УДК 519.6

АППРОКСИМАЦИЯ ДВОЙНЫХ И ТРОЙНЫХ ИНТЕГРАЛАХ КВАДРАТУРНЫМИ ФОРМУЛАМИ НА РАВНОМЕРНОЙ СЕТКЕ

Пастухов Д.Ф., Пастухов Ю.Ф.
Полоцкий государственный университет

Аннотация: На равномерной сетке получены квадратурные интегральные формулы свосьмым порядком погрешности. Приведены формулы и алгоритм для двойных на прямоугольнике и для тройных интегралов в параллелепипеде с восьмым порядком погрешности. Эти формулы обеспечивают уменьшение погрешности расчёта в 256 раз при уменьшении шага сетки в 2 раза. Численно показано, что все алгоритмы имеют 8 порядок погрешности.

Ключевые слова: порядок погрешности, шаблон весовых коэффициентов

APROXIMATION DOUBLE AND TRIPLE INTEGRAL NUMERICAL FORMULA ON EVEN NET

Pastuhov YU.F., Pastuhov D.F.
Polockiy state university

The Abstract: Numerical integral formulas c eighth rather inaccuracy are received On even net. The Broughted formulas and algorithm for double on rectangle and for triple integral in parallelepiped with eighth rather inaccuracy. These formulas provide the reduction to inaccuracy calculation in 256 once at reduction of the step of the net in 2 times. Is it Numerically shown that all algorithms have 8 orders to inaccuracy.

The Keywords: order to inaccuracy, pattern weight factor

Введение

Как известно, среди интегральных квадратурных формул наибольший порядок аппроксимации имеют квадратурные формулы Гаусса. Для их получения нужно построить ортогональный на отрезке $[a, b]$ многочлен степени n (все n корней многочлена расположены на отрезке $[a, b]$) с заданной весовой функцией $p(x) > 0, \forall x \in [a, b]$ [2]. Согласно теореме Абеля – Руффини произвольные многочлены степени большей четвёртой имеют корни, для которых невозможно указать замкнутую формулу для решений, т.е. формулу, содержащую только арифметические операции и корни произвольной степени. Можно проверить, что точность вычисления рациональной дроби и корня произвольной степени для любого компилятора имеет одинаковую двойную относительную точность(16 значащих цифр). Согласно теореме Гаусса ортогональный многочлен степени n имеет квадратурную формулу Гаусса точную для всех многочленов степени не выше $2n-1$ (порядок алгебраической точности)[2].

Интегральная формула Гаусса, все узлы и весовые коэффициенты которой могут быть записаны через радикалы, точна для всех многочленов степени не выше $2n-1 = 2*4-1 = 7$. Тогда квадратуры Гаусса записанные в радикалах могут иметь максимальный порядок погрешности равный 9. Корни ортогональных многочленов, равные узлам квадратурной формулы Гаусса (с числом больше четырёх) необходимо искать, например, с помощью формулы касательных Ньютона, что потребует не менее 250 итерации и более 1000 флопов. В данной работе построены квадратурные формулы с равномерным шагом, узлы и весовые коэффициенты которых выражаются арифметическими операциями (+,-,*,/) ,т.е. они имеют двойную точность. При этом их порядок погрешности равен 8(даже для составных квадратурных формул), что сравнимо с порядком погрешности для квадратурных формул Гаусса с 4 узлами. Если мы уменьшим шаг сетки в 2 раза, то модуль разности точного и приближённого значений интеграла уменьшится в $2^8 = 256$ раз (в величину погрешности заведомо добавляется два нуля после запятой). Поскольку $256^7 \approx 7.2*10^{16} \approx 10^{17}$, то относительная погрешность должна достигнуть двойной точности для чисел с плавающей запятой 10^{-16} при уменьшении шага сетки в $2^7 = 128$. Можно разделить отрезок на 126 частей с учётом формулы $n = 6k, k \in N$.

Математические приложения определённых интегралов подробно описаны в[1].

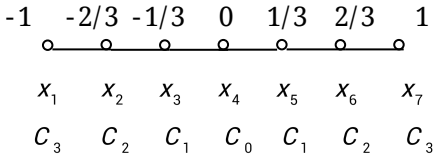
Рассмотрим последовательно одномерную формулу для отрезка, двухмерную для прямоугольника, трёхмерную для параллелепипеда.

1.Составная формула для отрезка, порядок аппроксимации.

Рассмотрим канонический отрезок[-1,1], на котором в силу симметрии узлы квадратурной формулы расположены симметрично относительно нуля, а весовые коэффициенты, соответствующие симметричным узлам имеют равные положительные

значения, как в формуле Симпсона. Разделим отрезок [-1,1] на 6 равных частей, т.е. используем 7 равноотстоящих узлов.

Получим формулу (по узлам $x_1 = -1,\ x_2 = -2/3,\ x_3 = 1/3,\ x_4 = 0,\ x_5 = 1/3,\ x_6 = 2/3,\ x_7 = 1$, $n = 6$):



$$\int_{-1}^{1} f(z)\,dz = C_0 f(0) + C_1(f(-1/3) + f(1/3)) + C_2(f(-2/3) + f(2/3)) + C_3(f(-1) + f(1)) \quad (1)$$

Подставляя в формулу (1) в качестве функции $f(z)$ степенные мономы, начиная с нулевой степени, используя только чётные степени мономов, получим систему уравнений:

$$f(z) \equiv 1,\ \int_{-1}^{1} f(z)\,dz = \int_{-1}^{1} dz = 2 = C_0 + 2C_1 + 2C_2 + 2C_3$$

$$f(z) = z^2,\ \int_{-1}^{1} f(z)\,dz = \int_{-1}^{1} z^2\,dz = \frac{2}{3} = \frac{2}{9}C_1 + \frac{8}{9}C_2 + 2C_3 \Leftrightarrow 3 = C_1 + 4C_2 + 9C_3$$

$$f(z) = z^4,$$

$$\int_{-1}^{1} f(z)\,dz = \int_{-1}^{1} z^4\,dz = \frac{2}{5} = \frac{2}{81}C_1 + \frac{32}{81}C_2 + 2C_3 \Leftrightarrow 81 = 5C_1 + 80C_2 + 405C_3$$

$$f(z) = z^6,$$

$$\int_{-1}^{1} f(z)\,dz = \int_{-1}^{1} z^6\,dz = \frac{2}{7} = \frac{2}{729}C_1 + \frac{128}{729}C_2 + 2C_3 \Leftrightarrow 729 = 7C_1 + 448C_2 + 5103C_3$$

Нужно решить систему линейных уравнений с 4 неизвестными.

$$\begin{cases} 2 = C_0 + 2C_1 + 2C_2 + 2C_3 \\ 3 = C_1 + 4C_2 + 9C_3 \\ 81 = 5C_1 + 80C_2 + 405C_3 \\ 729 = 7C_1 + 448C_2 + 5103C_3 \end{cases} \Leftrightarrow C_0 = \frac{68}{105},\ C_1 = \frac{9}{140},\ C_2 = \frac{18}{35},\ C_3 = \frac{41}{420} \quad (2)$$

Из формул (1) и (2) получим формулу (3):

$$\int_{-1}^{1} f(z)\,dz = S_f = \frac{68}{105} f(0) + \frac{9}{140}(f(-1/3) + f(1/3)) + \frac{18}{35}(f(-2/3) + f(2/3)) + \frac{41}{420}(f(-1) + f(1)) \quad (3)$$

В частности, если $f(z) \equiv 1,\ \int_{-1}^{1} f(z)\,dz = 2 = \frac{68}{105} + 2\left(\frac{9}{140} + \frac{18}{35} + \frac{41}{420}\right)$ - имеем длину канонического отрезка [-1,1].

Шаблон весовых коэффициентов для составной формулы по решению системы уравнений (2)(коэффициенты на внутренних границах удваиваются):

$$\left\{\frac{41}{420}, \frac{18}{35}, \frac{9}{140}, \frac{68}{105}, \frac{9}{140}, \frac{18}{35}, \frac{41}{210}, \frac{18}{35}, \frac{9}{140}, \frac{68}{105}, \frac{9}{140}, \frac{18}{35}, \frac{41}{420}\right\} (n = 12)$$

Для интеграла на отрезке $[a, b]$ составная формула (на базе (3)) с равномерным шагом сетки $n = 6k,\ k \in N$:

$$I_1 = \int_a^b f(x)\,dx = 3hS_f = \frac{3(b-a)}{n} S_f,\ h = \frac{(b-a)}{n},\ S_f = \sum_0^n C_j f(x_j),\ x_j = a + h * j \quad (4)$$

Где в квадратурной формуле (4) значение однократного интеграла I_1 пропорционально шагу равномерной сетки $h = \frac{(b-a)}{n}$, коэффициент 3 выбран согласно принципу соответствия (при делении отрезка[-1,1] на 6 равных частей $h = 1/3, 3h = 1, S_f = 2, I_1 = 3hS_f = 2 = b - a, b = 1, a = -1$)

Удобно разбить S_f ($n = 6k, k \in N$) в формуле(4) на 5 отдельных слагаемых

$$S_f = S_1 + S_2 + S_3 + S_4 + S_5 \quad (5)$$

1) если $j = 0$ *или* $j = n$: $S_1 = \frac{41}{420}(f(a) + f(b)), C_j = \frac{41}{420};$ (5.1)

2) если $j \equiv 1 \bmod 6$ *или* $j \equiv 5 \bmod 6$: $S_2 = \frac{18}{35}\left(\sum_{s=0}^{k-1} f(a + h(1 + 6s)) + \sum_{s=0}^{k-1} f(a + h(5 + 6s))\right)$ (5.2)

$C_j = \frac{18}{35};$

3) если $j \equiv 2 \bmod 6$ *или* $j \equiv 4 \bmod 6$: $S_3 = \frac{9}{140}\left(\sum_{s=0}^{k-1} f(a + h(2 + 6s)) + \sum_{s=0}^{k-1} f(a + h(4 + 6s))\right)$ (5.3)

$C_j = \frac{9}{140};$

4) если $j \equiv 3 \bmod 6$: $S_4 = \frac{68}{105}\left(\sum_{s=0}^{k-1} f(a + h(3 + 6s))\right)$ (5.4)

$C_j = \frac{68}{105};$

5) если $j \equiv 0 \bmod 6$: $S_5 = \frac{41}{210}\left(\sum_{s=1}^{k-1} f(a + h(6s))\right)$ (5.4)

$C_j = \frac{41}{210};$

Алгоритмическое разбиение квадратуры S_f в(4) на 4 слагаемых (формулы (5)) обусловлено экономией машинного времени и уменьшением суммарной ошибки. Действительно, в (5) мы складываем все узловые значения функции с одинаковым весом C_j и только один раз умножаем полученную сумму на этот вес C_j.

Рассмотрим пример: $b = 2, a = 0, f(x) = x^9, I(f) = \int_a^b f(x)\,dx = \frac{x^{10}}{10}\Big|_0^2 = 102.4$

Программа соответствует построенному алгоритму и формулам (4), (5.1-5.4), написана на языке C++ и приведена ниже:

```
////////////////////////////////////////////////////////////////////////////////
#include<stdio.h>
#include<math.h>

double f(double x);
int const n=60,k=n/6;
 main()
{
int s,j;
        double a,b,c,d,h1,h2,bb[n+1],x,y,x1;
        double sum,sum1,sum2,sum3,bac1,bac,delta,epsilon;
        printf("n=%d k=%d\n",n,k);
```

```
        a=0.0;
        b=2.0;
        h1=(b-a)/double(n);
        bac1=0.0;
        sum=0.0;
        sum1=0.0;
        sum2=0.0;
sum3=0.0;
        for(s=0;s<=k-1;s++)
        {
        x=a+h1*(1.0+6.0*double(s));
        x1=a+h1*(5.0+6.0*double(s));
        sum=sum+f(x)+f(x1);
        }
        sum=sum*(18.0/35.0);
                for(s=0;s<=k-1;s++)
{
x=a+h1*(2.0+6.0*double(s));
x1=a+h1*(4.0+6.0*double(s));
sum1=sum1+f(x)+f(x1);
}
sum1=sum1*(9.0/140.0)           ;
for(s=0;s<=k-1;s++)
{
        x=a+h1*(3.0+6.0*double(s));
        sum2=sum2+f(x);
        }
        sum2=sum2*(68.0/105.0);
for(s=1;s<=k-1;s++)
{
x=a+6.0*h1*double(s);

sum3=sum3+f(x)          ;
}
sum3=sum3*(41.0/210.0);

bac1=sum        +sum1+sum2+sum3+(f(a)+f(b))*(41.0/420.0);
bac1=bac1*3.0*h1;

delta=102.4-bac1;
epsilon=delta/bac1;
printf("int 1 =%14lf, exact=%14lf delta=%14lf eps=%16lf\n",bac1, 102.4,delta,epsilon);
}
double f(double x)
{
        return x*x*x*x*x*x*x*x*x*x;
}
//////////////////////////////////////////////////////////////////////////
```

При n=120 k=20 программа возвращает значения:

int 1 =102.40000000000461, exact=102.40000000000001 delta=-0.00000000000460 eps=-0.0000000000000450

Press any key to continue

При n=60 k=10 программа ворвращает значения:

int 1 =102.40000000118513, exact=102.40000000000001 delta=-0.00000000118513 eps=-0.0000000000115735

Press any key to continue

Откуда видно, что абсолютная погрешность изменяется

$$\left|\frac{delta_2}{delta_1}\right| = \frac{0.00000000\quad 118513}{0.00000000\quad 000460} = 257,6 \approx 2^8$$

Т.е. порядок погрешности полученной составной одномерной формулы равен 8.

Параметр $eps = \frac{delta}{S_f} = \frac{I(f) - S_f}{S_f}$ – относительная погрешность не должна быть меньше 10^{-16}, выполнение асимптотики (порядок погрешности $p = 8$) можно ожидать при $|eps| \geq 10^{-15}$.

2.Построение двумерного алгоритма

Двумерную квадратурную формулу на прямоугольнике получим как декартовое произведение двух одномерных формул, заданных на отрезке, по аналогии с формулой (4):

$$I_2 = \int_a^b \int_c^d f(x, y)\,dxdy = 9 h_1 h_2 S_f,\ h_1 = \frac{(b-a)}{n},\ h_2 = \frac{(d-c)}{n},\ S_f = \sum_{j=0}^{n} \sum_{i=0}^{n} C_{i,j} f(y_j x_i), \quad (6)$$

$$x_i = a + h_1 * i,\ y_j = c + h_2 * j;\ i,\ j = \overline{0, n}$$

Где: $C_{i,j}$ - весовые коэффициенты определяются формулой (7)

В частности, при $f(x, y) \equiv 1$, $I_2 = 4$, $a = c = -1$, $b = d = 1$ получаем площадь канонического квадрата со стороной 2([-1,1]*[-1,1]). $h_1 = h_2 = 1/3$, $S_f = 4$ - как повторная сумма при фиксированной внешней переменной суммирования j (при этом сумма по внутренней переменной i равна 2 при фиксированной переменной j) в соответствии с формулой (7).

$$\begin{cases} C_{i,j} = C_i * C_j;\ i,\ j = \overline{0, n} \\ C_i = \frac{18}{35} (если\ i \equiv 1 \bmod 6\ \ или\ \ i \equiv 5 \bmod 6) \\ C_i = \frac{9}{140} (если\ i \equiv 2 \bmod 6\ \ или\ \ i \equiv 2 \bmod 6) \\ C_i = \frac{68}{105} (если\ i \equiv 3 \bmod 6\ ) \\ C_i = \frac{41}{210} (если\ i \equiv 0 \bmod 6, i > 0, i < n) \\ C_i = \frac{41}{420} (если\ \ i = 0, i = n) \end{cases} \quad (7)$$

Рассмотрим пример: найти $I_2 = \int_0^{10} \int_0^{10} \exp(x) y^4\,dxdy = \frac{y^5}{5} (\exp\ x) \Big|_0^{10} = \frac{10^5}{5} (\exp\ 10 - 1)$.

В данном примере выбраны большие размеры области интегрирования, кроме того экспоненциальная функция растёт значительно быстрее степенной, именно при этих условиях можно проследить асимптотику для порядка погрешности (p=8).

Согласно формулам (6),(7) построим алгоритм для программы на Visual C++ 6.0:

```
////////////////////////////////////////////////////////////////////
#include<stdio.h>
#include<math.h>

double f(double x, double y);
int const n=120,k=n/6;
 main()
{
int s,j;
        double a,b,c,d,h1,h2,bb[n+1],x,y,x1;
        double sum,ss,sum1,sum2,sum3,bac1,bac,bac2,delta,epsilon;
        printf("n=%d k=%d\n",n,k);
        a=0.0;
        b=10.0;
        c=a;
        d=b;
                h1=(b-a)/double(n);
```

```
            h2=(d-c)/double(n);
            bac1=0.0;
            bac2=0.0;
            sum=0.0;
            sum1=0.0;
            sum2=0.0;
sum3=0.0;
for (j=0;j<=n;j++)
{
y=c+h2*double(j);
bac1=0.0;
sum=0.0;
            sum1=0.0;
            sum2=0.0;
            sum3=0.0;
//////////////////////////////
            for(s=0;s<=k-1;s++)
            {
            x=a+h1*(1.0+6.0*double(s));
            x1=a+h1*(5.0+6.0*double(s));
            sum=sum+f(x,y)+f(x1,y);
            }
            sum=sum*(18.0/35.0);
                  for(s=0;s<=k-1;s++)
{
x=a+h1*(2.0+6.0*double(s));
x1=a+h1*(4.0+6.0*double(s));
sum1=sum1+f(x,y)+f(x1,y);
}
sum1=sum1*(9.0/140.0)           ;
for(s=0;s<=k-1;s++)
{
            x=a+h1*(3.0+6.0*double(s));
            sum2=sum2+f(x,y);
            }
            sum2=sum2*(68.0/105.0);
for(s=1;s<=k-1;s++)
{
x=a+6.0*h1*double(s);

sum3=sum3+f(x,y)        ;
}
sum3=sum3*(41.0/210.0);

bac1=sum        +sum1+sum2+sum3+(f(a,y)+f(b,y))*(41.0/420.0);
if(j==0  ||j==n)
{
             bb[j]= bac1*(41.0/420.0);

}
else       if(j%6==1  ||j%6==5)
{
             bb[j]=  bac1*(18.0/35.0);
}
else if(j%6==2  ||j%6==4)
{
            bb[j]= bac1*(9.0/140.0);
}
else if(j%6==3)
{
            bb[j]=   bac1*(68.0/105.0);
}
```

```
else if(j%6==0 && j>0 && j<n)
{
        bb[j]=bac1*(41.0/210.0);
        }
//////////////////////
}
for(j=0;j<=n;j++)
{
        bac2=bac2+bb[j];
}

bac2=bac2*(9.0*h2*h1);
ss=(exp(10.0)-1.0)*(10.0*10.0*10.0*10.0*10.0*10.0)/5.0;
delta=ss-bac2;
epsilon=delta/bac1;
printf(" eps=%.16lf\n",epsilon);

printf("int 2 =%.10lf, exact=%.10lf delta=%.10lf\n",bac2,ss,delta);
}
double f(double x, double y)
{
        return exp(x)*y*y*y*y;
}

////////////////////////////////////////////////////////////////////////////////////////
```

При n=120 k=20 программа возвращает значения:
eps=-0.0000000000012346
int 2 =440509315.8972221000, exact=440509315.8961343800 delta=-0.0010877252
Press any key to continue
При n=240 k=40 программа возвращает значения:
eps=-0.0000000000000022
int 2 =440509315.8961383100, exact=440509315.8961343800 delta=-0.0000039339
Press any key to continue

Откуда видно, что абсолютная погрешность изменяется

$$\left|\frac{delta_1}{delta_2}\right| = \frac{0.00108772\quad 52}{0.00000393\quad 39} = 276,5 > 256 = 2^8$$

Т.е. порядок погрешности полученной составной двумерной формулы не меньше 8.

Параметр $eps = \frac{delta}{S_f} = \frac{I(f) - S_f}{S_f} - относительная\ погрешность$ не должна быть меньше 10^{-16}, выполнение асимптотики (порядок погрешности $p = 8$) можно ожидать при $|eps| \geq 10^{-15}$, что в последних двух расчётах выполнено.

3.Построение алгоритма для трёхмерной формулы

Учитывая предыдущий опыт построения алгоритма I_2 по I_1, получим алгоритм трёхмерной квадратурной формулы для объёмного интеграла I_3 как декартово произведение трёх одномерных на отрезке квадратурных формул:

$$I_3 = \int_a^b \int_c^d \int_e^f f(x, y, z)\,dxdydz = 27\,h_1 h_2 h_3 S_f,\ h_1 = \frac{(b-a)}{n},\ h_2 = \frac{(d-c)}{n},\ h_3 = \frac{(f-e)}{n}$$

$$S_f = \sum_{k=0}^{n}\sum_{j=0}^{n}\sum_{i=0}^{n} C_{i,j,k}\, f(x_i y_j z_k), \qquad (8)$$

$$x_i = a + h_1 * i,\ y_j = c + h_2 * j,\ z_k = c + h_3 * k;\ i, j, k = \overline{0, n}$$

$$\begin{cases} C_{i,j,k} = C_i * C_j * C_k; i, j, k = \overline{0, n} \\ C_i = \frac{18}{35} (\text{если } i \equiv 1 \bmod 6 \text{ или } i \equiv 5 \bmod 6) \\ C_i = \frac{9}{140} (\text{если } i \equiv 2 \bmod 6 \text{ или } i \equiv 2 \bmod 6) \\ C_i = \frac{68}{105} (\text{если } i \equiv 3 \bmod 6) \\ C_i = \frac{41}{210} (\text{если } i \equiv 0 \bmod 6, i > 0, i < n) \\ C_i = \frac{41}{420} (\text{если } i = 0, i = n) \end{cases} \quad (9)$$

В частности, при $f(x, y, z) \equiv 1$, $I_3 = 8$, $a = c = e = -1$, $b = d = f = 1$ получаем объём канонического куба со стороной 2([-1,1]*[-1,1]*[-1,1]). $h_1 = h_2 = h_3 = 1/3$, $S_f = 8$ - как повторная тройная сумма при фиксированных внешних переменной суммирования j,k (при этом сумма по внутренней переменной i равна 2 при фиксированных переменных j,k) в соответствии с формулой (9).

Приведём пример: найти $I_3 = \int_0^4 \int_0^4 \int_0^4 \exp(x) y^4 z^5 dxdydz = \frac{y^5}{5} \frac{z^6}{6} (\exp x)\Big|_0^4 = \frac{4^{11}}{30} (\exp 4 - 1)$

Согласно формулам (8),(9) построим алгоритм для программы на Visual C++ 6.0:

```
    #include<stdio.h>
#include<math.h>

double f(double x, double y, double z);
int const n=120,k=n/6;
 main()
{
int s,j,j1;
        double a,b,c,d,e,f1,h1,h2,h3,aa[n+1],bb[n+1],x,y,z,x1,xx;
        double sum,sum1,sum2,sum3,sum4,epsilon,bac1,bac,bac2,bac3,delta;
                printf("n=%d k=%d\n",n,k);
        a=0.0;
        c=a;
        e=a;
        b=4.0;
        d=b;
        f1=b;
        h1=(b-a)/double(n);
        h2=(d-c)/double(n);
        h3=(f1-e)/double(n);
        for(j1=0;j1<=n;j1++)
        {
        z=e+h3*double(j1);
for(j=0;j<=n;j++)
{
        bac1=0.0;

        y=c+h2*double(j);
        sum=0.0;
        sum1=0.0;
        sum2=0.0;
        sum3=0.0;
        for(s=0;s<=k-1;s++)
        {
        x=a+h1*(1.0+6.0*double(s));
        x1=a+h1*(5.0+6.0*double(s));
        sum=sum+f(x,y,z)+f(x1,y,z) ;
        }
```

```
        sum=sum*(18.0/35.0);
                for(s=0;s<=k-1;s++)
{
x=a+h1*(2.0+6.0*double(s));
        x1=a+h1*(4.0+6.0*double(s));
sum1=sum1+f(x,y,z)+f(x1,y,z)    ;
}
sum1=sum1*(9.0/140.0)           ;
for(s=0;s<=k-1;s++)
{
        x=a+h1*(3.0+6.0*double(s));
        sum2=sum2+f(x,y,z);
        }
        sum2=sum2*(68.0/105.0);
for(s=1;s<=k-1;s++)
{
        x=a+h1*(6.0*double(s));
        sum3=sum3+f(x,y,z);
        }
        sum3=sum3*(41.0/210.0);
bac1=3.0*h1*(sum        +sum1+sum2+sum3+(41.0/420.0)*(f(a,y,z)+f(b,y,z)));
if( (j==n) ||(j==0))
{
bb[j]     = (41.0/420.0)*bac1;
}
else if((j%6==1)  ||(j%6==5) )
{
bb[j]     = (18.0/35.0)*bac1;
}
else if((j%6==2)  ||(j%6==4) )
{
bb[j]     = (9.0/140.0)*bac1;
}
else if(j>0 && j<n && j%6==0)
{
        bb[j]=(41.0/210.0)*bac1;
}
else if(j%6==3)
{
        bb[j]=(68.0/105.0)*bac1;
}
}
bac=0.0;
for(j=0;j<=n;j++)
{
bac=bac+bb[j];
}
bac=bac*h2*3.0;
if(j1==0  || j1==n)
{
aa[j1]    =(41.0/420.0)*bac;
}
else if(j1%6==1 ||j1%6==5)
{
aa[j1]=(18.0/35.0)*bac;
}
else if(j1%6==2 ||j1%6==4)
{
        aa[j1]=(9.0/140.0)*bac;
}
else if(j1%6==0 && j1>0 && j1<n)
{
```

```
            aa[j1]=(41.0/210.0)*bac;
}
else if(j1%6==3 )
{
            aa[j1]=(68.0/105.0)*bac;
}
}
bac2=0.0;
for(s=0;s<=n;s++)
{
            bac2=bac2+aa[s];
            }
            bac2=bac2*h3*3.0;
            xx=(-1.0+exp(4.0))*1024.0*1024.0*4.0/30.0;
delta=bac2-xx;
epsilon=delta/bac2;
printf(" eps=%.16lf\n",epsilon);

printf("int 3 =%.25lf, exact=%.25lf delta=%.25lf\n",bac2, xx,delta);
}
double f(double x, double y, double z)
{
            return exp(x)*y*y*y*y*z*z*z*z*z;
}
```

При n=60 k=10 программа возвращает значения:

```
eps=0.0000000000004155
int 3 =7493564.5025570132000000000000000, exact=7493564.5025538998000000000000000
delta=0.0000031134113669395447000
Press any key to continue
n=120 k=20
eps=0.0000000000000014
int 3 =7493564.5025539100000000000000000, exact=7493564.5025538998000000000000000
delta=0.0000000102445483207702640
Press any key to continue
```

Откуда видно, что абсолютная погрешность изменяется

$$\left|\frac{delta_1}{delta_2}\right| = \frac{0.00000311\quad 3411366939\quad 5447000}{0.00000001\quad 0244548320\quad 7702640} = 303.9 > 256 = 2^8$$

Т.е. порядок погрешности полученной составной трёхмерной формулы не меньше 8.

Параметр $eps = \frac{delta}{S_f} = \frac{I(f) - S_f}{S_f} - относительная\ погрешность$ не должна быть меньше 10^{-16}, выполнение асимптотики (порядок погрешности $p = 8$) можно ожидать при $|eps| \geq 10^{-15}$, что в последних двух расчётах выполнено.

Выводы

1) Поскольку интегральная составная квадратурная формула для центральных прямоугольников (1 узел аппроксимации) имеет второй порядок погрешности, составная формула Симпсона (3 узла) имеет 4 порядок погрешности, то формула для 5 узлов должна иметь 6 порядок погрешности, наконец полученная в данной работе составная интегральная квадратурная формула на 7 узлах при делении канонического отрезка на 6 частей может иметь 8 порядок погрешности. Данное предположение подтверждено численно.
2) Приведены квадратурные интегральные формулы на равномерной сетке с 8 порядком погрешности, алгоритм для двойного определённого интеграла на прямоугольнике и непрерывных функций без особенностей.
3) Приведены квадратурные интегральные формулы на равномерной сетке с 8 порядком погрешности, алгоритм для тройного определённого интеграла в параллелепипеде и непрерывных функций без особенностей.
4) Численно показано, что все три алгоритма имеют восьмой порядок погрешности.

Литература

1)Математический анализ в вопрросах изадачах В.Ф.Бутузов, Н.Ч.Крутицкая, Г.Н.Медведев, А.А.Шишкин;
Под редакцией В.Ф.Бутузова - 4-е изд., исправ. – М.: Физико-математическая литература, 2001.-480 с.
2)Н.С.Бахвалов, Н.П.Жидков, Г.М.Кобельков. Численные методы. – 7 – е изд.: БИНОМ. Лаборатория знаний,2011. – 636 с. – (Классический университетский учебник).