МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Інститут прикладної математики та фундаментальних наук
Кафедра прикладної математики

Звіт

про виконання лабораторної роботи №3
з курсу “Чисельні методи частина 2”
на тему:
«ЛІНІЙНІ БАГАТОКРОКОВІ МЕТОДИ ЧИСЕЛЬНОГО РОЗВ’ЯЗУВАННЯ ЗАДАЧІ КОШІ ДЛЯ ЗВИЧАЙНИХ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ»

Виконав:
студент гр. ПМ-41
Дудяк М.С.
Прийняв:
доцент
Пізюр Я. В.

Львів-2016

Варіант 6

Постановка задачі

Задано задачу Коші для системи звичайних диференціальних рівнянь:

$$\begin{cases} u_1' = -2000u_1 + 1000u_2 + 1 \\ u_2' = u_1 - u_2 \end{cases},$$

$$\begin{aligned} & t \in [0,4] \\ & u_1(0) = u_2(0) = 0 \\ & h_0 = 5 \cdot 10^{-3} \end{aligned}.$$

1. Використовуючи мову програмування Fortran, за заданим зразком написати програму для розв'язування даної системи методом диференціювання назад.

Аналіз задачі

Матриця Якобі правих частин цієї ситеми ЗДР має вигляд

$$J = \begin{pmatrix} -2000 & 1000 \\ 1 & -1 \end{pmatrix}$$

Оскільки власні значення матриці Якобі $\lambda_1 = -0.5, \lambda_2 = -2000$, то задача є жорсткою.

Теоретичні відомості

Явище жорсткості. Суть явища жорсткості полягає в тому, що розв’язок, який потрібно обчислити, змінюється повільно, однак існують швидкі згасаючі збурення. Наявність таких збурень перешкоджає знаходженню чисельного розв’язку, який повільно змінюється.

Для лінійних систем диференціальних рівнянь зі сталими коефіцієнтами подібні компоненти розв’язку, які сильно відрізняються, виникають, коли матриця системи містить сильно розкидані власні значення.

Формули диференціювання назад. Багатокрокові формули Адамса основані на чисельному інтегруванні, тобто інтеграл в (1) апроксимується деякою квадратурною формулою. Тепер розглянемо багатокрокові методи, які грунтуються на чисельному диференціюванні.

Припустимо, що відомі значення $y_{n-k+1}, y_{n-k+2}, \ldots, y_n$ розв’язку диференціального рівняння (1) §2 . Щоб вивести формулу для y_{n+1}, використаємо інтерполяційний многочлен $Q(t)$, який проходить через точки

$\{(x_j, y_j) \mid j = \overline{n-k+1, n+1}\}$. Як і многочлен (4) , його можна виразити через різниці назад , а саме

$$Q(t) = Q(t_n + s\tau) = \nabla^0 y_{n+1} + \frac{s-1}{1!}\nabla y_{n+1} + \frac{(s-1)s}{2!}\nabla^2 y_{n+1} + \\ + \ldots + \frac{(s-1)s\ldots(s+k-2)}{k!}\nabla^k y_{n+1}. \tag{7}$$

Визначимо тепер невідоме значення y_{n+1} так, щоб многочлен $Q(t)$ задовольняв диференціальне рівняння хоча б в одному вузлі сітки, тобто

$$Q'(t_{n+1-r}) = f(t_{n+1-r}, y_{n+1-r}).$$

Враховуючи , що $s = (t - t_n)/\tau$, продиференціюємо (7) по змінній t

$$Q'(t) = \frac{1}{\tau}\sum_{j=1}^{k}\frac{d}{ds}\left(\frac{(s-1)s\ldots(s+j-2)}{j!}\right)\nabla^j y_{n+1}.$$

Для $r = 1$ одержимо явні формули

$$\sum_{j=1}^{k}\delta_j \nabla^j y_{n+1} = \tau f_n ,$$

де

$$\delta_1 = 1, \quad \delta_j = \frac{d}{ds}\left[\frac{(s-1)s\ldots(s+j-2)}{j!}\right]_{s=0} = -\frac{1}{j(j-1)}, \quad j \geq 2 .$$

При $k = 1$ будемо мати явний метод Ейлера, а при $k = 2$ правило середньої точки

$$\frac{1}{2}y_{n+1} - \frac{1}{2}y_{n-1} = \tau f_n.$$

У випадку $k = 3$ формула має вигляд

$$\frac{1}{3}y_{n+1} + \frac{1}{2}y_n - y_{n-1} + \frac{1}{6}y_{n-2} = \tau f_n .$$

Однак вона нестійка, як і всі решта формул при $k > 3$ (див. п.4), а тому непридатна для розрахунків.

Кращі властивості мають формули, які одержуються з (7) при $r = 0$. Це неявні формули

$$\sum_{j=1}^{k}\delta_j^* \nabla^j y_{n+1} = \tau f_{n+1} \tag{8}$$

з коефіцієнтами

$$\delta_j^* = \frac{d}{ds}\left[\frac{(s-1)s\ldots(s+j-2)}{j!}\right]_{s=1},$$

які після диференціювання набувають вигляду

$$\delta_j^* = \frac{1}{j} \quad \text{при} \quad j \geq 1 .$$

Тому (8) зводиться до формули

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = \tau f_{n+1} .$$

Такі багатокрокові методи називають формулами диференціювання назад. Вони вперше були виведені Кертісом і Хіршфельдером.

Наведемо приклади цих формул, виразивши різниці назад через y_{n-i}

$$y_{n+1} - y_n = \tau f_{n+1}, k = 1,$$
$$\frac{3}{2} y_{n+1} - 2 y_n + \frac{1}{2} y_{n-1} = \tau f_{n+1}, k = 2,$$
$$\frac{11}{6} y_{n+1} - 3 y_n + \frac{3}{2} y_{n-1} - \frac{1}{3} y_{n-2} = \tau f_{n+1}, k = 3,$$
$$\frac{25}{12} y_{n+1} - 4 y_n + 3 y_{n-1} - \frac{4}{3} y_{n-2} + \frac{1}{4} y_{n-3} = \tau f_{n+1}, k = 4,$$
$$\frac{137}{60} y_{n+1} - 5 y_n + 5 y_{n-1} - \frac{10}{3} y_{n-2} + \frac{5}{4} y_{n-3} - \frac{1}{5} y_{n-4} = \tau f_{n+1}, k = 5.$$

Формули диференціювання назад мають вигляд:

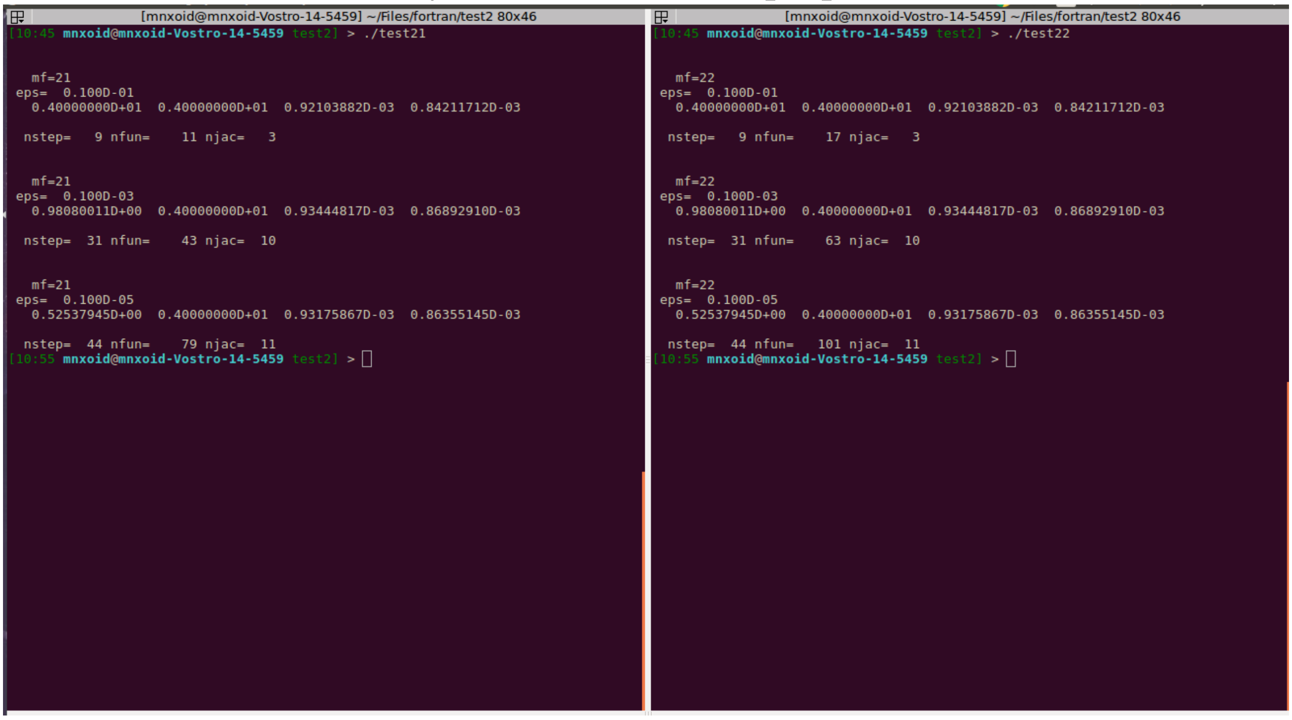
$$\sum_{j=0}^{k} \alpha_j y_{n-j+1} = \tau f_{n+1} .$$

Текст програми

```
   program test1
      implicit real*8(a-h,o-z)
      dimension y(10,13),ymax(10),error(10),pw(100),
     1          fsave(20),iwork(10)
      common/stcom1/t,h,hmin,hmax,eps,n,mf,kflag,jstart,maxord
      common/stcom2/hused,nqused
      common/stcom3/ml,mu
      common/stcom4/nstep,nfun,njac
      nydim=10
      eps=1.d-2
      kb=0
401   continue
      n=2
      t=0.0d0
      tend=4.d0
      y(1,1)=0.d0
      y(2,1)=0.d0
      h=5.0d-3
      hmax=tend
      hmin=1.d-15
      jstart=0
      mf=21
      maxord=5
      write(0,20) mf,eps
20    format(//3x,'mf=',i2/,' eps='d11.3)
      nstep=0
      nfun=0
      njac=0
      do 30 i=1,n
30    ymax(i)=1.d0
40    continue
      call stiff(y,ymax,error,pw,fsave,iwork,nydim)
      if(kflag.eq.0)go to 60
      write(0,50) kflag
50    format(/'  kflag=',i2/)
      stop
60    continue
      if(dabs(tend-t).le.1.d-15) go to 90
      if(tend-t-h) 80,40,40
80    e=tend-t
      s=e/h
      do 85 i=1,n
      do 85 j=1,jstart
85    y(i,1)=y(i,1)+y(i,j+1)*s**j
      t=t+e
      go to 60
90    continue
      write(0,556) h,t,(y(i,1),i=1,n)
556   format(1x,5d16.8)
```

```
      write(0,95) nstep,nfun,njac
95    format(/'  nstep=',i4,' nfun= ',i5,' njac=',i4)
      kb=kb+1
      if(kb.ge.3) go to 402
      eps=eps*1.d-2
      go to 401
402   continue
      stop
      end
      subroutine diffun (n,t,y,ydot)
      implicit real*8 (a-h,o-z)
      dimension y(1),ydot(1)
      ydot(1)=-2.d3*y(1)+1.d3*y(2)+1.0
      ydot(2)=y(1)-y(2)
      return
      end
      subroutine pederv(n,t,y,pw,nydim)
      implicit real*8 (a-h,o-z)
      dimension y(1),pw(1)
      pw(1)=-2.0d3
      pw(2)=1.d0
      pw(nydim+1)=1.d3
      pw(nydim+2)=-1.d0
      return
      end
```

Результат виконання програми



```
[mnxoid@mnxoid-Vostro-14-5459] ~/Files/fortran/test2 80x46
[10:45 mnxoid@mnxoid-Vostro-14-5459 test2] > ./test21


  mf=21
eps=  0.100D-01
  0.40000000D+01  0.40000000D+01  0.92103882D-03  0.84211712D-03

 nstep=   9 nfun=    11 njac=   3


  mf=21
eps=  0.100D-03
  0.98080011D+00  0.40000000D+01  0.93444817D-03  0.86892910D-03

 nstep=  31 nfun=    43 njac=  10


  mf=21
eps=  0.100D-05
  0.52537945D+00  0.40000000D+01  0.93175867D-03  0.86355145D-03

 nstep=  44 nfun=    79 njac=  11
[10:55 mnxoid@mnxoid-Vostro-14-5459 test2] >
```

```
[mnxoid@mnxoid-Vostro-14-5459] ~/Files/fortran/test2 80x46
[10:45 mnxoid@mnxoid-Vostro-14-5459 test2] > ./test22


  mf=22
eps=  0.100D-01
  0.40000000D+01  0.40000000D+01  0.92103882D-03  0.84211712D-03

 nstep=   9 nfun=    17 njac=   3


  mf=22
eps=  0.100D-03
  0.98080011D+00  0.40000000D+01  0.93444817D-03  0.86892910D-03

 nstep=  31 nfun=    63 njac=  10


  mf=22
eps=  0.100D-05
  0.52537945D+00  0.40000000D+01  0.93175867D-03  0.86355145D-03

 nstep=  44 nfun=   101 njac=  11
[10:55 mnxoid@mnxoid-Vostro-14-5459 test2] >
```

Аналіз результатів

При збільшенні точності зменшується похибка(значення y_i наближаються до точних). Також помітно зростання кількості викликів процедур diffun і pederv, та кількості кроків.

Висновок

В цій лабораторній роботі я навчився: з допомогою мови програмування Fortran і написаної на ній процедури STIFF розв’язувати жорсткі системи ЗДР методом диференціювання назад з заданою точністю.