Several kinds of variable step size methods for computer numerical integration

Hu Jia

Nanchang Normal University, Department of mathematics and computer science

Nanchang China 330000

Keywords: variable step trapezoidal quadrature method, adaptive trapezoidal quadrature method, variable step size Simpson quadrature method, Romberg integration

Abstract: This paper introduced several numerical solution methods including variable step trapezoidal quadrature method, adaptive trapezoidal quadrature method, variable step size Simpson quadrature method and Romberg integration.

1、Introduction

Accuracy is an important problem in numerical integration. In order to improve the accuracy, it’s usually required to minimize the length of each subdivision interval in the integral formula, namely reduce the step size. If step size is too small, the computational workload will increase and because of the increase in the number of items, the accumulation of the error will increase, so in the actual calculations, we usually use Integral method of variable step size.

2 Variable step trapezoidal quadrature method

The base of variable step trapezoidal integration is trapezoidal formula. In the calculation, it needs to subdivide the interval according to the accuracy requirements gradually. In the process of subdivision, in order to make full use of the value of the function has been obtained, it always needs to bisect the small cell. Taking variable step trapezoidal quadrature method to calculate definite integral $y=\int_b^a f(x)dx$ and trapezoidal quadrature formula takes two endpoints in the interval [ai,bi] to construct a linear function to approximate f(x), so

$\int_{ai}^{bi} f(x)dx \approx \int_{ai}^{bi} p1(x)dx = \frac{bi-ai}{2}[f(ai)+f(bi)]$. First, it needs to divide the interval [a,b] into n copies and trapezoidal quadrature formula is used in each interval to carry out the integration and then sum up all the results. The integration result is shown as follows.

$yn=\frac{h}{2}[f(a)+f(b)]+\sum_{k=1}^{n-1} h\times f(a+k\times h)$

Where h is the width of each interval namely the step size of the integration. However, the initial selection of linear functions in n intervals sometimes cannot fully approximate f(x). Variable step trapezoidal quadrature method requires to divide every small interval into two parts, so the integration result changes into the following one.

$y2n=\frac{h}{4}[f(a)+f(b)]+\sum_{k=1}^{2n-1}\frac{h}{2}\times f\left(a+k\times\frac{h}{2}\right)=\frac{yn}{2}+\sum_{k=0}^{n-1}\frac{h}{2}\times f(a+k\times h+\frac{h}{2})$

After the continuous division of the interval, until the difference of integral value after two times integration is less than the required accuracy, namely $|y2n-yn|<\varepsilon$. At this time, y2n is the final integral result. In order to prevent the endless division, it is generally supposed to limit the minimum step size. When the step size is less than the specific value h0, even if the result does not meet the accuracy requirements, the division will not continue.

Algorithm thought

(1) Dividing the n copies, calculating step h and integral value yn according to the initial interval

$h=\frac{b-a}{n}, yn=\frac{h}{2}[f(a)+f(b)]+\sum_{k=1}^{n-1} h\times f(a+k\times h)$

(2)Getting the integral value after the step size is halved y2n=$\frac{yn}{2}+\sum_{k=0}^{n-1}\frac{h}{2}\times f(a+k\times h+\frac{h}{2})$

(3) Updating the absolute value difference of the two integration, the number of the division, the integral value and the step size

d=|y2n-yn|， yn=y2n,n=2n,h=$\frac{h}{2}$

(4) If d>ε and h>h0, then turn to (2) to continue division. Otherwise, return the integral value yn, exit function

```
          /*==================================
          //Function name：tzi1
            //Function description：Variable step trapezoidal quadrature method
          //Input parameter：a Integral lower limit，b Integral upper limit
                       n0 Initial partition number
                       eps accuracy requirement
                       h0 minimum step size
                       f pointer to the integrand
          return value：integral approximation ==================================*/
#include "math.h"
double tzil(a,b,n0,eps,h0,f)
double a,b,eps,h0;
int n0;
double (*f)()
{
  int n,k;
  double z,z2,h,d,x,t;
  n=n0;
  h=(b-a)/n;
  z=(*f)(a)+(*f)(b)/2.0;
  for(k=1;k<n;k++)
{
  x=a+k*h;
  z=z+(*f)(x);
}
z=z*h;
do
{ t=0.0;
    for(k=0;k<n;k++)
    {x=a+(k+0.5)*h;
      t=t+(*f)(x);
      }
z2=(z+h*t)/2.0;
d=fabs(z2-z);
z=z2;
h=h/2.0;
n=2*n;
}
while((d>eps)&&(h>h0));
return(z);
}
```

3Adaptive trapezoid quadrature method

Adaptive trapezoidal quadrature method is actually based on trapezoidal quadrature method. It has been improved based on the variable step trapezoidal quadrature method. In the variable step trapezoidal quadrature method, when the difference of two integral values cannot meet the accuracy requirements, it divides all the intervals into two parts. But in fact, not all the intervals need to re-integration. Adaptive trapezoid quadrature method is based on such idea, its process is as follows.

Supposed that the precision requirement of an integral interval called zero level subinterval. First, the initial integral approximation $z^{(0)}$ of an integral interval is obtained according to the trapezoidal formula. Then the integral interval is divided into two equal sub ranges called 1 level subinterval. Then the initial integral approximation $z_0^{(1)}$ and $z_1^{(1)}$ of each 1 level subinterval are obtained taking the trapezoidal formula. If inequality$| z^{(0)}-(z_0^{(1)}+ z_1^{(1)})|< \varepsilon/2$ holds, it is assumed that the integral interval has met the accuracy requirements and the integral approximation of the function on the interval is $z_0^{(1)}+ z_1^{(1)}$。Otherwise, two 2 level subintervals are divided from each 1 level subinterval and whether they meet the precision accuracy is checked respectively namely the inequality $| z^{(1)} -(z_0^{(2)} + z_1^{(2)})|< \varepsilon/4$.

For those which meets the precision requirements, it stops division but for those which cannot meet the precision requirement, it continues dividing until all the intervals meet the accuracy demand.

In order to prevent the endless division, generally it should limit the minimum step size. When the step size is less than the specific value of h0, even if it does not meet the accuracy requirements, the division will not continue.

Algorithm thought

(1) Calculating the step h and the integral approximation zi of each integral interval according to the initial value n

h=b-a/n

for i=0,1,…n-1

do $z_i=\frac{h}{2}$ [f（a+i×h）+f（a+i×h+1）]

(2) Using zi as the initial value to calculate the integral approximation z_i’ on each interval

z_i’=subtz(x_i,x_{i+1},y_i,y_{i+1},h,z_i,eps,h0,f)

(3) The sum of the integral values of all intervals is the total integral value. The subtz process is as follows

① Calculating the interval approximation of two subintervals

$h=\frac{h}{2}$, $z_0=\frac{h}{2}$（y_0+f（x_0+h）), $z_1=\frac{h}{2}$（f（x_0+h）+ y_1）

② If it already meets the precision requirements or it is not allowed to divide, it returns the z0+z1. Otherwise, it calculates the integral approximation z0’ and z1’ of two subintervals by taking the z0 and z1 as the initial value to the recursive function subtz and returns the value z0’+z1’.

```
/*==================================
//Function name：tzi2
//Function description：Adaptive trapezoidal integration
//Input parameter：a integral lower limit，b integral upper limmit
                   n0 initial division number
                   eps accuracy requirement
                   h0 minimum step size
                   f pointer to the integrand
     return value：integral approximation
  ==================================*/
     #include "math.h"
     double tzil2(a,b,n0,eps,h0,f)
     double a,b,eps,h0;
     int   n0;
```

```
        double (*f)();
        {
          int n,k;
          double z,h,t;
          double x0,x1,y0,y1;
          double subtz();
          n=n0;
          h=(b-a)/n;
          z=0.0;
          for(k=0;k<n;k++)
          {
            x0=a+k*h;
            x1=a+(k+1)*h;
            y0=(*f)(x0);
            y1=(*f)(x1);
            t=h*(y0+y1)/2.0;
            t=subtz(x0,x1,y0,y1,h,t,eps/n,h0,f);
            z=z+t;}
            return (z);
        }
        static double subtz(x0,x1,y0,y1,h,t,eps,h0,f)
        double x0,x1,y0,y1,h,t;
        double eps,h0;
        double (*f)()
        {double x,y,t1,t2,d,z;
          x=x0+h/2.0;
          y=(*f)(x);
          t1=h*(y0+y)/4.0;
          t2=h*(y+y1)/4.0;
          d=fabs(t-(t1+t2));
          if((d<eps)||(h>h0))
          z=t1+t2;
        else
        { t1=subtz(x0,x,y0,y,h/2.0,t1,eps/2.0,h0,f);
            t2=subtz(x,x1,y,y1,h/2.0,t2,eps/2.0,h0,f);
            z=t1+t2;
        }
        return(z);
    }
```

4 Variable step Simpson quadrature method

The basic idea of variable step Simpson quadrature method is same as variable step trapezoidal quadrature method, but the basic formulas of variable step Simpson quadrature method is not Simpson formula, but the trapezoidal formula.

Simpson quadrature formula selects three equidistant nodes a_i，$(a_i+b_i)/2$，bi in the small cell [ai, bi] to construct quadratic functions to approximate the integrand, so $\int_{ai}^{bi} f(x)dx \approx \int_{ai}^{bi} P_2(x)dx = \frac{bi-ai}{6}[f(ai)+4f\left(\frac{ai+bi}{2}\right)+f(bi)]$. In the variable step trapezoidal quadrature method, first it needs to divide the integral interval into n small intervals, then the quadrature is shown as follow:

$Z_n = \frac{h}{2}[f(a)+f(b)] + \sum_{k=1}^{n-1} h \times f(a+k\times h)$

Where h is the width of each small interval namely the integral step. Dividing each small interval into two parts, then the quadrature changes into:

$Z_{2n}=\frac{Zn}{2}+\sum_{k=0}^{n-1}\frac{h}{2}\times f(a+k\times h+\frac{h}{2})$

From the two integral results above, it can construct the integral result of Simpson quadrature formula, namely $S_n=\frac{4Z2n-Zn}{3}$. It is same as the variable step trapezoidal quadrature method that you can obtain the result through the continuous division until the difference of two integral results is less than the precision requirement, namely

$|S_{2n} - S_n|<\varepsilon$. And now the S_{2n} is the finally integral result.

In order to prevent the endless division, generally it should limit the minimum step size. When the step size is less than the specific value of h0, even if it does not meet the accuracy requirements, the division will not continue.

Algorithm thought

(1)Calculating step size n and integral value s_1 according to the initial number of divided intervals

$h=\frac{b-a}{n}$ ， $Z_n=\frac{h}{2}[f(a)+f(b)]+\sum_{k=1}^{n-1}h\times f(a+k\times h)$， $s_1=z_n$

(2)Calculating the integral value s_2 after the step size is halved

$Z_{2n}=\frac{Zn}{2}+\sum_{k=0}^{n-1}\frac{h}{2}\times f\left(a+k\times h+\frac{h}{2}\right)$， $S_n=\frac{4Z2n-Zn}{3}$

(3) Updating the absolute value difference of the two integration, the number of the division, the integral value and the step size

$d=|s_2-s_1|$ ， $z_n=z_{2n}$， $s_1=s_2$， $n=2n$， $h=\frac{h}{2}$

(4)If $d>\varepsilon$ and $h>h0$, then return to the second process to continue division. Otherwise return to the value s_1 and exit the function

```
/*==================================
  //Function name：simps1
  //Function description：Variable step Simpson quadrature method
  //Input parameter：a integral lower limit，b integral upper limit
                    n0 initial partition number
                    eps precision requirement
                    h0 minimum step size
                    f pointer to the integrand
  return value：integral approximation
  ==================================*/
#include "math.h"
double simps1(a,b,n0,eps,h0,f)
double a,b,eps,h0;
int n0;
double (*f)()
{
int n,k;
double z,z2,s,s2,h,d,x,t;
n=n0;
h=(b-a)/n;
z=((*f)(a)+(*f)(b))/2.0;
for(k=1;k<n;k++)
{
x=a+k*h;
z=z+(*f)(x);
}
z=z*h;
s=z;
do
```

```
        {t=0.0;
          for(k=0;k<n;k++)
        {x=a+(k+0.5)*h;
          t=t+(*f)(x);
            }
        z2=(z+h*t)/2.0;
        s2=(4.0*z2-z)/3.0;
        d=fabs(s2-s);
        z=z2;
        s=s2;
        h=h/2.0;
        n=2*n;
        }while((d>eps)&&(h>h0));
        return(s);
    }
```

5Romberg quadrature method

Romberg quadrature method is also called the numerical integration of successive half accelerating convergence method. In the calculations it uses the improved order number and the decrease of the step two measures to improve the accuracy.

It can be verified that 2m order Newton Cotes formula can be obtained by the combination of two-order-lower Newton Cotes formula.

$R_{m+1}(h)=\frac{4^m Rm\left(\frac{h}{2}\right)-Rm(h)}{4^m-1}$

Where Rm（h）refers to the value calculated by 2m-2 order Newton Cotes formula when the step size is h. When h=0, $R_1(h)$ is the result calculated by the trapezoidal quadrature formula when the step size is h. This recursive formula is Romberg integral formula.

Constantly increase the order until the difference of the two integral values is less than the accuracy, namely

$$|R_{m+1}(h)-R_m(h)|<\varepsilon$$

$R_{m+1}(h)$ is the final integral result.

Algorithm thought

(1)Calculating step size n and integral value r_0 according to the initial number of divided intervals

$h=\frac{b-a}{n}$ ， $r_0=\frac{h}{2}[f(a)+f(b)]+\sum_{k=1}^{n-1} h\times f(a+k\times h)$

(2)Calculating the integral value z by trapezoidal formula with halved step size and taking z as the initial value of the recursive calculation.

$z=\frac{r_0}{2}+\sum_{k=0}^{n-1}\frac{h}{2}\times f(a+k\times h+\frac{h}{2})$

(3) Recursive calculation Romberg integral value rm, and update the storage of
ri (i=0,1, m-1)

```
t=1.0
for k=1,2,…m
do t=4t
    q=$\frac{t\times z-r_{k-1}}{t-1}$
    $r_{k-1}$=z
    z=q
```

(4)Updating the absolute value difference of the two integration, the number of the division, the integral value and the step size

$d=|q-r_{m-1}|$

$r_m=q$，m=m+1，n=2n，$h=\frac{h}{2}$

(5) If d<ε and m<10, then return to the second process to continue calculation. Otherwise return to the value q and exit the function

```
             /*===================================
            //Function name：rhg
            //Function description：Romberg quadrature method
            //Input parameter：a integral lower limit，b integral upper limit
                           n0 initial partition number
                           eps precision requirement
                           h0 minimum step size
                           f pointer to the integrand
            return value：integral approximation
            ===================================*/
    #include "math.h"
    double rhg(a,b,n0,eps,f)
    double a,b,eps;
    int n0;
    double (*f)();
    {
      int n,k,m;
      double r[10],z,g,h,d,x,t;
      m=1;
      n=n0;
      h=(b-a)/n;
      z=((*f)(a)+(*f)(b))/2.0;
    for(k=1;k<n;k++)
    {
      x=a+k*h;
      z=z+(*f)(x);
    }
    r[0]=z*h;
    do
    {
      z=0.0;
      for(k=0;k<n;k++)
      {x=a+(k+0.5)*h;z=z+(*f)(x);}
      z=(r[0]+h*z)/2.0;
      t=1.0;
      for(k=1;k<m+1;k++)
      {t=4.0*t;q=(t*z-r[k-1])/(t-1.0);
        r[k-1]=z;z=q;              }
      d=fabs(q-r[m-1]);
      r[m]=q;
      m=m+1;
      h=h/2.0;
      n=2*n;
    }while(d>eps)&&(m<10);
    return (q);
    }
```

6 Conclusion

Now taking the four methods to calculate the definite integral $t=\int_0^4 \frac{\exp(-x^2)}{1+x2} dx$, the results are as follows.

tzil1: t=0.67165
tzil2: t=0.67165
simp：t=0.67165
rbg: t=0.67165

Generally speaking, it usually obtains relatively large error when directly applying these formula into the whole interval [a, b]. So it can divide the whole integral interval into a number of small ones and take these quadrature formulas on each small interval. This kind of quadrature is called compound formula, which is just introduced in this paper.

Reference

[1]Daying Yi. Calculation method [M]. The second edition. Hangzhou；Zhejiang University Press,2002.

[2]Li He. Calculation method [M].Wuhan: Wuhan University of Hydraulic and Electrical Engineering Press, 1998.

[3]Xiuzhen Li. Mathematical experiment [M].Beijing: China Machine Industry Press, 2008.