slow. We will have a similar convergence for a “good’’ integrand when we approximate the integral with the formula with error term $R(f,N)=c\sqrt{h^3}=c\sqrt{1/N^3}$.

8. Behavior of the Error in Practice

We have to emphasize that in practical calculations the error in the final result depends on the accumulation of rounding errors. With the increasing number of subintervals the theoretical error decreases, but the number of arithmetic operations, each of which contains an error in the order of machine epsilon, increases. Therefore it is reasonable to use as many subintervals as we need to get the theoretical error term to exceed the machine epsilon. The total error starts to increase when the number of subintervals increases beyond a certain unknown number dependent on the integrand, the programming language, and the computer.

To illustrate this effect with our SAGE codes we use the class RealField with 32 bits. The effect when the Sage 64-bit floating-point system is used is similar but the computing time increases. The numerical results for three integrals,

$$\int_0^1 \sin(5x)\,\mathrm{d}x,\quad \int_0^1 20x\exp\left(5x^2\right)\mathrm{d}x,\quad \int_0^1\left(x^{1}0+x^{7}\right)\mathrm{d}x, \tag{42}$$

are shown in **Figure 6**. When the number of subintervals is relatively small, the error decreases as the number of subintervals increases. When the number of subintervals becomes greater than a certain number, the error starts to increase. The number after which the error starts to increase depends on the integrand.

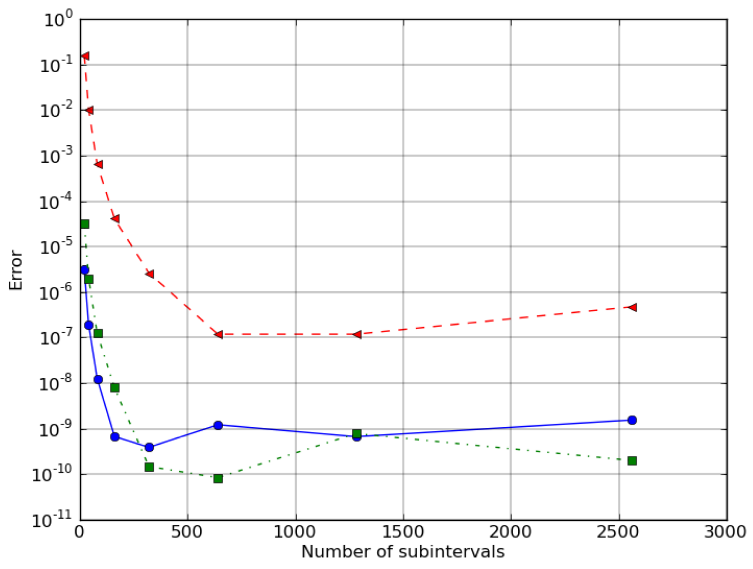

Figure 6. The error behavior for $\int_0^1 \sin(5x)\mathrm{d}x$ the blue solid line, $\int_0^1 20x\exp(5x^2)\mathrm{d}x$ the red dashed line, and $\int_0^1 (x^{10}+x^7)\mathrm{d}x$ the green dash-dot line.

Table 8. Numerical results for $\int_0^1 \sqrt{x}\mathrm{d}x$ calculated with Midpoint Rule.

N	$\tilde{I}$	$R(f,N)$	$R(f,N)/R(f,2N)$
20	0.6672953399204201	0.0006286732537534867	
40	0.6668943288044112	0.00022766213774461086	2.761430864093555
80	0.6667485056870116	8.183902034497592e−05	2.7818287265041413
160	0.6666959382144143	2.927154774767793e−05	2.795855588178391
320	0.6666770999933831	1.0433326716463576e−05	2.8055814356400846

Acknowledgements

This work was supported by NSF grant HRD-0928797.

References

[1] Marinov, T.T. and Marinova, R.S. (2010) Coefficient Identification in Euler-Bernoulli Equation from Over-Posed Data. *Journal of Computational and Applied Mathematics*, **235**, 450-459. http://dx.doi.org/10.4153/CJM-1969-153-1

[2] Marinov, T.T. and Vatsala, A. (2008) Inverse Problem for Coefficient Identification in Euler-Bernoulli Equation. *Computers and Mathematics with Applications*, **56**, 400-410. http://dx.doi.org/10.1112/blms/25.6.527

[3] Ackleh, A.S., Kearfott, R.B. and Allen, E.J. (2009) Classical and Modern Numerical Analysis: Theory, Methods and Practice, Chapman & Hall/CRC Numerical Analysis and Scientific Computing. CRC Press, Boca Raton. http://dx.doi.org/10.1142/S0218196791000146

[4] Press, W.H. and Teukolsky, S.A. and Vetterling, W.T. and Flannery, B.P. (2007) Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, New York. http://dx.doi.org/10.1142/S0218196799000199

[5] Forsythe, G.E. and Malcolm, M.A. and Moler, C.B. (1977) Computer Methods for Mathematical Computations. Prentice Hall Professional Technical Reference, Upper Saddle River. http://dx.doi.org/10.1051/ita/2012010

[6] Marinov T.T. and Deng, K. (2011) Characteristic Line Based Schemes for Solving a Quasilinear Hierarchical Size-Structured Model. *Journal of Scientific Computing*, **46**, 452-469. http://dx.doi.org/10.1016/j.tcs.2009.12.005