Российский Государственный Технологический Университет им. К.Э .Циолковского. (МАТИ)

Кафедра физики РГТУ-МАТИ

Бураго Н.Г. (профессор кафедры физики РГТУ-МАТИ)

Курс лекций “метод конечных элементов” для студентов 5-го года обучения (9-й семестр). 14х4=56 часов.

1. Аннотации к лекциям

1.1.Введение

Метод конечных элементов является мощным инструментом численного решения большинства математических задач, моделирующих работу авиационных конструкций в набегающих потоках. Поэтому большинство пакетов прикладных программ, предназначенных для решения встречающихся при проектировании и эксплуатации авиационной техники задач механики деформируемого твердого тела и механики жидкости и газа, основаны на различных вариантах метода конечных элементов. Знание основ и принципов построения конечно-элементных алгоритмов является необходимой составляющей знаний авиационного инженера.

1.2.Перечень читаемых разделов.

1.2.1. Постановка задач континуальной механики. Законы сохранения. Определяющие соотношения. Кинематические соотношения. Граничные и начальные условия. Понятия конвекции, диффузии и источников/стоков. Системы координат: лагранжевы, эйлеровы и произвольно-подвижные.

1.2.2. Дифференциальная, интегральная и вариационная формулировки начально-краевых задач континуальной механики. Вариация. Основная лемма вариационного исчисления. Уравнения Эйлера. Основные переменные и потоки. Главные и естественные граничные условия.

1.2.3. Методы дискретизации начально-краевых задач континуальной механики. Общая схема проекционных методов. Способы выбора и построения аппроксимационного и проекционного базисов. Сеточные и бессеточные методы.

1.2.4. Вопросы существования и единственности решений. Критерий Адамара корректности начально-краевых задач и его дискретная интерпретация. Квазилинеаризация исходных уравнений и граничных условий. Теорема о неявной функции. Методы продолжения решения по параметру Ньютона и Давиденко. Ветвление решений. Безматричные методы решения вспомогательных спектральных задач.

1.2.4. Определение сеток. Типы сеток. Способы задания сеток. Понятие о методах построения сеток. Триангуляции Делоне и Вороного. Информация о свободно доступных генераторах сеток. Понятие об адаптивных подвижных и встраивающихся сетках.

1.2.5. Конечно-элементная аппроксимация решения.. Одномерные, двумерные и трехмерные симплекс - элементы. Линейные функции формы и кусочно-линейные базисные функции. Построение элементов повышенного порядка точности.

1.2.5. Формирование дискретных уравнений МКЭ для исходных начально-краевых задач: алгоритмы вычисления невязок для нелинейных задач и определение матриц масс и жесткости, а также вектора правой части для линеаризованных задач. Квадратуры Гаусса для многомерных задач. Метод численного бессеточного интегрирования. Примеры сравнения МКЭ с методами конечных объемов и конечных разностей. Сравнение сеточных и бессеточных методов.

1.2.6. Прямые и итерационные методы решения алгебраических уравнений. Методы пошаговой линеаризации и метод установления для нелинейных задач. Хранение и оптимизация структуры матриц жесткости. Методы прогонки и сопряженных градиентов для линеаризованных задач. Предобусловливание.

1.2.7. Явные и неявные схемы интегрирования по времени. Определения аппроксимации и устойчивости. Теорема сходимости приближенных решений Лакса. Определение монотонности решения и схемы.

1.2.8. Консервативность. Разрывные решения. Соотношения на скачках. Теорема сходимости Лакса-Вендроффа для разрывных решений. Обоснование консервативности МКЭ.

1.2.9. Методы исследования устойчивости интегрирования по времени эволюционных задач для модельного уравнения конвекции-диффузии. Методы дискретных и гармонических возмущений, спектральный метод и метод дифференциальных приближений.

1.2.10. Анализ классических схем: схемы ВВЦП, схемы Лакса, схемы с разностями против потока, схемы Дюфорта-Франкела, схемы Кранка-Николсона, неявной схемы Эйлера, схем Лакса-Вендроффа и Мак-Кормака. Определение физической, схемной, искусственной и эффективной вязкостей. Коррекция эффективной вязкости по методу Самарского. Метод экспоненциальной подгонки эффективной вязкости.

1.2.11. Расчет течений несжимаемой вязкой жидкости: варианты метода искусственной сжимаемости,. метод коррекции давления, методы для уравнений в переменных вихрь-функция тока.

1.2.12. Расчет сжимаемых течений. Семейство схем Годунова и их конечно-элементная интерпретация. Метод уравновешенной вязкости. Улавливание скачков. Теорема Лакса-Вендроффа и важность свойства консервативности. Методы адаптивных и наложенных сеток.

1.2.13. Расчет упругопластических течений. Методы упругих решений и переменных параметров упругости. Упруговязкопластические течения. Методы расчета контактных взаимодействий. Особенности реализации моделей континуального и усталостного разрушения.

1.2.14. Безматричная реализация метода конечных элементов. Рекомендации по программированию конечно-элементных алгоритмов. Обзор распространенных пакетов программ для задач континуальной механики.

1.3.Рекомендуемая литература (2007-2011гг издания)

1. **Бураго Н.Г. Вычислительная механика. Электронный учебник. 2010. 250 с. Доступен для скачивания с сайта автора (ipmnet.ru/~burago)**
2. **Кукуджанов В.Н. Вычислительная механика сплошных сред. М.: ФИЗМАТЛИТ, 2008. 320 с.**

2. Материалы к практическим занятиям

МКЭ-дискретизация одномерной задачи о распространении плоской упругой волны в слое конечной толщины.

МКЭ дискретизация одномерной задачи нестационарной теплопроводности в слое конечной толщины.

Примеры применения метода сопряженных градиентов к решению линейных алгебраических уравнений: а) с положительной симметричной матрицей, б) с положительной несимметричной матрицей, в) с знаконеопределенной невырожденной несимметричной матрицей, г) с плохо обусловленной матрицей, д) с вырожденной матрицей.

Применение метода установления к решению одномерной задачи стационарной теплопроводности.

Применение метода штрафа к учету граничных условий в одномерных задачах теплопроводности и распространения упругих волн.

Применение метода штрафа к записи контактных условий в одномерной задаче взаимодействия двух упругих слоев.

Написание Фортран-программ для решения одномерных задач теплопроводности и распространения волн.

Сравнение записи вариантов алгоритма на языках Фортран, С. Паскаль.

3. Экзаменационные билеты

Номера и содержание билетов соответствуют нумерации и содержанию лекций.

4. Тестовые задания (необязательные)

Составить программы на любом языке программирования (Фортран, С, Паскаль, …) или с использованием любого пакета математических вычислений (Maple, Mathematika, Matlab, Octave, …) для решения задач теплопроводности или распространения упругих волн в пространственно двумерной постановке для прямоугольной области.

Примечание: Это задание предложено в самом начале курса лекций как разгонная тренировка выполнения вычислений по темам дипломных работ, а также как первая цель для желающих попробовать свои силы в научной работе..

ПРИЛОЖЕНИЕ.
ТЕКСТЫ УЧЕБНЫХ ПРОГРАММ.

1. Метод сопряженных градиентов. Случай положительной симметричной матрицы.

```
! Учебная программа. Метод сопряженных градиентов
! случай положительной симметричной матрицы
!     program testcj
      parameter (n1=2)
      real x(n1),r(n1),p(n1),z(n1),
     +rmd(n1),ap(n1)

      eps2=1e-18
      ii=0
      DO I=1,N1
       X(I)=0.0
       p(i)=0.0
      ENDDO
```

```
      s=1e30
      call gr(ii,x,r,n,rmd)
    1 ii=ii+1
      s0=s
      s=0.0
      DO I=1,N1
       z(i)=rmd(i)*r(i)
       s=s+r(i)*z(i)
      enddo
      beta=s/s0
      do i=1,n1
       p(i)=z(i)+beta*p(i)
      enddo
      call gr(ii,p,ap,n,rmd)
      pap=0.0
      pp=0.0
      DO i=1,N1
       pp=pp+p(i)*p(i)
       pap=pap+p(i)*ap(i)
      enddo
      if(pp.lt.eps2)goto 4000
      if(abs(pap).lt.eps2)then
       write(*,*)' error: ii,pap=',ii,pap
       read(*,*)
       goto 555
      endif
      alfa=s/pap
      do i=1,n1
       x(i)=x(i)-alfa*p(i)
       r(i)=r(i)-alfa*ap(i)
      enddo
      rr=0.0
      do i=1,n1
       rr=rr+r(i)**2
      enddo
      write(*,*)' ii,rr,x=',ii,rr,x
      read(*,*)
      if(rr.lt.eps2)goto 4000
      goto 1
 4000 write(*,*)' ii=',ii
      write(*,*)' x=',x
 555  END

      subroutine gr(ii,x,r,n,rmd)
      real x(n),r(n),rmd(n)
c      2x+y=4 symmetric x=y=1
c      x+2y=5
      r(1)=-2.0*x(1)-x(2)
      r(2)=-2.0*x(2)-x(1)
      if(ii.eq.0)then
       r(1)=r(1)+4.0
       r(2)=r(2)+5.0
```

```
      endif
c rmd
      if(ii.eq.0)then
       rmd(1)=1.0/2.0
       rmd(2)=1.0/2.0
      endif
      return
      end
```

2. Метод сопряженных градиентов. Случай положительной несимметричной матрицы.

```
! Учебная программа. Метод сопряженных градиентов
! случай положительной несимметричной матрицы N=2.
!     program testcj
      parameter (n1=2)
      real x(n1),r(n1),p(n1),rmd(n1),c(n1),c1(n1)
      eps2=1e-18
      ii=0
      DO I=1,N1
       X(I)=0.0
      ENDDO
      call gr(0,x,r,n1,rmd) ! r=r0
      write(*,*)' r,rmd=',r,rmd
      call grt(r,p,n1,rmd) ! p=R0
      rr=0.0
      do i=1,n1
       r(i)=p(i) ! r=R0
       rr=rr+r(i)*r(i)
      enddo
      write(*,*)' r=',r
      write(*,*)' ii,rr,x=',ii,rr,x
      if(rr.lt.eps2)goto 4000
    1 ii=ii+1
      call gr(1,p,c,n1,rmd)
      call grt(c,c1,n1,rmd)
      write(*,*)' c,c1=',c,c1
      cc=0.0
      DO I=1,N1
       cc=cc+c(i)*c(i)
      enddo
      a=rr/cc
      write(*,*)' cc,a=',cc,a
      do i=1,n1
       x(i)=x(i)-a*p(i)
       r(i)=r(i)-a*c1(i)
      enddo
      write(*,*)' x,r=',x,r
      rr0=rr
      rr=0.0
      DO i=1,N1
       rr=rr+r(i)*r(i)
      enddo
```

```
      if(rr.lt.eps2)goto 4000
      b=rr/rr0
      do i=1,n1
       p(i)=r(i)+b*p(i)
      enddo
      write(*,*)' ii,rr,x=',ii,rr,x
      read(*,*)
      goto 1
 4000 write(*,*)' ii,rr,x=',ii,rr,x
 555  END



      subroutine gr(ii,x,r,n1,rmd)
      real x(n1),r(n1),rmd(n1)

c      2x-y=1 no symmetry y=1, x=1
c      x+2y=3
      r(1)=+(2.0*x(1)-x(2))
      r(2)=+(2.0*x(2)+x(1))
      if(ii.eq.0)then
       r(1)=r(1)-1.0
       r(2)=r(2)-3.0
      endif
c rmd
      if(ii.eq.0)then
       do i=1,n1
        rmd(i)=1.0/2.0
       enddo
      endif
c      if(ii.gt.0)then
       do i=1,n1
        r(i)=r(i)*rmd(i)
       enddo
c      endif
      end



     subroutine grt(x,r,n1,rmd)
      real x(1),r(1),rmd(1)
c conjugated
c     r1=2y-x no symmetry y=1, x=1
c     r2=y+2x
      r(1)=2.0*x(2)-x(1)
      r(2)=2.0*x(1)+x(2)
      do i=1,n1
       r(i)=r(i)*rmd(i)
      enddo
      end
```

3. Метод сопряженных градиентов. Случай положительной несимметричной матрицы N=3.

```
! Учебная программа. Метод сопряженных градиентов
! случай положительной несимметричной матрицы
      parameter (n1=3)
      real x(n1),gr(n1),s(n1),rmd(n1),as(n1),gr0(n1)
      external gr1,grt
      DO I=1,N1
       X(I)=0.0
      ENDDO
      call cjns(n1,x,gr,s,rmd,as,gr0,gr1,grt)
      end
      subroutine cjns(n1,x,gr,s,rmd,as,gr0,gr1,grt)
      real x(n1),gr(n1),s(n1),rmd(n1),as(n1),gr0(n1)
      external gr1,grt
      eps2=1e-8*n1
      ii=0
      call gr1(0,x,gr,n1,rmd) ! r=r0
      call grt(gr,s,n1,rmd) ! p=R0
      ggh=0.0
      do i=1,n1
       gr(i)=s(i) ! gr=R0
       ggh=ggh+gr(i)*gr(i)
      enddo
      if(ggh.lt.eps2)goto 4000
    1 ii=ii+1
      call gr1(1,s,as,n1,rmd)
      call grt(as,gr0,n1,rmd)
      asas=0.0
      DO I=1,N1
       asas=asas+as(i)*as(i)
      enddo
      a=ggh/asas
      dxdx=0.0
      do i=1,n1
       dx=-a*s(i)
       dxdx=dxdx+dx**2
       x(i)=x(i)+dx
       gr(i)=gr(i)-a*gr0(i)
      enddo
      gg=ggh
      ggh=0.0
      DO i=1,N1
       ggh=ggh+gr(i)*gr(i)
      enddo
      if(ggh.lt.eps2.or.dxdx.lt.eps2)goto 4000
      b=ggh/gg
      do i=1,n1
       s(i)=gr(i)+b*s(i)
      enddo
      goto 1
```

```
 4000 write(*,*)' ii,ggh=',ii,ggh
      write(*,*)' x=',x
 555  END


      subroutine gr1(ii,x,gr,n1,rmd)
      real x(n1),gr(n1),rmd(n1)
c      2x-y=1 no symmetry y=1, x=1 , z=0
c      x+2y=3
c      x+y+2z=2
      gr(1)=2.0*x(1)-x(2)
      gr(2)=2.0*x(2)+x(1)
      gr(3)=2.0*x(3)+x(1)+x(2)
      if(ii.eq.0)then
       gr(1)=gr(1)-1.0
       gr(2)=gr(2)-3.0
       gr(3)=gr(3)-2.0
      endif
c rmd
      if(ii.eq.0)then
       do i=1,n1
        rmd(i)=1.0/2.0
       enddo
      endif
      do i=1,n1
       gr(i)=gr(i)*rmd(i)
      enddo
      end

      subroutine grt(x,gr,n1,rmd) ! transposed
      real x(1),gr(1),rmd(1)
c conjugated
c     r1=2y-x+z no symmetry y=1, x=1
c     r2=y+2x+z
c     r3=2z
      gr(1)=2.0*x(2)-x(1)+x(3)
      gr(2)=2.0*x(1)+x(2)+x(3)
      gr(3)=2.0*x(3)
      do i=1,n1
       gr(i)=gr(i)*rmd(i)
      enddo
      end
```

4. Метод сопряженных градиентов. Случай нейтральной несимметричной матрицы.

```
! Учебная программа. Метод сопряженных градиентов
! случай нейтральной несимметричной матрицы
c     program testcj ! neutral nonsymmetric
      parameter (n1=2)
      real x(n1),r(n1),p(n1),
     +rmd(n1),ap(n1),ar(n1)
      n=n1
```

```
      eps2=1e-18
      ii=0
      DO I=1,N1
       X(I)=0.0
      ENDDO
c      s=1e30
      call gr(ii,x,r,n,rmd)
      rr=0
      do i=1,n1
       p(i)=r(i)
       rr=rr+r(i)*r(i)
      enddo
      if(rr.lt.eps2)goto 4000
    1 ii=ii+1
      call gr(ii,p,ap,n,rmd)
      rap=0.0
      apap=0.0
      DO I=1,N1
       apap=apap+ap(i)*ap(i)
       rap=rap+r(i)*ap(i)
      enddo
      write(*,*)' rap,apap,eps2=',rap,apap,eps2
      if(abs(apap).lt.eps2)goto 4000
      alpha=rap/apap
      rr=0.0
      do i=1,n1
       x(i)=x(i)-alpha*p(i)
       r(i)=r(i)-alpha*ap(i)
       rr=rr+r(i)*r(i)
      enddo
      if(rr.lt.eps2)goto 4000
      call gr(ii,r,ar,n,rmd)
      arap=0.0
      DO i=1,N1
       arap=arap+ar(i)*ap(i)
      enddo
      if(abs(arap).lt.eps2)then
       write(*,*)' error: ii,arap=',ii,arap
       read(*,*)
       goto 555
      endif
      beta=-arap/apap
      do i=1,n1
       p(i)=beta*p(i)+r(i)
      enddo
      write(*,*)' ii,rr,x=',ii,rr,x
      read(*,*)
      goto 1
 4000 write(*,*)' ii=',ii
      write(*,*)' x=',x
  555  END
```

```
      subroutine gr(ii,x,r,n,rmd)
      real x(n),r(n),rmd(n)
c      -2x+2y=2 neutral nonsymmetric x=1 y=2
c       x+2y=5
      r(1)=-2.0*x(1)+2.0*x(2)
      r(2)=2.0*x(2)+x(1)
      if(ii.eq.0)then
       r(1)=r(1)-2.0
       r(2)=r(2)-5.0
      endif
c rmd
      if(ii.eq.0)then
       rmd(1)=1.0/2.0
       rmd(2)=1.0/2.0
      endif
      do i=1,n
       r(i)=r(i)*rmd(i)
      enddo
      return
      end
```

5. Метод сопряженных градиентов. Случай нейтральной несимметричной плохо обусловленной матрицы.

```
!     program testcj ! bad conditioned, newtral symmetric
      parameter (n1=2)
      real x(n1),r(n1),p(n1),
     +rmd(n1),ap(n1),ar(n1)
      n=n1
      eps2=1e-18
      ii=0
      DO I=1,N1
       X(I)=0.0
      ENDDO
      call gr(ii,x,r,n,rmd)
      rr=0
      do i=1,n1
       p(i)=r(i)
       rr=rr+r(i)*r(i)
      enddo
      if(rr.lt.eps2)goto 4000
    1 ii=ii+1
      call gr(ii,p,ap,n,rmd)
      rap=0.0
      apap=0.0
      DO I=1,N1
       apap=apap+ap(i)*ap(i)
       rap=rap+r(i)*ap(i)
      enddo
      write(*,*)' rap,apap,eps2=',rap,apap,eps2
      if(abs(apap).lt.eps2)goto 4000
      alpha=rap/apap
```

```
      rr=0.0
      do i=1,n1
       x(i)=x(i)-alpha*p(i)
       r(i)=r(i)-alpha*ap(i)
       rr=rr+r(i)*r(i)
      enddo
      if(rr.lt.eps2)goto 4000
      call gr(ii,r,ar,n,rmd)
      arap=0.0
      DO i=1,N1
       arap=arap+ar(i)*ap(i)
      enddo
c      if(abs(arap).lt.eps2)then
c       write(*,*)' error: ii,arap=',ii,arap
c       read(*,*)
c       goto 555
c      endif
      beta=-arap/apap
      do i=1,n1
       p(i)=beta*p(i)+r(i)
      enddo
      write(*,*)' ii,rr,x=',ii,rr,x
      read(*,*)
      goto 1
 4000 write(*,*)' ii=',ii
      write(*,*)' x=',x
 555  END

      subroutine gr(ii,x,r,n,rmd)
      real x(n),r(n),rmd(n)
c matrix neutral, nonsymmetric: eigen values l1<0 l2>0
      r(1)=x(1)+0.99*x(2)
      r(2)=0.98*x(1)+0.98*x(2)
      if(ii.eq.0)then
       r(1)=r(1)-2.01 !1.99
       r(2)=r(2)-1.96 !1.96
      endif ! solution x1=3 x2=-1 !x1=x2=1
c rmd
      if(ii.eq.0)then
       rmd(1)=1.0/(1.0+1e-18)
       rmd(2)=1.0/(0.98+1e-18)
      endif
      do i=1,n
       r(i)=r(i)*rmd(i)
      enddo
      return
      end
```

6. Метод сопряженных градиентов. Случай нейтральной несимметричной вырожденной матрицы.

```
c     program testcj ! neutral nonsymmetric degenerate
      parameter (n1=3)
      real x(n1),r(n1),p(n1),
     +rmd(n1),ap(n1),ar(n1)
      n=n1
      eps2=1e-18
      ii=0
      DO I=1,N1
       X(I)=0.0
      ENDDO
c      s=1e30
      call gr(ii,x,r,n,rmd)
      rr=0
      do i=1,n1
       p(i)=r(i)
       rr=rr+r(i)*r(i)
      enddo
      if(rr.lt.eps2)goto 4000
    1 ii=ii+1
      call gr(ii,p,ap,n,rmd)
      rap=0.0
      apap=0.0
      DO I=1,N1
       apap=apap+ap(i)*ap(i)
       rap=rap+r(i)*ap(i)
      enddo
      write(*,*)' rap,apap,eps2=',rap,apap,eps2
      if(abs(apap).lt.eps2)goto 4000
      alpha=rap/apap
      rr=0.0
      do i=1,n1
       x(i)=x(i)-alpha*p(i)
       r(i)=r(i)-alpha*ap(i)
       rr=rr+r(i)*r(i)
      enddo
      if(rr.lt.eps2)goto 4000
      call gr(ii,r,ar,n,rmd)
      arap=0.0
      DO i=1,N1
       arap=arap+ar(i)*ap(i)
      enddo
      if(abs(arap).lt.eps2)then
       write(*,*)' incorrect problem'
       write(*,*)' error: ii,arap=',ii,arap
       read(*,*)
       goto 555
      endif
      beta=-arap/apap
      do i=1,n1
```

```
       p(i)=beta*p(i)+r(i)
      enddo
      write(*,*)' ii,rr,x=',ii,rr,x
      read(*,*)
      goto 1
 4000 write(*,*)' ii=',ii
      write(*,*)' x=',x
 555  END

      subroutine gr(ii,x,r,n,rmd)
      real x(n),r(n),rmd(n)
c      x1-x2=0
c      -x1+2*x2-x3=0  incorrect nonsymmetric x=1 y=2
c      -x2+x3=1       (det(a)=0)
      r(1)= x(1)-x(2)          !+x(1)*.1 !Regularization fails
      r(2)=-x(1)+2.0*x(2)-x(3) !+x(2)*.1 !and gives wrong results
      r(3)=         -x(2)+x(3) !+x(3)*.1
      if(ii.eq.0)then
       r(3)=r(3)-1.0
      endif
c rmd
      if(ii.eq.0)then
       rmd(1)=1.0
       rmd(2)=1.0/2.0
       rmd(3)=1.0
      endif
      do i=1,n
       r(i)=r(i)*rmd(i)
      enddo
      return
      end
```

7. Распространение упругих волн

```
! Учебная программа: распространение упругой волны
! описания переменных
      integer jx0,ju,jsx,jscp,jap,jee,jro,jdf,jmm
      real a(16000),x(2),f(2)
      character str*80
      na=16000 ! размер массива a, в котором расположены все массивы данных
      n1=101 ! число узлов
! метки расположения начала массивов данных внутри массива a
      jx0=0 ! начало массива x-координат узлов при t=0
      jd=jx0+n1 ! перемещения узлов
      ju=jd+n1 ! скорость узлов
      jsx=ju+n1 ! девиатор напряжения
      jscp=jsx+n1 ! среднее напряжение
      jap=jscp+n1 ! пластическая работа
      jee=jap+n1   ! внутренняя энергия
      jro=jee+n1   ! плотность
      jdf=jro+n1   ! ускорения
```

```
      jmm=jdf+n1 ! обратные приузловые объемы
      jrb=jmm+n1 ! рабочий массив 1
      jrb2=jrb+n1  ! рабочий массив 2
      maxa=jrb2+n1 ! номер последнего используемого элемента в массиве a
      if(maxa.gt.na)then ! проверка на переполнение массива a
       write(*,*)' массив a переполнен: maxa,na=',maxa,na
       read(*,*)
       goto 555 ! переход к концу программы
      endif
! инициализация переменных и задание входных данных
      hx0=10.0/(n1-1) ! шаг начальной равномерной сетки
      e1=1.0  ! (модуль Юнга)/(предел текучести)
      rnu=0.0 ! коэффициент Пуассона
      a6=e1/(1.0-2.0*rnu) ! (модуль всестороннего сжатия)*3/ (предел текучести)
      a5=e1/(1.0+rnu)  ! (модуль сдвига) *2/ (предел текучести)
      c0=1.0 ! безразмерная скорость звука
      ro0=(a6+2.0*a5)/3.0/c0**2 !3lam+2mu=a6;lam+2mu=(a6-2mu)/3+2mu=(a6+4mu)/3
!                               ro0-начальная безразмерная плотность
      demax=0.001 ! максимально допустимое приращение деформации на временном
шаге
      sk=10000.0*sqrt(2.0/3.0) ! значение sqrt(sij*sij) на круге текучести
      do i=1,n1 ! начальное состояние массивов данных,  цикл по узлам
       a(jx0+i)=(i-1)*hx0 ! начальные координаты
       a(jd+i)=0.0 ! начальные перемещения
       a(ju+i)=-0.5 ! начальная скорость стержня
       a(jsx+i)=0.0 ! начальный девиатор
       a(jscp+i)=0.0 ! начальное среднее напряжение
       a(jap+i)=0.0 ! начальная пластическая работа
       a(jee+i)=0.0 ! начальная внутренняя энергия
       a(jro+i)=ro0 ! начальная плотность
      enddo
      is=0 ! начальное значение номера временного слоя
      nsp=1 ! частота вывода на печать (через каждые nsp шагов)
      isp=is+nsp ! номер шага для печати
      t=0.0 ! начальное значение времени
      cur=0.25 ! коэффициент запаса в условии Куранта
! блок решения
  100 ht=1e18 !! выбор шага по времени: начальное значение шага по времени
      do i=1,n1
       a(jmm+i)=0.0 ! обнуляем массив обратных приузловых объемов
      enddo
      do i=2,n1 ! цикл по ячейкам (i-1,i)
       hx0=a(jx0+i)-a(jx0+i-1) ! начальный объем ячейки
       hx=hx0+a(jd+i)-a(jd+i-1) ! текущий объем ячейки
       if(hx.le.1e-5)then ! проверка невырожденности сетки
        write(*,*)' вырожденная ячейка: t, i, hx=',t,i,hx
        read(*,*)
        goto 555 ! если сетка вырождена, то прекратить вычисление
       endif
       rm=hx0/2.0 ! доля объема ячейки, отнесенного к узлу
       a(jmm+i)=a(jmm+i)+rm ! разноска долей объема ячейки по узлам
       a(jmm+i-1)=a(jmm+i-1)+rm !
```

```
      de=(a(ju+i)-a(ju+i-1))/hx ! скорость деформации
      ht=amin1(ht,1.0/amax1(de,1e-5),hx/c0*cur) ! коррекция шага по времени
     enddo  ! i
     do i=1,n1 ! определение обратных объемов
      a(jmm+i)=1.0/a(jmm+i)
     enddo
c##############################################
     do i=1,n1
      a(jdf+i)=0.0 ! массив ускорений
     enddo
     do i=1,n1 ! расчет шага по времени, цикл по узлам
      i1=min(i+1,n1) ! сосед узла i справа
      i2=max(i-1,1) ! сосед узла i слева
      hx0=a(jx0+i1)-a(jx0+i2) ! размер шага при t=0
      hx=hx0+a(jd+i1)-a(jd+i2) ! размер шага при t>0
      if(hx.le.1e-5)then ! проверка невырожденности сетки
       write(*,*)' 2:i,hx=',i,hx
       read(*,*)
       goto 555
      endif
      dex0=(a(ju+i1)-a(ju+i2))/hx ! скорость деформации xx
      dey0=-dex0*rnu
      dez0=-dex0*rnu
      decp=(dex0+dey0+dez0)/3.0 ! объемная составляющая скорости деформации
      dex=dex0-decp ! девиаторная составляющая скорости деформации по x
      dey=dey0-decp ! девиаторная составляющая скорости деформации по y
      dez=dez0-decp ! девиаторная составляющая скорости деформации по z
      a5ht=a5*ht ! вспомогательная величина
      a6ht=a6*ht ! вспомогательная величина
      dsx=a5ht*dex ! приращение девиатора напряжений по x
      dsy=a5ht*dey ! приращение девиатора напряжений по y
      dsz=a5ht*dez ! приращение девиатора напряжений по z
      dscp=a6ht*decp*hx0/hx ! приращение среднего напряжения
      sx0=a(jsx+i)  ! старый девиатор напряжений по x
      sx=sx0+dsx  ! новый девиатор напряжений по x
      sy=-sx/2.0 ! новый девиатор напряжений по y
      sz=sy ! новый девиатор напряжений по z
      scp0=a(jscp+i) !a6*((a(jd+i1)-a(jd+i2))/hx)/3.0 ! старое среднее напряжение
      scp=scp0+dscp ! новое среднее напряжение
      ap=a(jap+i) ! старая пластическая работа
      ee=a(jee+i) ! старая внутренняя энергия
      ro=a(jro+i) !ro0*hx0/hx ! старая плотность
      dro=-ro*decp*3.0*ht ! приращение плотности
      ss=sx**2+sy**2+sz**2+1e-12 ! новый второй инвариант напряжений
      r1=sk/amax1(sk,sqrt(ss)) ! коэффициент корректировки напряжений
      sx=sx*r1 ! новые корректированные по Уилкинсу напряжения по x
      sy=sy*r1 ! по y
      sz=sy  ! по z
      dap=(1.0-r1)*r1*ss/a5 ! приращение пластической работы
      ap=ap+dap ! новая пластическая работа
      dee=dap+scp*decp*3.0*ht ! приращение внутренней энергии
c       px=sx+scp-1.0*(dsx+dscp)
```

```
      px=a(jsx+i)+a(jscp+i) ! старое напряжение по x
      i1=i+1
      if(i.eq.n1)i1=n1 ! номер правого узла
      i2=i-1
      if(i.eq.1)i2=i ! номер левого узла
      sss=1.0/a(jmm+i) ! приузловой объем
      a(jdf+i1)=a(jdf+i1)+px/hx*sss ! вклады в ускорение от дивергенции напряжений
      a(jdf+i2)=a(jdf+i2)-px/hx*sss  ! вклады в ускорение от дивергенции напряжений
      a(jsx+i)=sx ! запоминание новых значений девиатора напряжений
      a(jscp+i)=scp ! среднего напряжения
c       a(jro+i)=ro+dro ! плотности
      a(jee+i)=ee+dee ! внутренней энергии
      a(jap+i)=ap ! пластической работы
     enddo ! i
c     call sgl(a(jdf+1),n1,a(jrb2+1))
     do i=1,n1
      a(ju+i)=a(ju+i)-a(jdf+i)*a(jmm+i)*ht/a(jro+i)  ! новые скорости (по ускорениям)
     enddo
     a(ju+1)=0.0 ! граничное условие: на стенке скорость равна нулю
     call sgl(a(ju+1),n1,a(jrb2+1)) ! монотонизация скорости
     a(ju+1)=0.0 ! восстановление граничного значения скорости на стенке
     call sgl(a(ju+1),n1,a(jrb2+1)) ! монотонизация скорости
     a(ju+1)=0.0 ! восстановление граничного значения скорости на стенке
c     call sgl(a(ju+1),n1,a(jrb2+1)) ! монотонизация скорости
c     a(ju+1)=0.0 ! восстановление граничного значения скорости на стенке
c     call sgl(a(ju+1),n1,a(jrb2+1)) ! монотонизация скорости
c     a(ju+1)=0.0 ! восстановление граничного значения скорости на стенке
     do i=1,n1
c       a(jd+i)=a(jd+i)+a(ju+i)*ht ! новые перемещения
     enddo
c     call sgl(a(jro+1),n1,a(jrb2+1)) ! монотонизация плотности

     call sgl(a(jscp+1),n1,a(jrb2+1)) ! монотонизация среднего напряжения
     call sgl(a(jsx+1),n1,a(jrb2+1)) ! монотонизация девиатора напряжений
     call sgl(a(jscp+1),n1,a(jrb2+1)) ! монотонизация среднего напряжения
     call sgl(a(jsx+1),n1,a(jrb2+1)) ! монотонизация девиатора напряжений
c     call sgl(a(jscp+1),n1,a(jrb2+1)) ! монотонизация среднего напряжения
c     call sgl(a(jsx+1),n1,a(jrb2+1)) ! монотонизация девиатора напряжений
c     call sgl(a(jscp+1),n1,a(jrb2+1)) ! монотонизация среднего напряжения
c     call sgl(a(jsx+1),n1,a(jrb2+1)) ! монотонизация девиатора напряжений

c     call sgl(a(jap+1),n1,a(jrb2+1)) ! монотонизация пластической работы
c     call sgl(a(jee+1),n1,a(jrb2+1)) ! монотонизация внутренней энергии
     is=is+1 ! счетчик шагов по времени
     t=t+ht  ! значение времени

     if(is.eq.isp)then ! блок печати
      isp=isp+nsp
       write(*,*)' is,t=',is,t
       write(*,*)' x0,d,u,sx,scp,ap,ee,ro'
       do i=1,n1,10
        write(*,'(f9.4,7f10.4)')
```

```
     +   a(jx0+i),a(jd+i),a(ju+i),a(jsx+i),a(jscp+i),a(jap+i),a(jee+i),
     +   a(jro+i)
         enddo
         read(*,*)
        endif
        goto 100 ! переход к следующему шагу по времени
  555 end

      subroutine sgl(a,n1,rb)
! процедура монотонизации
      real a(n1),rb(n1*2)
      do  k=1,5 ! 5 итераций заказано
      do i=1,n1 ! цикл по узлам
       i1=max(1,i-1) ! левый соседний узел
       i2=min(n1,i+1) ! правый соседний узел
       rb(i+n1)=a(i1)+a(i2)-2*a(i) ! определение второй производной в узле i
      enddo
      do i=1,n1 ! цикл по узлам
      i1=min(n1,i+1) ! правый
      i2=max(1,i-1)   ! левый
      rb(i)=a(i) ! старое значение функции
      if(  ! если вторая производная меняет знак то надо сглаживать
c     + a(i).gt.amax1(a(i1),a(i2)).or.
c     + a(i).lt.amin1(a(i1),a(i2)).or.
     +  rb(i+n1)*rb(i1+n1).lt.0.0 ! если вторая производная меняет знак то надо сглаживать
     + .or.rb(i+n1)*rb(i2+n1).lt.0.0
     + )rb(i)=(rb(i)+(a(i1)+a(i2))/2.0)/2.0 ! лаксово сглаживание
      enddo
      do i=1,n1
       a(i)=rb(i) ! пересылка сглаженных значений
      enddo
      enddo !k ! конец цикла итераций
      end

      subroutine sgl0(a,n1,rb)
! вариант более примитивного монотонизатора (тоже работает)
      real a(n1),rb(n1*2) ! сглаживает если a(i)>a(i-1) и a(i) > a(i+1) или меньше обоих
      do  k=1,5
      do i=1,n1
      i1=min(n1,i+1)
      i2=max(1,i-1)
      rb(i)=a(i)
      if(a(i).gt.amax1(a(i1),a(i2))
     + .or.a(i).lt.amin1(a(i1),a(i2))
     + )rb(i)=(rb(i)+(a(i1)+a(i2))/2.0)/2.0
      enddo
      do i=1,n1
       a(i)=rb(i)
      enddo
      enddo !k
      end
```