Криворожский государственный педагогический институт
Кафедра информатики и прикладной математики

А.П. Полищук
С.А. Семериков

ЧИСЛЕННЫЕ МЕТОДЫ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ МЕТОДОЛОГИИ

Учебное пособие

Раздел 5: Дифференциальные уравнения
Двухточечные краевые задачи

Кривой Рог

1999

Полищук А.П., Семериков С.А. Численные методы в объектно-ориентированной методологии. Раздел «Дифференциальные уравнения». Двухточечные краевые задачи. Учебное пособие. – Кривой Рог: КГПИ, 1998. – 12 с.

Авторы:

Полищук А.П.	к. т. н., с. н. с., доцент кафедры информатики и прикладной математики.
Семериков С.А.	магистр математики.

Рецензенты:

Рашевский Н.А.	к. ф.-м. н., доцент кафедры математики (КГПИ)
Теплицкий И.А.	учитель-методист, зам. директора по научной работе (Центрально-Городская гимназия)

Под общей редакцией доктора физико-математических наук, профессора В.Н. Соловьёва.

Рекомендовано к печати на заседании кафедры информатики КГПИ, протокол №1 от 31.08.98 г.

Підп. до друку 28.12.98
Друк №3. Друк офсетний
Умовн. фарбо-відб. 0,5
Тираж 300

Формат 80x84 1/16.
Умовн. друк. арк. 0,6
Зам. №12-2482

КДПІ, 324086, Кривий Ріг-86, пр. Гагаріна, 54

Криворізька міська друкарня
324050, Кривий Ріг-50, пр. Металургів, 28.

Оглавление

5.5. Двухточечные краевые задачи

5.5.1. Метод конечных разностей для линейных краевых (граничных) задач

Пусть задано обыкновенное дифференциальное уравнение порядка p или эквивалентная ему система из p уравнений первого порядка. Необходимо найти решение $f(t)$ уравнения на интервале $[t_0, t_n]$, удовлетворяющее граничным условиям функциональной связи между искомой функцией и ее производными

$$\varphi_k(f(t_k), f^{(1)}(t_k), \ldots, f^{(p-1)}(t_k)=0.$$

Для существования и единственности решения граничной задачи необходимо, чтобы количество граничных условий k совпадало с порядком заданного дифференциального уравнения. Следовательно, граничную задачу можно поставить для уравнения, порядок которого не ниже второго.

Граничные условия могут быть заданы в любой из точек интервала и, в частности, на границах интервала $[t_0, t_n]$ и в том числе на его границах. Для упрощения выкладок мы и будем рассматривать граничную задачу для линейного дифференциального уравнения второго порядка с переменными коэффициентами

$$f^{(2)}(t)+p(t)f^{(1)}(t)+q(t)f(t)=r(t).$$

Граничные условия зададим в виде линейной комбинации искомой функции $f(t)$ и ее первой производной $f^{(1)}(t)$ в точках t_0 и t_n:

$$g_1f(t_0)+g_2f^{(1)}(t_0)=g_3,\ g_4f(t_n)+g_5f^{(1)}(t_n)=g_6.$$

На первом этапе решения задачи составляются конечно-разностные схемы исходного дифференциального уравнения и граничных условий, в которых производные заменяются их приближенными выражениями через конечные разности.

Интервал $[t_0, t_n]$ разобьем на n–1 часть с шагом $h=(t_n–t_0)/n$.

Искомую функцию $f(t)$ в окрестности узла t_i представим в виде разложения в ряд Тейлора

$$f(t)=f(t_i)+(t–t_i)f^{(1)}(t_i)+(1/2)(t–t_i)^2f^{(2)}(t_i)+\ldots$$

В точках t_{i+1} и t_{i-1}, отстоящих на расстоянии h от узла t_i, пользуясь разложением, получим следующие значения функции $f(t)$:

$$f(t_{i+1})=f(t_i)+hf^{(1)}(t_i)+(1/2)h^2f^{(2)}(t_i)+\ldots$$
$$f(t_{i-1})=f(t_i)–hf^{(1)}(t_i)+(1/2)h^2f^{(2)}(t_i)+\ldots$$

Приближенные значения для первой и второй производных

функции $f(t)$ в точке t_i получим путем вычитания и сложения левых и правых частей последних выражений:

$$f^{(1)}(t_i)=\frac{f(t_{i+1})-f(t_{i-1})}{2h}+O(h^2),$$

$$f^{(2)}(t_i)=\frac{f(t_{i+1})-2f(t_i)+f(t_{i-1})}{h^2}+O(h^2).$$

Заменяя производные в исходном уравнении их приближенными значениями, получим конечно-разностную схему этого уравнения

$$\frac{f(t_{i+1})-2f(t_i)+f(t_{i-1})}{h^2}+p(t_i)\frac{f(t_{i+1})-f(t_{i-1})}{2h}+q(t_i)f(t_i)=r(t_i),\ i=1,\ ...,\ n.$$

Аналогичным образом заменим граничные условия их конечно-разностными представлениями

$$g_1(t_0)f(t_0)+g_2\frac{f(t_1)-f(t_{-1})}{2h}=g_3(t_0),$$

$$g_4(t_n)f(t_n)+g_5\frac{f(t_{n+1})-f(t_{n-1})}{2h}=g_6(t_n).$$

Совокупность разностных схем уравнения и граничных условий представляет собой систему $(n+3)$ линейных алгебраических уравнений относительно неизвестных $f(t_{-1})$, $f(t_0)$, ..., $f(t_{n+1})$. Матрицы этой системы приводятся к трехдиагональному виду, и тогда систему можно решить методом прогонки. Из первого конечно-разностного уравнения для граничных условий выразим неизвестное $f(t_{-1})$ и подставим его в конечно-разностное уравнение системы при $i=0$, в результате получим линейное соотношение

$$f(t_0)=k_0-l_0f(t_1),\ \text{где}$$

$$k_0=\frac{h^2r(t_1)g_2+h(2-hp(t_1))g_3}{(h^2q(t_1)-2)g_2+h(2-hp(t_1))g_1},$$

$$l_0=\frac{2g_2}{(h^2q(t_1)-2)g_2+h(2-hp(t_1))g_1}$$

При $i=1$ уравнение $f(t_0)=k_0-l_0f(t_1)$ приводится к виду $f(t_1)=k_1-l_1f(t_2)$, а для произвольного значения i получим $f(t_i)=k_i-l_if(t_{i+1})$. Пограничные коэффициенты k_i, l_i вычисляются по рекуррентным формулам:

$$k_i = \frac{2h^2 r(t_i) - (2 - hp(t_i))k_{i-1}}{2h^2 q(t_i) - 4 - (2 - hp(t_i))l_{i-1}},$$

$$l_i = \frac{2 + hp(t_i)}{2h^2 q(t_i) - 4 - (2 - hp(t_i))l_{i-1}},$$

полученным путем приведения разностного уравнения системы к форме $f(t_i)=k_i-l_i f(t_{i+1})$ после замены значений $f(t_{i-1})=k_{i-1}-l_{i-1}f(t_i)$.

Из конечно-разностной схемы для второго граничного условия после подстановки выражений для $f(t_{n+1})$ и $f(t_{n-1})$ из $f(t_i)=k_i-l_i f(t_{i+1})$ и $f(t_{i-1})=k_{i-1}-l_{i-1}f(t_i)$ найдем соотношение для вычисления неизвестного $f(t_n)$:

$$f(t_n) = \frac{2hg_6 + (k_{n-1} - k_n / l_n)g_5}{2hg_4 + (l_{n-1} - 1/l_n)g_5}.$$

Затем в процессе обратного хода метода прогонки, полагая последовательно $i=n-1, n-2, \ldots, 0$, по формуле $f(t_i)=k_i-l_i f(t_{i+1})$ вычисляются значения остальных искомых величин $f(t_{n-1}), f(t_{n-2}), \ldots, f(t_0)$. Так как погрешности аппроксимации первой и второй производной пропорциональны квадрату шага, этот вариант метода имеет второй порядок.

5.5.2. Метод стрельбы для граничных задач

Метод стрельбы мы рассмотрим на примере граничной задачи для *системы* обыкновенных дифференциальных уравнений второго порядка

$$\begin{cases} \dfrac{df_1(t)}{dt} = F_1(t, f_1(t), f_2(t)) \\ \dfrac{df_2(t)}{dt} = F_2(t, f_1(t), f_2(t)) \end{cases}.$$

Необходимо найти решение этой системы уравнений на интервале $[t_0, t_n]$, удовлетворяющее граничным условиям

$$\varphi(t_0, f_1(t_0), f_2(t_0))=0,$$
$$\psi(t_n, f_1(t_n), f_2(t_n))=0.$$

Сущность метода стрельбы заключается в сведении решения граничной задачи к многократному решению задачи Коши для заданной системы дифференциальных уравнений.

Предположим, что

$$f_1(t_0)=\varsigma,$$

где ς – произвольное число, которое можно задать, используя априорную информацию о характере решения $f_1(t)$. Подставим предполагаемое значение ς в первое граничное условие

$$\varphi(t_0, \varsigma, f_2(t_0))=0.$$

Теперь последнее соотношение является уравнением относительно одного неизвестного $f_2(t)$. Для простых функций φ удается записать аналитическое решение указанного уравнения. В общем случае решение этого уравнения находится одним из численных методов. В результате аналитического или численного решения получим

$$f_2(t_0, \varsigma)=\beta.$$

Таким образом, сформулирована задача Коши для исходной системы дифференциальных уравнений с начальными условиями $f_1(t_0)=\varsigma$ и $f_2(t_0, \varsigma)=\beta$ в точке t_0. Решение задачи Коши можно провести одним из известных методов решения для подобных задач и получить с необходимой точностью значения функций $f_1(t)$ и $f_2(t)$ в точке $t=t_n$: $f_1(t_n, \varsigma)$ и $f_2(t_n, \varsigma)$. Последние результаты поставим во второе граничное условие

$$\psi(t_n, f_1(t_n, \varsigma), f_2(t_n, \varsigma))=0,$$

которое не будет выполняться, так как число ς выбрано нами произвольным.

Последнее соотношение можно рассматривать как уравнение $\psi(\varsigma)=0$ относительно переменной ς. Значение $\varsigma=\varsigma^*$, являющееся корнем этого уравнения, удовлетворяет каждому граничному условию. Следовательно, решениями поставленной граничной задачи будут функции $f_1(t, \varsigma^*)$ и $f_2(t, \varsigma^*)$, определенные на интервале $[t_0, t_n]$ для задачи Коши.

Решение уравнения $\psi(\varsigma)=0$ требует большого объема вычислений, так на каждой итерации необходимо осуществлять интегрирование задачи Коши для исходной системы уравнений. Каждая итерация порождает для функций $f_1(t, \varsigma)$ и $f_2(t, \varsigma)$ траектории, которые в зависимости от параметра ς либо приводят, либо не приводят к цели – обращению левой части уравнения $\psi=0$ в нуль. Отсюда и происходит название метода стрельбы. Самым экономичным методом решения таких уравнений является метод Ньютона, но он требует на каждой итерации вычисления не только левой части $\psi(\varsigma)$, но и ее производной по параметру ς, что

вызывает необходимость интегрирования лишней пары дифференциальных уравнений. Поэтому для решения уравнения $\psi(t_n, f_1(t_n, \varsigma), f_2(t_n, \varsigma))=0$ чаще используют метод секущих, алгоритм которого в конкретном случае выглядит так:

$$\varsigma_{i+1}=\varsigma_i-\frac{\varsigma_i-\varsigma_{i-1}}{\psi(\varsigma_i)-\psi(\varsigma_{i-1})}\psi(\varsigma_i),$$ где i – номер итерации.

Погрешность решения зависит от выбранного шага и метода интегрирования задачи Коши, а также от погрешности вычисления ς^* методом секущих. Первая из этих погрешностей определит и полосу шума для функции $\psi(\varsigma)$, при попадании в которую вычисляемых значений $\psi(\varsigma_i)$ итерационный процесс уточнения ς приходится прекращать. Такую взаимосвязь погрешностей приходится учитывать, чтобы избежать лишних итераций метода секущих. Чтобы избежать рекурсивного обращения к подпрограмме метода секущих при решении уравнений $\varphi(f_2(t))=0$ и $\psi(\varsigma)=0$, соответствующий блок рекомендуется переписать дважды.

Алгоритм метода стрельбы существенно упрощается для линейных граничных задач, когда правые части системы ОДУ и граничные условия представляют собой линейную комбинацию функций $f_1(t)$ и $f_2(t)$:

$$\frac{df_1(t)}{dt}=f_2(t),$$

$$\frac{df_2(t)}{dt}=r(t)-p(t)f_2(t)-q(t)f_1(t),$$

$$g_1f_1(t_0)+g_2f_2(t_0)=g_3,$$
$$g_4f_1(t_n)+g_5f_2(t_n)=g_6.$$

Начальные условия соответствующей задачи Коши принимают вид:

$$f_1(t_0)=\varsigma, f_2(t_0)=(g_3-g_1\varsigma)/g_2.$$

Решения задачи Коши будут иметь линейную зависимость от ς, поэтому и левая часть уравнения $\psi(\varsigma)=0$ будет линейной функцией аргумента ς. Следовательно, значение ς_2, найденное по формуле секущих, при $i=1$ будет точным корнем уравнения. Таким образом, для линейной граничной задачи достаточно трижды решить задачу Коши.

5.5.3. Программная реализация класса граничных задач

```
#include <values.h>
#include <conio.h>
#include "matrix.h"

/*Для удобства определим пару типов для работы с вещественными матрицами и векторами*/
typedef vector<double> dvector;
typedef matrix<double> dmatrix;

/*Этот класс предназначен для тестирования методов решения граничных задач*/
class GrZadDemo
{
  double min, max, grusl[6];/*минимум, максимум и массив граничных условий*/
 public:
  GrZadDemo()/*Инициализация массива граничных условий*/
  {
    min=1,max=1.6,
    grusl[0]=3,grusl[1]=0.5,grusl[2]=2.075678,
    grusl[3]=2,grusl[4]=0.7,grusl[5]=0.5118773;
  }
  double getmin() {return min;}/*Получение минимума*/
  double getmax() {return max;}/*Получение максимума*/
  double getgrusl(int posit)/*получение граничных условий по номеру*/
  {
    if(posit>6||posit<1)
      throw xmsg("Неверные параметры");
    return grusl[posit-1];
  }
  double getrange() {return 2;}/*Порядок уравнения*/
  /*Вектор значений коэффициентов в данной точке*/
  dvector getvalue(double x)
  {
    if(x<min||x>max)
      //возвращает значение функции Бесселя
      throw xmsg("Выход за предела диапазона");
    dvector pqr=3;
    pqr[0]=1.0/x, pqr[1]=1, pqr[2]=0;
```

```
    return pqr;
  }
  //вычисляем производную
  dvector getderive(double x, dvector y)
  {
    dvector f=2, pqr=getvalue(x);
    f[0]=y[1];
    f[1]=pqr[2]-pqr[0]*y[1]-pqr[1]*y[0];
    return f;
  }
};


/*Тестирование метода конечных разностей*/
void test1()
{
  GrZadDemo test;
  int  r=test.getrange();/*порядок системы дифференциальных уравнений*/
  int n=100; //число разбиений
  int m=n+r; //число уравнений в ортогонализируемой системе
  dvector g(r*(r+1));   //массив для граничных условий
  double min=test.getmin(), max=test.getmax();
  double h=(max-min)/n; //шаг
  // ЗАПОЛНИМ МАССИВ ГРАНИЧНЫХ УСЛОВИЙ
  for(int i=0;i<r*(r+1);i++)
    g[i]=test.getgrusl(i+1);
  dmatrix mtr(m,m),right(m,1);
  // сформируем матрицу СЛАУ
  // Заполним первую и последнюю строки основной матрицы СЛАУ
  mtr[0][0]=-g[1]/(2*h);
  mtr[0][2]=-mtr[0][0];
  mtr[0][1]=g[0];
  right[0][0]=g[2];
  // всего строк: m :от 0 до m-1, m-1 - последняя
  mtr[m-1][m-2]=g[4]/(2*h);
  mtr[m-1][m-4]=-g[4]/(2*h);
  mtr[m-1][m-3]=g[3];
  right[m-1][0]=g[5];
  //Далее матрицу заполняем со строки 1 по m-2 включительно
  for(long i=1;i<=m-2;i++)
  {
```

```
    dvector pqr=test.getvalue(min+(i-1)*h);
    mtr[i][i-1]=1/(h*h)-pqr[0]/(2*h);
    mtr[i][i]=pqr[1]-2/(h*h);
    mtr[i][i+1]=1/(h*h)+pqr[0]/(2*h);
    right[i][0]=pqr[2];
  }
  /*Для тестирования выведем полученную трёхдиагональную матрицу*/
  cout<<"3-diagonal matrix\n"<<mtr<<endl;
  dmatrix result=SLAE_Orto(mtr,right);/*решим СЛАУ*/
  cout<<"Solution\n";
  /*Решением являются значения функции Бесселя нулевого порядка*/
  for(long i=0;i<m-1;i++)
    cout<<"X="<<min+i*h<<" Y="<<result[i][0]<<endl;
}


double psi(int l,double a,double h,dvector g,double
min,GrZadDemo test,int n)
{//интегратор по числу разбиений с выводом результата
  dvector y=2;
  y[0]=a;
  y[1]=(g[2]-g[0]*a)/g[1];
  double x=min;
  if(l==1)
    cout<<x<<" "<<a<<" "<<y[1]<<endl;
  for(int m=0;m<n;m++)
  {
    dvector f=test.getderive(x,y);
    y+=h*f;
    x+=h;
    if(l==1)
      cout<<x<<" "<<y[0]<<" "<<y[1]<<endl;
  }
  if(l==0)
    return g[3]*y[0]+g[4]*y[1]-g[5];
}


/*Тестирование метода стрельбы*/
void test2()
{
  GrZadDemo test;
```

```
  int  r=test.getrange();/*порядок системы дифференциальных уравнений*/
  int n=100; //число разбиений
  dvector g(r*(r+1));   //массив для граничных условий
  double min=test.getmin(), max=test.getmax();
  double h=(max-min)/n; //шаг
  for(int i=0;i<r*(r+1);i++)
    g[i]=test.getgrusl(i+1);
  double  t=psi(0,max,h,g,min,test,n);/*три раза решаем задачу Коши*/
  double k=psi(0,min,h,g,min,test,n);
  psi(1,max-(max-min)/(t-k)*t,h,g,min,test,n);
}


void main()
{
  test1();//Метод конечных разностей
  getch();
  test2();//Метод стрельбы
  getch();
}
```