**Princ.f90**

!

! main (nonlineq)

! July 15, 2011

!

! implicit real \*8 (a-h,o-z)

 use msflib

 logical chamada

 parameter(neqmax=15)

 dimension u(neqmax),g(neqmax),h(neqmax)

 common /pert/ epsilon

 common /method/ iflag

 common /parpicard/ alfa

!

! input file

!

 open (unit=1,file='inp1.txt',status='old')

!

! output file

!

 open (unit=2,file='out1.txt',status='unknown')

 open (10,file='dados1.txt')

 open (11,file='dados2.txt')

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!

! Decide what method to use (iflag = 1 - exact Newton; 2 - approximate Newton; 3 - Picard)

!

 read(1,\*)iflag

 write(\*,\*)'iflag=',iflag

 read(1,\*)alfa

 write(\*,\*)'alfa=',alfa

 read (1,\*)maxit,neq,tol1,tol2,epsilon

 write(\*,\*)'maxit=',maxit,'neq=',neq

 write(\*,\*)'tol1=',tol1,'tol2=',tol2,'epsilon=',epsilon

!

! dados para o grafico das duas equacoes

!

 xmax=1.

 xmin=0.

 dx=(xmax-xmin)/100

 xx=xmin

 do i=1,101

 y1=cos(xx)

 write(10,\*)xx,y1

 ! write(\*,\*)xx,y1

 y2=asin(xx)

 write(11,\*)xx,y2

 ! write(\*,\*)xx,y2

 xx=xx+dx

 enddo

 close(10)

 close(11)

 chamada=systemqq('wgnuplot dados.gnu')

! stop

! read initial guesses

 read(1,\*)(u(i),i=1,neq)

 write(\*,\*)'the initial values are u(i):'

 write(\*,\*)(i,u(i),i=1,neq)

 ! call fcn(neq,u,g)

! gnorm=rnorm2(neq,g,neqmax)

! write(2,\*)'--------------------------------------------------- i = ',i

 ! write(\*,\*)'gnorm=',gnorm

! write(2,\*)'gnorm=',gnorm

! stop

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!

! solution of nonlinear equations

!

 if(iflag.eq.1.or.iflag.eq.2) then

 call mnewt(maxit,u,neq,tol1,tol2)

 endif

 if(iflag.eq.3) then

 do i=1,maxit

 call fcn(neq,u,g)

 gnorm=rnorm2(neq,g,neqmax)

 write(\*,\*)'gnorm=',gnorm

 write(2,\*)'gnorm=',gnorm

 if(gnorm.le.tol1) then

 write(\*,\*)'converged in ',i-1,' iterations'

 write(2,\*)'converged in ',i-1,' iterations'

 stop

 endif

 write(\*,\*) 'Picard method'

 write(2,\*) 'Picard method'

 write(\*,\*)'--------------------------------------------------- i = ',i

 write(2,\*)'--------------------------------------------------- i = ',i

 call fcn2(neq,u,h)

 do j=1,neq

 u(j)=alfa\*u(j)+(1.-alfa)\*h(j)

 write(\*,\*)'u(',j,')=',u(j)

 write(2,\*)'u(',j,')=',u(j)

 enddo

 enddo

 endif

 stop

 end

!

!----------------------------------------------------------------

 function rnorm2(n,x,nd)

!

! compute euclidean norm of a vector

!

! implicit real \*8 (a-h,o-z)

 dimension x(nd)

 sum=0.d0

 do i=1,n

 sum=sum+x(i)\*x(i)

 enddo

 aux=sqrt(sum)

 rnorm2=aux

 return

 end

!------------------------------------------------------------

 function rninf(n,x,nd)

!

! compute infinity norm of a vector

!

! implicit real \*8 (a-h,o-z)

 dimension x(nd)

 rmax=0.d0

 do i=1,n

 if(abs(x(i)).gt.rmax) then

 rmax=abs(x(i))

 endif

 enddo

 rninf=rmax

 return

 end

!------------------------------------------------------------------

 subroutine usrfun(u,n,np,g,a)

!

! compute the values of g (through function fcn)

! for a given u and the jacobean

!

! implicit real \*8 (a-h,o-z)

 parameter (np1=15)

 dimension u(n),g(np),a(np,np),ge(np1)

 common /pert/ epsilon

 call fcn(n,u,g)

! do it=1,n

! write(\*,\*)'g(',it,')=',g(it)

! enddo

!

! build the jacobean numerically

!

! write(\*,\*)'epsilon=',epsilon

 do j=1,n

 u(j)=u(j)+epsilon

 call fcn(n,u,ge)

 u(j)=u(j)-epsilon

 do i=1,n

 a(i,j)=(ge(i)-g(i))/epsilon

! write(\*,\*)'ge(',i,')=',ge(i)

 enddo

 enddo

 return

 end

!234567890123456789012345678901234567890123456789012345678901234567890

!--------------------------------------------

**Inp1.txt**

1 ! iflag = 1 - exact Newton; 2 - approximate Newton; 3 - Picard (successive substitution)

 0.1 ! alfa = parameter for Functional iteration and Picard methods (0<alfa<1)

 100 2 1.d-6 1.d-6 1.d-5 ! maxit = Nr max iteracoes; neq = Nr equacoes; tol1 = 1a tol; tol2 = 2a tol; epsilon = delta para derivada numerica

 1.d0 1.d0 ! u(i) = valores iniciais para o vetor de incógnitas

**lubksb.f90**

 SUBROUTINE lubksb(a,n,np,indx,b)

 ! implicit real \*8 (a-h,o-z)

 dimension indx(np),a(np,np),b(np)

 ii=0

 do 12 i=1,n

 ll=indx(i)

 sum=b(ll)

 b(ll)=b(i)

 if (ii.ne.0)then

 do 11 j=ii,i-1

 sum=sum-a(i,j)\*b(j)

11 continue

 else if (sum.ne.0.) then

 ii=i

 endif

 b(i)=sum

12 continue

 do 14 i=n,1,-1

 sum=b(i)

 do 13 j=i+1,n

 sum=sum-a(i,j)\*b(j)

13 continue

 b(i)=sum/a(i,i)

14 continue

 return

 END

**ludcmp.f90**

 SUBROUTINE ludcmp(a,n,np,indx,d)

 ! implicit real \*8 (a-h,o-z)

 PARAMETER (NMAX=10,TINY=1.0e-20)

 dimension indx(np),a(np,np),vv(NMAX)

 d=1.

 do 12 i=1,n

 aamax=0.

 do 11 j=1,n

 if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))

11 continue

 if (aamax.eq.0.) then

! do ja=1,n

! do jb=1,n

! write(\*,\*)'a(',ja,',',jb,')=',a(ja,jb)

! enddo

! enddo

 pause 'singular matrix in ludcmp'

 endif

 vv(i)=1./aamax

12 continue

 do 19 j=1,n

 do 14 i=1,j-1

 sum=a(i,j)

 do 13 k=1,i-1

 sum=sum-a(i,k)\*a(k,j)

13 continue

 a(i,j)=sum

14 continue

 aamax=0.

 do 16 i=j,n

 sum=a(i,j)

 do 15 k=1,j-1

 sum=sum-a(i,k)\*a(k,j)

15 continue

 a(i,j)=sum

 dum=vv(i)\*abs(sum)

 if (dum.ge.aamax) then

 imax=i

 aamax=dum

 endif

16 continue

 if (j.ne.imax)then

 do 17 k=1,n

 dum=a(imax,k)

 a(imax,k)=a(j,k)

 a(j,k)=dum

17 continue

 d=-d

 vv(imax)=vv(j)

 endif

 indx(j)=imax

 if(a(j,j).eq.0.)a(j,j)=TINY

 if(j.ne.n)then

 dum=1./a(j,j)

 do 18 i=j+1,n

 a(i,j)=a(i,j)\*dum

18 continue

 endif

19 continue

 return

 END

**mnewt.f90**

 SUBROUTINE mnewt(ntrial,x,n,tolx,tolf)

 ! implicit real \*8 (a-h,o-z)

 common /method/ iflag

 PARAMETER (NP=15)

 dimension x(n),indx(NP)

! USES lubksb,ludcmp,usrfun

 dimension fjac(NP,NP),fvec(NP),p(NP)

 do 14 k=1,ntrial

 write(\*,\*)'---------------k=',k

 if(iflag.eq.2) then

 write(\*,\*)'Approximate Newton Method'

 write(2,\*)'Approximate Newton Method'

 call usrfun(x,n,NP,fvec,fjac)

 endif

 if(iflag.eq.1) then

 write(\*,\*)'Exact Newton Method'

 write(2,\*)'Exact Newton Method'

 call usrfunex(x,n,NP,fvec,fjac)

 endif

 errf=0.

 do 11 i=1,n

 errf=errf+abs(fvec(i))

11 continue

 write(\*,\*)'errf=',errf

 write(2,\*)'errf=',errf

 if(errf.le.tolf) then

 write(\*,\*)'converged in ',k,' iterations'

 write(2,\*)'converged in ',k,' iterations'

 return

 endif

! if(errf.le.tolf)return

 do 12 i=1,n

 p(i)=-fvec(i)

12 continue

 call ludcmp(fjac,n,NP,indx,d)

 call lubksb(fjac,n,NP,indx,p)

 errx=0.

 do 13 i=1,n

 errx=errx+abs(p(i))

 x(i)=x(i)+p(i)

 write(\*,\*)'x(',i,')=',x(i)

 write(2,\*)'x(',i,')=',x(i)

13 continue

 enorm=rnorm2(n,p,np)

 write(\*,\*)'enorm=',enorm,'errx=',errx

 write(2,\*)'enorm=',enorm,'errx=',errx

 if(errx.le.tolx.or.enorm.le.tolx) then

 write(\*,\*)'converged in ',k,' iterations'

 write(2,\*)'converged in ',k,' iterations'

 return

 endif

14 continue

 return

 END

**sysaula.f90**

 subroutine fcn(n,fi,f)

!

! the system of equations

!

! implicit real \*8 (a-h,o-z)

 dimension fi(n),f(n)

 f(1)=cos(fi(1))-fi(2)

 f(2)=fi(1)-sin(fi(2))

! Sistema para o livro Calc Num

!

! f(1)=fi(1)\*\*3+2\*fi(2)\*\*2-fi(3)-20.

! f(2)=2\*fi(1)-fi(1)\*fi(2)+fi(3)-2.

! f(3)=fi(1)+fi(2)-fi(3)-5.

!

 return

 end

!-----------------------------------------------------------------

 subroutine usrfunex(u,n,np,g,a)

!

! compute the values of g (through function fcn)

! for a given u and the jacobean

!

! implicit real \*8 (a-h,o-z)

 dimension u(n),g(np),a(np,np)

!

! evaluate the values of g

!

 call fcn(n,u,g)

!

! build the jacobean exactly

!

 a(1,1)=-sin(u(1))

 a(1,2)=-1.

 a(2,1)=1.

 a(2,2)=-cos(u(2))

 return

 end

 !-----------------------------------------------------------------

 subroutine fcn2(n,fi,f)

!

! the system of equations

!

! implicit real \*8 (a-h,o-z)

 dimension fi(n),f(n)

! Sistema para o livro Calc Num

!

! f(1)=(-2\*fi(2)\*\*2+fi(3)+20.)\*\*(1./3.)

! f(2)=(2\*fi(1)+fi(3)-2.)/fi(1)

! f(3)=fi(1)+fi(2)-5.

!

 f(1)=sin(fi(2))

 f(2)=cos(fi(1))

 return

 end

!-----------------------------------------------------------------

**dados.gnu**

set data style linespoints

set grid

set xlabel 'variavel x'

set ylabel 'variavel y'

set title 'Sistema de 2 equacoes e 2 incognitas'

plot 'dados1.txt','dados2.txt'

pause -1