Referências bibliográficas

BIRD, R. B.; ARMSTRONG, R. C.; HASSAGER, O. **Dynamics of polymeric liquids**. Wiley & Sons, (1987)

CARVALHO, M. S. **Roll coating flows in rigid and deformable gaps**. Ph. D. thesis, University of Minnesota, (1995)

CARVALHO, M. S. **Elementos finitos**. Notas de aula, PUC-Rio, (2002)

CHAPRA, S. C.; CANALE, R. P. **Numerical methods for engineers.** McGraw Hill, (2002)

COHEN, E. D.; GUTOFF, E. B. **Modern coating and drying technology.** VCH Publishers Inc., (1992)

CRISTODOULOU, K. N. ; SCRIVEN, L. E. **Discretization of free surface flows and other moving boundary problems.** Jounal of computational physics, vol. 99, 39-55, (1992)

FENG, J. Q.; **Computational analysis of slot coating on a tensioned web.** AIChE J., Vol. 44, No 10, 2137 - 2143, (1998)

GATES, I. A. **Slot coating flows: feasibility, quality**. Ph.D. thesis, University of Minnesota, (1999)

HIGGINS, B. G.; SCRIVEN, L. E. **Capillary pressure and viscous pressure drop set bounds on coating bead operability.** Chemical Engineering Science, vol. 35, 673-682, (1980)

HUH, C.; SCRIVEN, L. E. **Hidrodynamic model of steady movement of a solid/liquid/liquid contact line.** J. Coll. Interface Sci., 35, 85, (1971)

JOOS, F. M. **A simple model of frequency response for slot coaters.** Technical session presentation 3th European Coating Symposium, University of Eriangen-Nurenberg, (1999)

KISTLER, S. F. **Liquid film coating.** Prantice Hall, (1987)

LIMA, E. L. **Análise no espaço R^{n}.** Coleção Matemática Universitária. IMPA, (2007)

LIMA, E. L. **Análise real vol. 2.** Coleção Matemática Universitária. IMPA, (2007)

LIU, T. J.; LIN, C. F.;CHANG Y. R.; CHANG H. M. **Comparison of vertical and horizontal slot die coatings.** Polymer Engineering and Science, vol. 47, 1927-1936, (2007)

LIU, T. J.; LIU, C. M.; LIN, F. H.; WU P. Y.; **Experimental study on tensioned-web slot coating.** Polymer Engineering and Science, vol. 47, 841-851, (2007)

LIU, T. J.; YU, W. J.; YU, T. A.; **Reduction of the minimum wet thickness in extrusion slot coating.** Chemical Engineering Science, vol. 50, n°6, 917-920, (1995)

MARTÍNEZ, J. M.; SANTOS, S. A. **Métodos computacionais de otimização.** Notas de aula, IMECC-UNICAMP, (1995)

NOCEDAL, J.; WRIGHT, S. J. **Numerical optimization.** Springer, (1999)

PANTON, R. L. **Incompressible flow.** Wiley & Sons, (2005)

PAPANASTASIOU, T. C.; MALAMATARIS, N.; ELLWOOD, K. **A new outflow boundary condition.** International Journal for Numerical Methods in Fluids, vol. 14, 587, (1992)

PARK, E.; SCRIVEN, L. E.; CARVALHO, M. S. **Physics of coating tensioned web over slot die.** Technical session presentation 12th international symposium for coating science and technology, Rochester, (2004)

PEREZ, E. B. **Otimização do processo de secagem na manufatura de fitas adesivas.** Dissertação de mestrado, PUC-Rio, (2004)

PEREZ, E. B.; CARVALHO, M. S. **Drying of thin films of polymer solutions coated over impermeable substrates.** Heat Transfer Engineering, vol. 28(6), 559-566, (2007)

ROMERO, O. J. **Limite de vazão minima do processo de revestimento por extrusão de soluções poliméricas.** Tese de doutorado, PUC-Rio, (2003)

ROMERO, O. J.; CARVALHO, M. S. **Free surface flow solver.** Tutorial manual, PUC-Rio, (2005)

ROMERO, O. J.; CARVALHO, M. S. **Response of slot coating flows to periodic disturbances.** Chemical Engineering Science, vol. 63, 2161-2173, (2008)

SARTOR, L. **Slot coating: fluid mechanics and die design**. Ph. D. thesis, University of Minnesota, (1990)

SCRIVEN, L. E**. Intermediate fluid mechanics.** Course notes, University of Minnesota, (1989)

ZEVALLOS, G. A.; CARVALHO, M. S. **Presence of multiples steady states and hysteresis loop in viscous flows**. Anotações, PUC-Rio, (2003)

Apêndice: Programa de minimização de funções com restrição tipo caixa

```
c**********************************************************************
      program MainOtim
c**********************************************************************
c     BEGIN PROLOGUE  MainOtim
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:    Eduardo Perez
c     Purpose: Slot Die Geometry Otimization
c======================================================================
c     INCLUDE FILES
c     Include
c======================================================================
      parameter (NProjVarMax = 10)
c======================================================================
c     EXTERNAL ARRAYS
c
      integer NProjVar,OutBound,LowActRtr(NProjVarMax),
     +          HghActRtr(NProjVarMax),iterMain,iterTotal
c
      real*8  Hess(NProjVarMax,NProjVarMax),ProjVarOld(NProjVarMax),
     +       ProjVar(NProjVarMax),GradObFct(NProjVarMax),
     +       LowBound(NProjVarMax),HighBound(NProjVarMax),
     +       MinBoundRange,MaxBoundRange,TrtRegLowBound(NProjVarMax),
     +       TrtRegHighBound(NProjVarMax),TrtRegSize,QuadIni,QuadFin,
     +       ObjFctIni,ObjFctFin,alpha,LowBoundOmega(NProjVarMax),
     +       HighBoundOmega(NProjVarMax),NegPrjGrad(NProjVarMax),
     +       QuadProjVar(NProjVarMax),QuadLowBoundOmega(NProjVarMax),
     +       QuadHighBoundOmega(NProjVarMax),TransFactor(NProjVarMax)
c
c**********************************************************************
c     ITERACTION COUNTER
c**********************************************************************
      iterMain=0
c
c**********************************************************************
c     PRINT HEADINGS
c**********************************************************************
```

```
c
      write(*,100)
c
c*********************************************************************
c     INITIAL TRUST REGION SIZE FACTOR
c*********************************************************************
c
    TrtRegSize=1.0
c
c*********************************************************************
c     PROPORTION OF THE QUADRACT FUNTION DECREASE REQUIRED TO ACEPT
c     x(k+1)
c*********************************************************************
c
      alpha=0.3
c
c*********************************************************************
c     TEST DATA
c*********************************************************************
c
      Include 'UserInfo.i'
c
c*********************************************************************
c     VERIFY IF INITIAL PROJECT VARIABLES ARE INSIDE BOUNDARIES
c*********************************************************************
c
    call BoundChk(NProjVarMax,NProjVar,LowBound,HighBound,ProjVar,
     +              OutBound)
c
    if (OutBound.eq.1) then
c
    write(*,*)"Project Variable(s) out of Range "
c
    pause
c
    goto 1000
c
    end if
c
c*********************************************************************
c     EVALUATE OBJECT FUNCTION VALUE ON x0
c*********************************************************************
c
      call ObjFctEval(NProjVarMax,ProjVar,
     +              ObjFctIni)
```

```
c
c*********************************************************************
c  CALCULATE THE GRADIENT OF THE OBJECT FUNCTION ON x0
c*********************************************************************
c
      call GradObjFctEval(NProjVarMax,ProjVar,
     +                    GradObFct)
c
c*********************************************************************
c  CALCULATE THE HESS OF THE OBJECT FUNCTION ON x0
c*********************************************************************
c
      call HessObjFctEval(NProjVarMax,ProjVar,
     +                    Hess)
c
c*********************************************************************
c     EVALUATION OF THE APROXIMATING QUADRATIC FUNCTION ON x0
c*********************************************************************
c
       QuadIni=ObjFctIni
c
c*********************************************************************
c  VERIFY ACTIVE RESTRICTIONS ON x0
c*********************************************************************
c
      call ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,ProjVar,
     +              LowActRtr,HghActRtr)
c
c*********************************************************************
c  CALCULATE THE NEGATIVE PROJECTED GRADIENT FOR THE OBJECT FUNCTION
c*********************************************************************
c
      call NegPrjGradObFct(NProjVarMax,NProjVar,LowActRtr,
     +                     HghActRtr,GradObFct,
     +                     NegPrjGrad)
c
c*********************************************************************
c  CALCULATE THE NORM OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
      GradNorm = dnrm2(NProjVar, NegPrjGrad, 1)
c
c*********************************************************************
c     BEGIN LOOP
c*********************************************************************
```

```
c
      do while (GradNorm>0.000001)
c
c*********************************************************************
c     DETERMINATION OF THE BOUNDARIES OF THE TRUST REGION
c*********************************************************************
c
 2000 call TrtRegBound (NProjVarMax,NProjVar,ProjVar,TrtRegSize,
     +                  TrtRegLowBound,TrtRegHighBound)
c
c*********************************************************************
c     DETERMINATION OF THE BOUNDARIES OF OMEGA,THE INTERSECTION BETWEEN
c     TRUST REGION AND PROJECT VARIABLES RANGE
c*********************************************************************
c
      call Omega(NProjVarMax,NProjVar,TrtRegLowBound,
     +           TrtRegHighBound,LowBound,HighBound,
     +           LowBoundOmega,HighBoundOmega)
c
c*********************************************************************
c     DETERMINE THE NARROWER AND THE WIDER BOUNDARY RANGE OF OMEGA
c*********************************************************************
c
    call MinMaxBndRng(NProjVarMax,NProjVar,LowBoundOmega,
     +                HighBoundOmega,
     +                MinBoundRange,MaxBoundRange)
c
c*********************************************************************
c     KEEP THE LAST ACCEPTED VALUE OF PROJECT VARIABLES
c*********************************************************************
c
      call OpProjVar(NProjVarMax,NProjVar,ProjVar,
     +               ProjVarOld)
c
c*********************************************************************
c     DETERMINATION OF THE SUITABLE INITIAL VALUE OF THE PROJECT VARIABLES
c     TO PROCEED TO MINIMIZATION OF APROXIMATING QUADRATIC FUNCTION IN
c     THE NEW REGION OMEGA
c*********************************************************************
c
      call IniProjVar  (NProjVarMax,NProjVar,LowBoundOmega,
     +                  HighBoundOmega,Hess,GradObFct,
     +                  ProjVar)
```

```
c*********************************************************************
c              DEFINE VARIABLE (x-xk) AND ITS BOUNDARY TO BE MINIMIZED
c    IN THE QUADRATIC APROXIMATING FUNCTION
c*********************************************************************
c
     call ProjVarLessProjVarOld(NProjVarMax,NProjVar,ProjVar,
    +              ProjVarOld,LowBoundOmega,HighBoundOmega,
    +    QuadProjVar,QuadLowBoundOmega,QuadHighBoundOmega,TransFactor)
c
c*********************************************************************
c    CALL CONJUGATED GRADIENT METHOD TO MINIMIZE A BOUNDED QUADRATIC
c              FUNCTION THAT APROXIMATES THE REAL FUNCTION CLOSE TO xk
c*********************************************************************
c
   call QuadBoundMin(NProjVarMax,NProjVar,Hess,GradObFct,
    +              QuadLowBoundOmega,QuadHighBoundOmega,
    +              MinBoundRange,MaxBoundRange,
    +              QuadProjVar,iterTotal)
c
c*********************************************************************
c    UPDATE PROJECT VARIABLES
c*********************************************************************
c
     call BackProjVar(NProjVarMax,NProjVar,QuadProjVar,TransFactor,
    +              ProjVar)
c
c*********************************************************************
c    CALCULATION OF THE VALUE OF THE QUADRATIC FUNCTION ON u*=(x*-xk)
c*********************************************************************
c
     call QuadValue (NProjVarMax,NProjVar,Hess,GradObFct,
    +             QuadProjVar,ObjFctIni,
    +             QuadFin)
c
c*********************************************************************
c    EVALUATE OBJECT FUNCTION ON x(k+1)
c*********************************************************************
c
     call ObjFctEval(NProjVarMax,ProjVar,
    +              ObjFctFin)
c
c*********************************************************************
c    CRITERIA TO DECIDE IF ACEPT x(k+1)
c*********************************************************************
c
```

```
   if (ObjFctFin.le.(ObjFctIni+alpha*(QuadFin-QuadIni))) then
c
c-----------------------------------------------------------------------
c  IF A GOOD DECREASE ON OBJECT FUNCTION WAS GOT, INCREASE TRUST
c     REGION SIZE FOR THE NEXT ITERATION
c-----------------------------------------------------------------------
c
   if (ObjFctFin.le.(ObjFctIni+(0.9)*(QuadFin-QuadIni))) then
c
   TrtRegSize=2*TrtRegSize
c
   end if
c
c-----------------------------------------------------------------------
c  CALCULATE THE GRADIENT OF THE OBJECT FUNCTION ON x(k+1)
c-----------------------------------------------------------------------
c
      call GradObjFctEval(NProjVarMax,ProjVar,
     +                    GradObFct)
c
c-----------------------------------------------------------------------
c  CALCULATE THE HESS OF THE OBJECT FUNCTION ON x(k+1)
c-----------------------------------------------------------------------
c
      call HessObjFctEval(NProjVarMax,ProjVar,
     +                    Hess)
c
c-----------------------------------------------------------------------
c  VERIFY ACTIVE RESTRICTIONS ON x(k+1)
c-----------------------------------------------------------------------
c
      call ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,ProjVar,
     +              LowActRtr,HghActRtr)
c
c-----------------------------------------------------------------------
c  CALCULATE THE NEGATIVE PROJECTED GRADIENT FOR THE OBJECT FUNCTION
c     ON x(k+1)
c-----------------------------------------------------------------------
c
      call NegPrjGradObFct(NProjVarMax,NProjVar,LowActRtr,
     +                     HghActRtr,GradObFct,
     +                     NegPrjGrad)
c
c-----------------------------------------------------------------------
c  CALCULATE THE NORM OF THE NEGATIVE PROJECTED GRADIENT
```

```
c-----------------------------------------------------------------------
c
     GradNorm = dnrm2(NProjVar, NegPrjGrad, 1)
c
c-----------------------------------------------------------------------
c
     ObjFctIni=ObjFctFin
   QuadIni=ObjFctIni
c
     iterMain=iterMain+1
c
c-----------------------------------------------------------------------
c  PRINT RESUME OF THIS ITERACTION
c-----------------------------------------------------------------------
c
   write(*,*) "-------------------------------------------------"
   write(*,*) "                                                 "
     write(*,*) "Iteraction #:",iterMain
     write(*,*) "Norm of function gradient:",GradNorm
   write(*,*) "Object Function Value:",ObjFctFin
     write(*,*) "# iteractions of quadratic minimization:",iterTotal
c
     do 10 iProjVar=1,NProjVar
c
     write(*,*) "Project Variables:",ProjVar(iProjVar)
c
  10  continue
c
c-----------------------------------------------------------------------
c
   else
c
c-----------------------------------------------------------------------
c     REDUCE TRUST REGION SIZE
c-----------------------------------------------------------------------
c
                TrtRegSize=(0.8)*TrtRegSize
c
c-----------------------------------------------------------------------
c     RETURN TO xk
c-----------------------------------------------------------------------
c
     call OpProjVar(NProjVarMax,NProjVar,ProjVarOld,
     +              ProjVar)
c
```

```
   goto 2000
c
   end if
c
c-----------------------------------------------------------------
c    END OF LOOP
c-----------------------------------------------------------------
c
   end do
c
c*********************************************************************
c    FORMAT STATEMENT
c*********************************************************************
c
 100  format (/2x,'*****************************'/,
     +        2x,'SLOT DIE GEOMETRY OTIMIZATION'/,
     +        2x,'       COAT - 2007          '/,
     +        2x,'     Eduardo Perez          '/,
     +        2x,'    Marcio S Carvalho       '/,
     +        2x,'*****************************'/)
c
c*********************************************************************
c    PRINT RESUME
c*********************************************************************
c
   write(*,*) "*****************************************************"
   write(*,*) "                                                     "
     write(*,*) "Critical Point Found."
     write(*,*) "Number of iteractions:",iterMain
   write(*,*) "Object Function Value:",ObjFctFin
   write(*,*) "                                                     "
c
     do 70 iProjVar=1,NProjVar
c
     write(*,*) "Project Variables:",ProjVar(iProjVar)
c
  70  continue
c
     pause
c
 1000 end
c
c**** MainOtim ends here ******************************************
c*********************************************************************
```

```
c***********************************************************************
      subroutine ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,
     +                    XProjVar,
     +                    LowActRtr,HghActRtr)
c
c***********************************************************************
c     BEGIN PROLOGUE  ActRestr
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Check active restrictions on current value of project
c              variable
c-----------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,NProjVar,LowActRtr(NProjVarMax),
     +        HghActRtr(NProjVarMax)
c
      real*8 XProjVar(NProjVarMax),LowBound(NProjVarMax),
     +       HighBound(NProjVarMax)
c
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
c     RESET VARIABLES
c-----------------------------------------------------------------------
c
    do 1 iProjVar=1,NProjVar
c
      LowActRtr(iProjVar)=0
c
      HghActRtr(iProjVar)=0
c
  1               continue
c
c-----------------------------------------------------------------------
c
    do 2 iProjVar=1,NProjVar
c
    if(XProjVar(iProjVar).eq.LowBound(iProjVar)) then
c
      LowActRtr(iProjVar)=1
c
    elseif(XProjVar(iProjVar).eq.HighBound(iProjVar)) then
```

```
c
      HghActRtr(iProjVar)=1
c
    else
c
      LowActRtr(iProjVar)=0
    HghActRtr(iProjVar)=0
c
    endif
c
   2              continue
c
      end subroutine
c
c**** ActRestr ends here ***************************************************
c*****************************************************************************

c*****************************************************************************
      subroutine ActRestrChg(NProjVarMax,NProjVar,LowActRtr,HghActRtr,
     +                  LowActRtrNew,HghActRtrNew,
     +                  RtrChgFlag)
c*****************************************************************************
c     BEGIN PROLOGUE  ActRestrChg
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Verify if active restrictions changed
c-----------------------------------------------------------------------------------------------------------
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,NProjVar,LowActRtr(NProjVarMax),
     +       HghActRtr(NProjVarMax),LowActRtrNew(NProjVarMax),
     +       HghActRtrNew(NProjVarMax),RtrChgFlag
c
c*****************************************************************************
c
    RtrChgFlag=0
c
      do 1 iProjVar=1,NProjVar
c
    if ((LowActRtrNew(iProjVar).ne.LowActRtr(iProjVar)).or.
     +   (HghActRtrNew(iProjVar).ne.HghActRtr(iProjVar))) then
c
    RtrChgFlag=1
c
```

```
   end if
c
 1continue
c
     endsubroutine
c
c**** ActRestrChg ends here ***********************************************
c*****************************************************************************

c*****************************************************************************
      subroutine BackProjVar(NProjVarMax,NProjVar,QuadProjVar,
     +                  TransFactor,
     +                  ProjVar)
c
c*****************************************************************************
c     BEGIN PROLOGUE BackProjVar
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer: Eduardo Perez
c     Purpose: Do ProjVar=QuadProjVar-Transfactor
c-----------------------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer NProjVar,iProjVar
c
      real*8  ProjVar(NProjVarMax),
     +          QuadProjVar(NProjVarMax),TransFactor(NProjVarMax)
c
      do 1 iProjVar=1,NProjVar
c
      ProjVar(iProjVar)=QuadProjVar(iProjVar)-TransFactor(iProjVar)
c
  1             continue
c
      end subroutine
c
c**** BackProjVar ends here ***********************************************
c*****************************************************************************


c*****************************************************************************
      subroutine BoundChk(NProjVarMax,NProjVar,LowBound,HighBound,
     +                  XProjVar,
     +                  OutBound)
```

```
c
c*********************************************************************
c    BEGIN PROLOGUE  BoundChk
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer:Eduardo Perez
c    Purpose: Verify if the project variables are inside the boundaries
c---------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     integer iProjVar,NProjVar,OutBound
c
     real*8 XProjVar(NProjVarMax),LowBound(NProjVarMax),
    +       HighBound(NProjVarMax)
c
c-------------------------------------------------------------------
c
   OutBound=0
c
     do 1 iProjVar=1,NProjVar
c
   if ((XProjVar(iProjVar).lt.LowBound(iProjVar)).or.
    +   (XProjVar(iProjVar).gt.HighBound(iProjVar))) then
c
   OutBound=1
c
   endif
c
  1             continue
c
     end subroutine
c
c**** BoundChk ends here ******************************************
c*********************************************************************


c*********************************************************************
     subroutine ChpGradCalc(NProjVarMax,NProjVar,LowActRtr,
    +                  HghActRtr,GradQuad,
    +                  ChpGrad)
c*********************************************************************
c    BEGIN PROLOGUE  ChpGradCalc
c    Date Written:  18 Oct 2007
c    Revision Date:
```

```
c    Programmer:Eduardo Perez
c    Purpose: Calculation of the choped gradient
c-------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     integer iProjVar,NProjVar,LowActRtr(NProjVarMax),
    +       HghActRtr(NProjVarMax)
c
     real*8 GradQuad(NProjVarMax,2),ChpGrad(NProjVarMax)
c
c**********************************************************************
c    CALCULATION OF THE CHOPED GRADIENT
c**********************************************************************
c
     do 1  iProjVar=1,NProjVar
c
   if ((LowActRtr(iProjVar).eq.0).and.
    +   (HghActRtr(iProjVar).eq.0)) then
c
   ChpGrad(iProjVar)=0
c
   elseif ((LowActRtr(iProjVar).eq.1).and.
    +      (GradQuad(iProjVar,1).gt.0)) then
c
   ChpGrad(iProjVar)=0
c
   elseif ((HghActRtr(iProjVar).eq.1).and.
    +      (GradQuad(iProjVar,1).lt.0)) then
c
   ChpGrad(iProjVar)=0
c
   else
c
   ChpGrad(iProjVar)=-GradQuad(iProjVar,1)
c
     endif
c
  1   continue
c
     endsubroutine
c
c**** ChpGradCalc ends here ***************************************************
c**********************************************************************
```

```
c*********************************************************************
      subroutine ConjGrad(NProjVarMax,NProjVar,Hess,GradObFct,
     +                    ProjVar,StepDir)
c
c*********************************************************************
c     BEGIN PROLOGUE  ConjGrad
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Unrestricted Minimization using Conjugate Gradient Method
c---------------------------------------------------------------------------------------------------
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,jProjVar,NProjVar,iter
      real*8  GradQuad(NProjVarMax,2),Hess(NProjVarMax,NProjVarMax),
     +        ProjVar(NProjVarMax,2),GradObFct(NProjVarMax),
     +        StepDir(NProjVarMax,2),GradQuadNew(NProjVarMax),GradNorm,
     +        GradTGrad,HessStep(NProjVarMax),StepTHessStep,
     +        StepLenght,GradTGradNew,BetaStepLenght
c
c*********************************************************************
c     CALCULATION OF THE QUADRATIC FUNCTION GRADIENT g=Hx+b
c*********************************************************************
c
      do 5  iProjVar=1,NProjVar
c
      do 10 jProjVar=1,NProjVar
c
    GradQuad(iProjVar,1)=Hess(iProjVar,jProjVar)*ProjVar(jProjVar,1)+
     +                   GradQuad(iProjVar,1)
c
   10 continue
c
                GradQuad(iProjVar,1)=GradQuad(iProjVar,1)+GradObFct(iProjVar)
c
    5  continue
c
c*********************************************************************
c     DIRECTION OF THE FIRST MINIMIZATION  d0=-g0
c*********************************************************************
c
      do 15 iProjVar=1,NProjVar
c
    StepDir(iProjVar,1)=-GradQuad(iProjVar,1)
```

```
c
 15           continue
c
     iter=0
c
c*********************************************************************
c    NORM OF THE QUADRATIC FUNCTION GRADIENT
c*********************************************************************
C
     do 17 iProjVar=1,NProjVar
c
   GradQuadNew(iProjVar)=GradQuad(iProjVar,1)
c
 17           continue

     GradNorm = dnrm2(NProjVar, GradQuadNew, 1)
c
c*********************************************************************
c     CALCULATION OF g'g TO BE USED ON DETERMINATION OF STEPLENGHT
c*********************************************************************
c
   GradTGrad=0
c
     do 20 iProjVar = 1,NProjVar
c
   GradTGrad=GradTGrad+GradQuad(iProjVar,1)*GradQuad(iProjVar,1)
c
 20  continue
c
c*********************************************************************
c    CALCULATION OF dk'Hdk: IF LESS THAN OR EQUAL TO ZERO THERE IS NO
c    MINIMUM IN dk DIRECTION, OTHERWISE LOOK FOR MINIMUM ON dk DIRECTION
c*********************************************************************
c
   StepTHessStep=0
c
     do 18  iProjVar=1,NProjVar
c
   do 19 jProjVar=1,NProjVar

c
   HessStep(iProjVar)=Hess(iProjVar,jProjVar)*StepDir(jProjVar,1)+
    +               HessStep(iProjVar)
c
 19  continue
```

```
c
  StepTHessStep=StepTHessStep +
   +            StepDir(iProjVar,1)*HessStep(iProjVar)
c
 18  continue
c
c*********************************************************************
c    BEGIN OF LOOP
c*********************************************************************
c
     do while ((GradNorm>0.000001).and.(StepTHessStep>0).and.
    +        (iter.lt.10))
c
  StepLenght=GradTGrad/StepTHessStep
c
  GradTGradNew=0
c
c*********************************************************************
c    CALCULATION OF x(k+1)=x(k)+a*dk AND g(k+1)=g(k)+a*H*dk
c*********************************************************************
     do 35 iProjVar=1,NProjVar
c
  ProjVar(iProjVar,2)=ProjVar(iProjVar,1)+
   +                    StepLenght*StepDir(iProjVar,1)
c
  GradQuad(iProjVar,2)=GradQuad(iProjVar,1)+
   +                     StepLenght*HessStep(iProjVar)
c
  GradTGradNew=GradTGradNew+
   +                GradQuad(iProjVar,2)*GradQuad(iProjVar,2)
c
 35  continue
c
c*********************************************************************
c
  BetaStepLenght=GradTGradNew/GradTGrad
c
  GradTGrad=GradTGradNew
c
c*********************************************************************
c    NEW DIRECTION TO MINIMIZATION d(k+1)=-g(k+1)+B*dk
c*********************************************************************

     do 40 iProjVar=1,NProjVar
c
```

```
      StepDir(iProjVar,2)=-GradQuad(iProjVar,2)+
     +                    BetaStepLenght*StepDir(iProjVar,1)
c
 40          continue
c
c*********************************************************************
c    VARIABLES UPDATE FOR THE NEXT ITERACTION
c*********************************************************************
c
      do 45 iProjVar=1,NProjVar
c
      StepDir(iProjVar,1)=StepDir(iProjVar,2)
      ProjVar(iProjVar,1)=ProjVar(iProjVar,2)
      GradQuad(iProjVar,1)=GradQuad(iProjVar,2)
      GradQuadNew(iProjVar)=GradQuad(iProjVar,2)
      HessStep(iProjVar)=0
c
 45          continue
c
c*********************************************************************
c    CALCULATION OF dk'Hdk: IF LESS THAN OR EQUAL TO ZERO THERE IS NO
c    MINIMUM IN dk DIRECTION, OTHERWISE LOOK FOR MINIMUM ON dk DIRECTION
c*********************************************************************
c
      StepTHessStep=0
c
      do 50  iProjVar=1,NProjVar
c
      do 55 jProjVar=1,NProjVar
c
      HessStep(iProjVar)=Hess(iProjVar,jProjVar)*StepDir(jProjVar,1)+
     +               HessStep(iProjVar)
c
 55  continue
c
      StepTHessStep=StepTHessStep +
     +           StepDir(iProjVar,1)*HessStep(iProjVar)
c
 50  continue
c
c*********************************************************************
c    NORM OF THE QUADRATIC FUNCTION GRADIENT
c*********************************************************************
c
      GradNorm = dnrm2(NProjVar,GradQuadNew, 1)
```

```
c
   iter=iter+1
c
     end do
c
c*********************************************************************
c    PRINT RESUME
c*********************************************************************
c
   if ((GradNorm.lt.0.000001).and.(iter.lt.10)) then
c
   write(*,*) "Critical Point Found."
   write(*,*) "Number of iteractions:",iter
c
   do 60 iProjVar=1,NProjVar
c
   write(*,*) "Project Variables:",ProjVar(iProjVar,2)
c
 60  continue
c
   end if
c
   if (StepTHessStep.lt.0) then
c
   write(*,*) "Direction of Negative Curvature Found"
   write(*,*) "Number of iteractions:",iter
c
   do 65 iProjVar=1,NProjVar
c
   write(*,*) "Step Direction:",StepDir(iProjVar,2)
c
 65  continue
c
   do 70 iProjVar=1,NProjVar
c
   write(*,*) "Last Project Variable Value:",ProjVar(iProjVar,2)
c
 70  continue

   end if
c
c*********************************************************************
c
     pause
c
```

```
      endsubroutine
c
c**** ConjGrad ends here ***************************************************
c**********************************************************************


c**********************************************************************
      subroutine DirCurvature(NProjVarMax,NProjVar,H,d,
     +                        dTHd)
c
c**********************************************************************
c     BEGIN PROLOGUE  DirCurvature
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Verify if the direction is of positive or negative
c              curvature
c----------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,jProjVar,NProjVar
c
      real*8 H(NProjVarMax,NProjVarMax),d(NProjVarMax),Hd(NProjVarMax),
     +       dTHd
c
c**********************************************************************
c     CALCULATION OF dk'Hdk
c**********************************************************************
c
   dTHd=0
c
      do 1 iProjVar=1,NProjVar
c
   do 2 jProjVar=1,NProjVar
c
   Hd(iProjVar)=H(iProjVar,jProjVar)*d(jProjVar)+Hd(iProjVar)
c
 2   continue
c
   dTHd=dTHd+d(iProjVar)*Hd(iProjVar)
c
      Hd(iProjVar)=0
c
 1   continue
```

```
c
      end subroutine
c
c**** DirCurvature ends here ***********************************************
c*********************************************************************


      double precision function dnrm2 ( n, dx, incx)
      integer i, incx, ix, j, n, next
      double precision   dx(*), cutlo, cuthi, hitest, sum, xmax,zero,one
      data   zero, one /0.0d0, 1.0d0/
c
c     euclidean norm of the n-vector stored in dx() with storage
c     increment incx .
c     if    n .le. 0 return with result = 0.
c     if n .ge. 1 then incx must be .ge. 1
c
c           c.l.lawson, 1978 jan 08
c     modified to correct failure to update ix, 1/25/92.
c     modified 3/93 to return if incx .le. 0.
c
c     four phase method     using two built-in constants that are
c     hopefully applicable to all machines.
c         cutlo = maximum of  dsqrt(u/eps)  over all known machines.
c         cuthi = minimum of  dsqrt(v)      over all known machines.
c     where
c         eps = smallest no. such that eps + 1. .gt. 1.
c         u   = smallest positive no.   (underflow limit)
c         v   = largest  no.            (overflow  limit)
c
c     brief outline of algorithm..
c
c     phase 1    scans zero components.
c     move to phase 2 when a component is nonzero and .le. cutlo
c     move to phase 3 when a component is .gt. cutlo
c     move to phase 4 when a component is .ge. cuthi/m
c     where m = n for x() real and m = 2*n for complex.
c
c     values for cutlo and cuthi..
c     from the environmental parameters listed in the imsl converter
c     document the limiting values are as follows..
c     cutlo, s.p.   u/eps = 2**(-102) for  honeywell.  close seconds are
c                   univac and dec at 2**(-103)
c                   thus cutlo = 2**(-51) = 4.44089e-16
c     cuthi, s.p.   v = 2**127 for univac, honeywell, and dec.
```

```
c                thus cuthi = 2**(63.5) = 1.30438e19
c     cutlo, d.p.   u/eps = 2**(-67) for honeywell and dec.
c                thus cutlo = 2**(-33.5) = 8.23181d-11
c     cuthi, d.p.   same as s.p.  cuthi = 1.30438d19
c     data cutlo, cuthi / 8.232d-11,  1.304d19 /
c     data cutlo, cuthi / 4.441e-16,  1.304e19 /
      data cutlo, cuthi / 8.232d-11,  1.304d19 /
c
      if(n .gt. 0 .and. incx.gt.0) go to 10
        dnrm2  = zero
        go to 300
c
   10 assign 30 to next
      sum = zero
      i = 1
      ix = 1
c                                   begin main loop
   20    go to next,(30, 50, 70, 110)
   30 if( dabs(dx(i)) .gt. cutlo) go to 85
      assign 50 to next
      xmax = zero
c
c                    phase 1.  sum is zero
c
   50 if( dx(i) .eq. zero) go to 200
      if( dabs(dx(i)) .gt. cutlo) go to 85
c
c                         prepare for phase 2.
      assign 70 to next
      go to 105
c
c                         prepare for phase 4.
c
  100 continue
      ix = j
      assign 110 to next
      sum = (sum / dx(i)) / dx(i)
  105 xmax = dabs(dx(i))
      go to 115
c
c                phase 2.  sum is small.
c                      scale to avoid destructive underflow.
c
   70 if( dabs(dx(i)) .gt. cutlo ) go to 75
c
```

```
c                  common code for phases 2 and 4.
c                  in phase 4 sum is large.  scale to avoid overflow.
c
  110 if( dabs(dx(i)) .le. xmax ) go to 115
        sum = one + sum * (xmax / dx(i))**2
        xmax = dabs(dx(i))
        go to 200
c
  115 sum = sum + (dx(i)/xmax)**2
      go to 200
c
c
c               prepare for phase 3.
c
   75 sum = (sum * xmax) * xmax
c
c
c     for real or d.p. set hitest = cuthi/n
c     for complex      set hitest = cuthi/(2*n)
c
   85 hitest = cuthi/float( n )
c
c                  phase 3.  sum is mid-range.  no scaling.
c
      do 95 j = ix,n
      if(dabs(dx(i)) .ge. hitest) go to 100
        sum = sum + dx(i)**2
        i = i + incx
   95 continue
      dnrm2 = dsqrt( sum )
      go to 300
c
  200 continue
      ix = ix + 1
      i = i + incx
      if( ix .le. n ) go to 20
c
c            end of main loop.
c
c            compute square root and adjust for scaling.
c
      dnrm2 = xmax * dsqrt(sum)
  300 continue
      return
      end
```

```
c**********************************************************************
      subroutine FstGradQuad(NProjVarMax,NProjVar,ProjVar,Hess,
     +                       GradObFct,LowActRtr,HghActRtr,
     +                 GradQuadAdj,GradQuad)
c
c**********************************************************************
c     BEGIN PROLOGUE  GradQuad
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Calculation of the quadratic function gradient for the
c              first step in the actual subspace.
c----------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,jProjVar,NProjVar,LowActRtr(NProjVarMax),
     +        HghActRtr(NProjVarMax)
c
      real*8 Hess(NProjVarMax,NProjVarMax),
     +       GradObFct(NProjVarMax),
     +       GradQuadAdj(NProjVarMax,2),GradQuad(NProjVarMax,2),
     +       ProjVar(NProjVarMax)
c
c**********************************************************************
c     RESET VARIABLES
c**********************************************************************
c
      do 1 iProjVar=1,NProjVar
c
      GradQuad(iProjVar,1)=0
c
                GradQuadAdj(iProjVar,1)=0
c
  1   continue
c
c**********************************************************************
c          CALCULATE QUADRATIC FUNCTION GRADIENT ON FULL MINIMIZATION
SUBSPACE
c**********************************************************************
c
      do 3 iProjVar=1,NProjVar
c
```

```
      do 5 jProjVar=1,NProjVar
c
     GradQuad(iProjVar,1)=Hess(iProjVar,jProjVar)*
     +                    ProjVar(jProjVar)+GradQuad(iProjVar,1)
c
 5   continue
c
              GradQuad(iProjVar,1)=GradQuad(iProjVar,1)+GradObFct(iProjVar)
c
      GradQuadAdj(iProjVar,1)=GradQuad(iProjVar,1)
c
 3   continue
c
c*********************************************************************
c       CALCULATE QUADRATIC FUNCTION GRADIENT ON ACTUAL MINIMIZATION
SUBSPACE
c*********************************************************************
c
      do 7 iProjVar=1,NProjVar
c
      if ((LowActRtr(iProjVar).eq.1).or.
     +  (HghActRtr(iProjVar).eq.1)) then
c
     GradQuadAdj(iProjVar,1)=0
c
     endif
c
 7            continue
c
      end subroutine
c
c**** FstGradQuad ends here ***************************************************
c*********************************************************************


c*********************************************************************
      subroutine GoChpGradDir(NProjVarMax,NProjVar,Hess,ChpGrad,
     +            ChpGradNorm,ProjVar,LowBound,
     +            HighBound,theta,MinBoundRange,
     +            LowActRtr,HghActRtr,ProjVarNew,iterTotal)
c
c*********************************************************************
c    BEGIN PROLOGUE  GoChpGradDir
c    Date Written:  18 Oct 2007
c    Revision Date:
```

```
c    Programmer:Eduardo Perez
c    Purpose: Define the steplenght on choped gradient direction, and
c             the new value of the project variables in this direction
c-------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     integer iProjVar,jProjVar,NProjVar,LowActRtr(NProjVarMax),
    +        HghActRtr(NProjVarMax),OutBound,iterTotal
c
     real*8  Hess(NProjVarMax,NProjVarMax),
    +        ProjVar(NProjVarMax),ProjVarNew(NProjVarMax),
    +        LowBound(NProjVarMax),HighBound(NProjVarMax),
    +        ChpGrad(NProjVarMax),ChpGradNorm,ChpGradTHessChpGrad,
    +        ChpDirStpLength,MinBoundRange,theta
c
c**********************************************************************
c    CHECK IF DIRECTION OF CHOPED GRADIENTE HAS POSITIVE OR NEGATIVE
c    CURVATURE
c**********************************************************************
c
     call DirCurvature(NProjVarMax,NProjVar,Hess,ChpGrad,
    +                  ChpGradTHessChpGrad)
c
   if (ChpGradTHessChpGrad.gt.0) then
c
c**********************************************************************
c    CALCULATE THE OPTIMUM STEP LENGHT TO MINIMIZE QUADRATIC ON
c    CHOPED GRADIENT      DIRECTION
c**********************************************************************
c
   ChpDirStpLength=(ChpGradNorm*ChpGradNorm)/(ChpGradTHessChpGrad)
c
c**********************************************************************
c    CALCULATION OF x(k+1)=x(k)+t*gc
c**********************************************************************
c
     do 1 iProjVar=1,NProjVar
c
   ProjVarNew(iProjVar)=ProjVar(iProjVar)+
    +                   ChpDirStpLength*ChpGrad(iProjVar)
c
  1             continue
c
c**********************************************************************
```

```
c    CHECK IF x(k+1) IS INSIDE THE BONDARIES, IF NOT, PROCEED LINEAR
c    SEARCH TO BRING IT INSIDE BONDARIES
c*********************************************************************
c
  OutBound=1
c
  do while (OutBound.eq.1)
c
    call BoundChk(NProjVarMax,NProjVar,LowBound,HighBound,
   +            ProjVarNew,
   +            OutBound)
c
c*********************************************************************
c    LINEAR SEARCH TO BRING x(k+1) INSIDE BONDARIES
c*********************************************************************
c
  if (OutBound.eq.1) then
c
    call LinSearch(NProjVarMax,NProjVar,ProjVar,OutBound,
   +             ProjVarNew,theta)
c
  end if
c
    end do
c
c*********************************************************************
c    MINIMIZATION IF CHOPED GRADIENT DIRECTION HAS NOT POSITIVE CURVATURE
c*********************************************************************
c
  elseif (ChpGradTHessChpGrad.le.0)     then
c
c*********************************************************************
c    CALCULATION OF x(k+1)=x(k)+t*gc
c*********************************************************************
c
    do 3 iProjVar=1,NProjVar
c
  ChpDirStpLength=MinBoundRange/ChpGradNorm
c
  ProjVarNew(iProjVar)=ProjVar(iProjVar)+
   +                    ChpDirStpLength*ChpGrad(iProjVar)
c
 3            continue
c
c*********************************************************************
```

```
c    END CHECK OF CURVATURE
c*********************************************************************
c
  end if
c
c*********************************************************************
c    UPDATE ACTIVE RESTRICTIONS
c*********************************************************************
c
     call ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,
    +            ProjVarNew,
    +            LowActRtr,HghActRtr)
c
c*********************************************************************
c    UPDATE PROJECT VARIABLES
c*********************************************************************
c
     do 5 jProjVar=1,NProjVar
c
   ProjVar(jProjVar)=ProjVarNew(jProjVar)
c
  5             continue
c
     iterTotal=iterTotal+1
c
     endsubroutine
c
c**** GoChpGradDir ends here ***********************************************
c*********************************************************************


c*********************************************************************
     subroutine GradObjFctEval(NProjVarMax,ProjVar,
    +                    GradObFct)
c*********************************************************************
c    BEGIN PROLOGUE  GradObjFctEval
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer:Eduardo Perez
c    Purpose: Evaluation of the gradient of the object function on xk
c
c-----------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
```

```
      real*8  ProjVar(NProjVarMax),X1,X2,X3,X4,X5,
     +          GradObFct(NProjVarMax)
c
c-----------------------------------------------------------------
c     RESET VARIABLES
c-----------------------------------------------------------------
c
      Include 'UserGradFctInfo.i'
c
      end subroutine
c
c**** GradObjFctEval ends here ***********************************************
c*******************************************************************************


c*******************************************************************************
      subroutine GradQuadNew(NProjVarMax,NProjVar,ProjVarNew,Hess,H,d,
     +                  StepLenght,GradObFct,
     +                  GradQuadAdj,GradQuad,dkTGradNew)
c
c*******************************************************************************
c     BEGIN PROLOGUE  GradQuadNew
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Calculation of the quadratic function gradient for the
c              next step. In the actual minimization subspace.
c-------------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,jProjVar,NProjVar
c
      real*8 H(NProjVarMax,NProjVarMax),Hess(NProjVarMax,NProjVarMax),
     +     d(NProjVarMax),Hd(NProjVarMax),GradObFct(NProjVarMax),
     +     GradQuadAdj(NProjVarMax,2),GradQuad(NProjVarMax,2),
     +     StepLenght,dkTGradNew,ProjVarNew(NProjVarMax)
c
c*******************************************************************************
c     CALCULATION OF Hdk
c*******************************************************************************
c
      do 1 iProjVar=1,NProjVar
c
    do 2 jProjVar=1,NProjVar
```

```
c
  Hd(iProjVar)=H(iProjVar,jProjVar)*d(jProjVar)+Hd(iProjVar)
c
 2  continue
c
 1  continue
c
c*********************************************************************
c    CALCULATION OF g(k+1)=H*x(k+1)+b IN THE FULL MINIMIZATION
c    SPACE
c*********************************************************************
c
    do 3 iProjVar=1, NProjVar
c
    GradQuad(iProjVar,2)=0
c
 3  continue
c
    do 4 iProjVar=1,NProjVar
c
    do 5 jProjVar=1,NProjVar
c
  GradQuad(iProjVar,2)=Hess(iProjVar,jProjVar)*
   +                    ProjVarNew(jProjVar)+GradQuad(iProjVar,2)
c
 5  continue
c
            GradQuad(iProjVar,2)=GradQuad(iProjVar,2)+GradObFct(iProjVar)
c
 4  continue
c
c*********************************************************************
c    CALCULATION OF g(k+1)=g(k)+a*H*dk IN THE ACTUAL MINIMIZATION
c    SUBSPACE
c*********************************************************************
c
    do 6 iProjVar=1,NProjVar
c
  GradQuadAdj(iProjVar,2)=GradQuadAdj(iProjVar,1)+
   +                     StepLenght*Hd(iProjVar)
c
  dkTGradNew=dkTGradNew+
   +               GradQuad(iProjVar,2)*GradQuad(iProjVar,2)
c
 6  continue
```

```
c
c*********************************************************************
c    RESET Hdk
c*********************************************************************
c
   do 7 iProjVar=1,NProjVar
c
   Hd(iProjVar)=0
c
  7   continue
c
     end subroutine
c
c**** GradQuadNew ends here ***************************************************
c*********************************************************************
```



```
c*********************************************************************
     subroutine HessFctGrdAdj(NProjVarMax,NProjVar,Hess,GradObFct,
    +                  LowActRtr,HghActRtr,
    +                  HessAdj,GradObFctAdj)
c*********************************************************************
c    BEGIN PROLOGUE  HessFctGrdAdj
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer:Eduardo Perez
c    Purpose: Adjust the Hessian and gradient of the object function
c           based on active restrictions
c---------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     integer iProjVar,jProjVar,NProjVar,LowActRtr(NProjVarMax),
    +       HghActRtr(NProjVarMax)
c
     real*8 Hess(NProjVarMax,NProjVarMax),GradObFct(NProjVarMax),
    +      HessAdj(NProjVarMax,NProjVarMax),GradObFctAdj(NProjVarMax)
c
c-----------------------------------------------------------------
c    HessAdj AND GradObFctAdj WILL CHANGE BASED ON THE CURRENT
c    MINIMIZATION SUBSPACE
```

```
c-----------------------------------------------------------------
c
      do 1 iProjVar=1,NProjVar
c
      do 2 jProjVar=1,NProjVar
c
    HessAdj(iProjVar,jProjVar)=Hess(iProjVar,jProjVar)
c
  2             continue
c
    GradObFctAdj(iProjVar)=GradObFct(iProjVar)
c
  1             continue
c
c-----------------------------------------------------------------
c
      do 3 iProjVar=1,NProjVar
c
    if ((LowActRtr(iProjVar).eq.1).or.
     +   (HghActRtr(iProjVar).eq.1)) then
c
      do 4 jProjVar=1,NProjVar
c
    HessAdj(iProjVar,jProjVar)=0
    HessAdj(jProjVar,iProjVar)=0
c
  4             continue
c
    GradObFctAdj(iProjVar)=0
c
    endif
c
  3             continue
c
      end subroutine
c
c**** HessFctGrdAdj ends here ***********************************************
c*****************************************************************************


c*****************************************************************************
      subroutine HessObjFctEval(NProjVarMax,ProjVar,
     +                          Hess)
c*****************************************************************************
c     BEGIN PROLOGUE  HessObjFctEval
```

```
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Evaluation of the Hessian of the object function on xk
c-----------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      real*8  ProjVar(NProjVarMax),X1,X2,X3,X4,X5,
     +          Hess(NProjVarMax,NProjVarMax)
c
c-----------------------------------------------------------------
c     RESET VARIABLES
c-----------------------------------------------------------------
      Include 'UserHessFctInfo.i'
c
      end subroutine
c
c**** HessObjFctEval ends here ***********************************************
c*****************************************************************************

c*****************************************************************************
      subroutine IniProjVar  (NProjVarMax,NProjVar,LowBound,HighBound,
     +                  Hess,GradObFct,
     +                  ProjVar)
c*****************************************************************************
c     BEGIN PROLOGUE  IniProjVar
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer: Eduardo Perez
c     Purpose: Determine the suitable initial value to project variables
c-----------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer NProjVar,iProjVar,jProjVar,OutBound,iterTotal,yes
c
      real*8 Hess(NProjVarMax,NProjVarMax),HessAbsRowSum(NProjVarMax),
     +      NormInfHess,ProjVarAux(NProjVarMax),ProjVar(NProjVarMax),
     +      GradObFct(NProjVarMax),LowBound(NProjVarMax),
     +      HighBound(NProjVarMax)
c
c-----------------------------------------------------------------------------------------------
c
c*****************************************************************************
```

```
c    SUM OF THE ABSOLUTE VALUES OF THE ROW ELEMENTS OF HESSIAN MATRIX
c**********************************************************************
c
     do 1 iProjVar=1,NProjVar
c
   HessAbsRowSum(iProjVar)=0
c
     do 5 jProjVar=1,NProjVar
c
   HessAbsRowSum(iProjVar)=HessAbsRowSum(iProjVar)+
    +                      ABS(Hess(iProjVar,jProjVar))
c
 5            continue
c
 1            continue
c
c**********************************************************************
c    NORM INFINITY OF THE HESSIAN MATRIX ON xk
c**********************************************************************
c
   NormInfHess=HessAbsRowSum(1)
c
     do 10 iProjVar=1,NProjVar
c
   if (HessAbsRowSum(iProjVar).gt.NormInfHess) then
c
   NormInfHess=HessAbsRowSum(iProjVar)
c
   end if
c
 10           continue
c
c**********************************************************************
c    DETERMINATION OF THE SUITABLE VALUE OF THE PROJECT VARIABLES
c**********************************************************************
c
     do 15 iProjVar=1,NProjVar
c
     ProjVarAux(iProjVar)=ProjVar(iProjVar)-
    +                GradObFct(iProjVar)/NormInfHess
c
 15           continue
c
c**********************************************************************
c    CHECK IF x(k) IS INSIDE THE BONDARIES, IF NOT, PROJECT IT ON THE
```

```
c    BONDARIES
c*********************************************************************
c
     call BoundChk(NProjVarMax,NProjVar,LowBound,HighBound,
    +              ProjVarAux,
    +              OutBound)
c
     if (OutBound.eq.1) then
c
     call PrjOnBound(NProjVarMax,NProjVar,Hess,GradObFct,
    +               LowBound,HighBound,
    +               ProjVar,ProjVarAux,iterTotal,yes)
c
     end if
c
c*********************************************************************
c    UPDATE INITIAL VALUE OF PROJECT VARIABLES
c*********************************************************************
c
     do 20 iProjVar=1,NProjVar
c
     ProjVar(iProjVar)=ProjVarAux(iProjVar)
c
  20           continue
c
c*********************************************************************
c
     end subroutine
c
c**** IniProjVar ends here ***************************************************
c*********************************************************************


c*********************************************************************
     subroutine LinSearch(NProjVarMax,NProjVar,ProjVar,OutBound,
    +                     ProjVarNew,theta)
c*********************************************************************
c    BEGIN PROLOGUE  LinSearch
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer:Eduardo Perez
c    Purpose: Linear Search in direction specified
c---------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
```

```
c
      integer iProjVar,NProjVar,OutBound
c
      real*8 ProjVar(NProjVarMax),ProjVarNew(NProjVarMax),theta
c
c-----------------------------------------------------------------------
c
    theta = 0.9
c
      if (OutBound.eq.1) then
c
      do 1 iProjVar=1,NProjVar
c
                ProjVarNew(iProjVar)=ProjVar(iProjVar)+
     +                    theta*(ProjVarNew(iProjVar)-ProjVar(iProjVar))
c
  1             continue
c
    endif
c
      endsubroutine
c
c**** LinSearch ends here ***************************************************
c***********************************************************************


c***********************************************************************
      subroutine MinMaxBndRng(NProjVarMax,NProjVar,LowBound,HighBound,
     +                    MinBoundRange,MaxBoundRange)
c***********************************************************************
c     BEGIN PROLOGUE  MinBndRng
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer: Eduardo Perez
c     Purpose: Determine the narrower boundary range of project variables
c-----------------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer NProjVar,iProjVar
c
      real*8  BoundRange,MinBoundRange,MaxBoundRange,
     +          LowBound(NProjVarMax),HighBound(NProjVarMax)
c-----------------------------------------------------------------------
c
```

```
   BoundRange=0
c
     MinBoundRange =HighBound(1)-LowBound(1)
     MaxBoundRange =HighBound(1)-LowBound(1)

c
     do 1 iProjVar=1,NProjVar
c
               BoundRange =HighBound(iProjVar)-LowBound(iProjVar)
c
   if (BoundRange.lt.MinBoundRange) then
c
   MinBoundRange=BoundRange
c
   else if (BoundRange.gt.MaxBoundRange) then
c
   MaxBoundRange=BoundRange
c
   end if
c
  1            continue
c
      end
c
c**** MinMaxBndRng ends here ****************************************
c*********************************************************************


c*********************************************************************
      subroutine NegPrjGradCalc(NProjVarMax,NProjVar,LowActRtr,
     +                    HghActRtr,GradQuad,
     +                    NegPrjGrad)
c*********************************************************************
c     BEGIN PROLOGUE  NegPrjGradCalc
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Calculation of the negative projected gradient          *
c---------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,NProjVar,LowActRtr(NProjVarMax),
     +       HghActRtr(NProjVarMax)
c
```

```
      real*8 GradQuad(NProjVarMax,2),NegPrjGrad(NProjVarMax)
c
c*********************************************************************
c     CALCULATION OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
      do 1  iProjVar=1,NProjVar
c
   if ((LowActRtr(iProjVar).eq.1).and.
     +   (GradQuad(iProjVar,1).gt.0)) then
c
   NegPrjGrad(iProjVar)=0
c
   elseif ((HghActRtr(iProjVar).eq.1).and.
     +       (GradQuad(iProjVar,1).lt.0)) then
c
   NegPrjGrad(iProjVar)=0
c
   else
c
   NegPrjGrad(iProjVar)=-GradQuad(iProjVar,1)
c
      endif
c
  1   continue
c
      endsubroutine
c
c**** NegPrjGradCalc ends here ********************************************
c*********************************************************************


c*********************************************************************
      subroutine NegPrjGradObFct(NProjVarMax,NProjVar,LowActRtr,
     +                    HghActRtr,GradObFct,
     +                    NegPrjGrad)
c*********************************************************************
c     BEGIN PROLOGUE  NegPrjGradObFct
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Calculation of the negative projected gradient
c---------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
```

```
c
      integer iProjVar,NProjVar,LowActRtr(NProjVarMax),
     +        HghActRtr(NProjVarMax)
c
      real*8 GradObFct(NProjVarMax),NegPrjGrad(NProjVarMax)
c
c*********************************************************************
c     CALCULATION OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
      do 1  iProjVar=1,NProjVar
c
   if ((LowActRtr(iProjVar).eq.1).and.
     +   (GradObFct(iProjVar).gt.0)) then
c
   NegPrjGrad(iProjVar)=0
c
   elseif ((HghActRtr(iProjVar).eq.1).and.
     +       (GradObFct(iProjVar).lt.0)) then
c
   NegPrjGrad(iProjVar)=0
c
   else
c
   NegPrjGrad(iProjVar)=-GradObFct(iProjVar)
c
      endif
c
  1   continue
c
      end subroutine
c
c**** NegPrjGradObFct ends here ***********************************************
c*********************************************************************


c*********************************************************************
      subroutine ObjFctEval(NProjVarMax,ProjVar,
     +                  F)
c*********************************************************************
c     BEGIN PROLOGUE  ObjFctEval
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Evaluation of the object function on xk
```

```
c-----------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     real*8  ProjVar(NProjVarMax),X1,X2,X3,X4,X5,F
c
c-------------------------------------------------------------
c    RESET VARIABLES
c-------------------------------------------------------------
c
     Include 'UserFctInfo.i'
c
     end subroutine
c
c**** ObjFctEval ends here ***************************************************
c*******************************************************************************
```



```
c*******************************************************************************
     subroutine Omega(NProjVarMax,NProjVar,TrtRegLowBound,
    +               TrtRegHighBound,LowBound,HighBound,
    +                        LowBoundOmega,HighBoundOmega)
c*******************************************************************************
c    BEGIN PROLOGUE  Omega
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer: Eduardo Perez
c    Purpose: Determine the intersection between trust region and
c           project variables range
c-----------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     integer NProjVar,iProjVar
c
     real*8  LowBound(NProjVarMax),HighBound(NProjVarMax),
    +       TrtRegLowBound(NProjVarMax),TrtRegHighBound(NProjVarMax),
    +       LowBoundOmega(NProjVarMax),HighBoundOmega(NProjVarMax)
c
c*******************************************************************
c    DETERMINATION OF THE BOUNDARIES OF OMEGA,THE INTERSECTION BETWEEN
c    TRUST REGION AND PROJECT VARIABLES RANGE
```

```
c*******************************************************************
c
     do 1 iProjVar=1,NProjVar
c
   LowBoundOmega(iProjVar)=LowBound(iProjVar)
   HighBoundOmega(iProjVar)=HighBound(iProjVar)
c
  1             continue
c-------------------------------------------------------------------
c
     do 10 iProjVar=1,NProjVar
c
   if (LowBound(iProjVar).lt.TrtRegLowBound(iProjVar)) then
c
   LowBoundOmega(iProjVar)=TrtRegLowBound(iProjVar)
c
   end if
c
   if (HighBound(iProjVar).gt.TrtRegHighBound(iProjVar)) then
c
   HighBoundOmega(iProjVar)=TrtRegHighBound(iProjVar)
c
   end if
c
  10           continue
c
     end subroutine
c
c**** Omega ends here ***********************************************
c*******************************************************************


c*******************************************************************
     subroutine OpProjVar(NProjVarMax,NProjVar,ProjVarY,
    +                     ProjVarX)
c*******************************************************************
c    BEGIN PROLOGUE OpProjVar
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer: Eduardo Perez
c    Purpose: Do ProjVarX=ProjVarY
c-------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
```

```
      integer NProjVar,iProjVar
c
      real*8  ProjVarY(NProjVarMax),ProjVarX(NProjVarMax)
c-----------------------------------------------------------------------
c
      do 1 iProjVar=1,NProjVar
c
      ProjVarX(iProjVar)=ProjVarY(iProjVar)
c
 1            continue
c
      end subroutine
c
c**** OpProjVar ends here ***************************************************
c*****************************************************************************


c*****************************************************************************
      subroutine PrjOnBound(NProjVarMax,NProjVar,Hess,GradObFct,
     +                  LowBound,HighBound,
     +                  ProjVar,ProjVarNew,iterTotal,yes)
c
c*****************************************************************************
c     BEGIN PROLOGUE  PrjOnBound
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Orthogonal projection of external points on project
c           variables boundary
c-----------------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,jProjVar,NProjVar,iterTotal,yes,count
c
      real*8 Hess(NProjVarMax,NProjVarMax),ProjVar(NProjVarMax),
     +      ProjVarNew(NProjVarMax),GradObFct(NProjVarMax),
     +      LowBound(NProjVarMax),HighBound(NProjVarMax),
     +      ProjVarTHessProjVar,ProjVarNewTHessProjVarNew,
     +      GradObFctTProjVar,GradObFctTProjVarNew,
     +      HessProjVar(NProjVarMax),HessProjVarNew(NProjVarMax),
     +      Quad,QuadNew,theta,ProjVarAux(NProjVarMax)
c
c-----------------------------------------------------------------------
c
```

```
   theta=1
   flag=1
   yes=1
   count=0
c
   do while (flag.eq.1)
c
     do 1 iProjVar=1,NProjVar
c
               ProjVarAux(iProjVar)=ProjVar(iProjVar)+
    +                    theta*(ProjVarNew(iProjVar)-ProjVar(iProjVar))
c
   if (ProjVarAux(iProjVar).lt.LowBound(iProjVar)) then
c
   ProjVarAux(iProjVar)=LowBound(iProjVar)
c
   elseif (ProjVarAux(iProjVar).gt.HighBound(iProjVar)) then
c
   ProjVarAux(iProjVar)=HighBound(iProjVar)
c
   endif
c
  1            continue
c
c**********************************************************************
c    CALCULATION OF THE VALUE OF THE QUADRATIC FUNCTION ON xk AND x(k+1),
c    q(x(k+1)) MUST BE LOWER THAN q(xk)
c**********************************************************************
c
   ProjVarTHessProjVar=0
   ProjVarNewTHessProjVarNew=0
   GradObFctTProjVar=0
   GradObFctTProjVarNew=0
c
     do 2 iProjVar=1,NProjVar
c
   do 3 jProjVar=1,NProjVar
c
   HessProjVar(iProjVar)=Hess(iProjVar,jProjVar)*
    +                  ProjVar(jProjVar)+HessProjVar(iProjVar)
c
   HessProjVarNew(iProjVar)=Hess(iProjVar,jProjVar)*
    +                ProjVarAux(jProjVar)+HessProjVarNew(iProjVar)
c
  3   continue
```

```
c
   ProjVarTHessProjVar=ProjVarTHessProjVar +
    +                 ProjVar(iProjVar)*HessProjVar(iProjVar)
c
   ProjVarNewTHessProjVarNew=ProjVarNewTHessProjVarNew +
    +                 ProjVarAux(iProjVar)*HessProjVarNew(iProjVar)
c
   GradObFctTProjVar=GradObFctTProjVar+GradObFct(iProjVar)*
    +                 ProjVar(iProjVar)
c
   GradObFctTProjVarNew=GradObFctTProjVarNew+GradObFct(iProjVar)*
    +                 ProjVarAux(iProjVar)
c
 2   continue
c
c---------------------------------------------------------------------
c     RESET VARIABLES
c---------------------------------------------------------------------
c
      do 4 iProjVar=1,NProjVar
c
   HessProjVar(iProjVar)=0
   HessProjVarNew(iProjVar)=0
c
 4   continue
c
c---------------------------------------------------------------------
c
   Quad=(0.5)*ProjVarTHessProjVar+GradObFctTProjVar
   QuadNew=(0.5)*ProjVarNewTHessProjVarNew+GradObFctTProjVarNew
c
   if (QuadNew.lt.Quad) then
c
      do 5 iProjVar=1,NProjVar
c
      ProjVarNew(iProjVar)=ProjVarAux(iProjVar)
c
 5   continue
c
   flag=0
c
   else
c
   theta=theta*0.8
c
```

```
c-----------------------------------------------------------------
c     WAY TO GET OUT OF THE LOOP IF QuadNew NEVER IS LESS THAN Quad
c-----------------------------------------------------------------
c
   count=count+1
c
                if (count.gt.20) then
c
      do 10 iProjVar=1,NProjVar
c
      ProjVarNew(iProjVar)=ProjVarAux(iProjVar)
c
  10  continue
c
   goto 1000
c
   end if
c
c-----------------------------------------------------------------
c
      end if
c
   end do
c
   iterTotal=iterTotal-1
c
 1000 end subroutine
c
c**** PrjOnBound ends here ***************************************************
c*******************************************************************************


c*******************************************************************************
      subroutine ProjVarLessProjVarOld(NProjVarMax,NProjVar,ProjVar,
     +          ProjVarOld,LowBoundOmega,HighBoundOmega,
     +    QuadProjVar,QuadLowBoundOmega,QuadHighBoundOmega,TransFactor)
c*******************************************************************************
c     BEGIN PROLOGUE ProjVarLessProjVarOld
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer: Eduardo Perez
c     Purpose: Do QuadProjVar=ProjVar-ProjVarOld and define the
c              boundaries of QuadProjVar
c-------------------------------------------------------------------------------------------------
c
```

```
c    INTERNAL & EXTERNAL ARRAYS
c
     integer NProjVar,iProjVar
c
     real*8  ProjVar(NProjVarMax),ProjVarOld(NProjVarMax),
    +       QuadProjVar(NProjVarMax),TransFactor(NProjVarMax),
    +      QuadLowBoundOmega(NProjVarMax),LowBoundOmega(NProjVarMax),
    +            QuadHighBoundOmega(NProjVarMax),HighBoundOmega(NProjVarMax)
c
c**********************************************************************
c               DEFINE THE NEW VARIABLE (x-xk)
c**********************************************************************
c
     do 1 iProjVar=1,NProjVar
c
     QuadProjVar(iProjVar)=ProjVar(iProjVar)-ProjVarOld(iProjVar)
c
 1             continue
c

c**********************************************************************
c               DEFINE THE BOUNDARIES OF THE TRUST REGION OF (x-xk)
c**********************************************************************
c
     do 5 iProjVar=1,NProjVar
c
   TransFactor(iProjVar)=ProjVar(iProjVar)-2*ProjVarOld(iProjVar)
c
     QuadLowBoundOmega(iProjVar)=LowBoundOmega(iProjVar)+
    +                      TransFactor(iProjVar)
c
     QuadHighBoundOmega(iProjVar)=HighBoundOmega(iProjVar)+
    +                       TransFactor(iProjVar)
c
 5             continue
c
     end subroutine
c
c**** ProjVarLessProjVarOld ends here ************************************
c**********************************************************************


c**********************************************************************
     subroutine QuadBoundMin(NProjVarMax,NProjVar,Hess,GradObFct,
    +              LowBound,HighBound,MinBoundRange,MaxBoundRange,
```

```
     +              ProjVar,iterTotal)
c**********************************************************************
c    BEGIN PROLOGUE  QuadBoundMin
c    Date Written:  18 Oct 2007
c    Revision Date:
c    Programmer:Eduardo Perez
c    Purpose: Quadratic Bounded Minimization using Conjugate Gradient
c           Method. It is a subproblem of the bounded minimization of a general function.
c           Quadratic to be solved is:
c           q(u)=(1/2)uT*H*u+bT*u
c           where H=hessian of objective function                *
c                b=gradient of objective function
c-----------------------------------------------------------------------------------------------------
c
c    INTERNAL & EXTERNAL ARRAYS
c
     integer iProjVar,jProjVar,NProjVar,iter,LowActRtr(NProjVarMax),
     +       HghActRtr(NProjVarMax),LowActRtrNew(NProjVarMax),
     +       HghActRtrNew(NProjVarMax),RtrChgFlag,OutBound,iterTotal,
     +       yes
c
     real*8  GradQuad(NProjVarMax,2),GradQuadAdj(NProjVarMax,2),
     +          Hess(NProjVarMax,NProjVarMax),neta,
     +       ProjVar(NProjVarMax),ProjVarNew(NProjVarMax),
     +       GradObFct(NProjVarMax),StepDir(NProjVarMax),
     +       StepDirNew(NProjVarMax),GradNorm,
     +       dkTGrad,StepTHessStep,StepLenght,dkTGradNew,
     +       BetaStepLenght,LowBound(NProjVarMax),
     +       HighBound(NProjVarMax),HessAdj(NProjVarMax,NProjVarMax),
     +       GradObFctAdj(NProjVarMax),NegPrjGrad(NProjVarMax),
     +       ChpGrad(NProjVarMax),ChpGradNorm,MinBoundRange,
     +       MaxBoundRange,NegCurvStpLength,StepDirNorm,theta,NPv
c
c-----------------------------------------------------------------
c
     neta=0.8
     iterTotal=0
c
c**********************************************************************
c    CHECK FOR INITIAL ACTIVE RESTRICTIONS
c**********************************************************************
c
     call ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,ProjVar,
     +          LowActRtr,HghActRtr)
```

```
c
c*********************************************************************
c    ADJUST OF HESSIAN AND OBJECTIVE FUNCTION GRADIENT TO MINIMIZATION
c    SUBSPACE
c*********************************************************************
c
 1000 call HessFctGrdAdj(NProjVarMax,NProjVar,Hess,GradObFct,
     +              LowActRtr,HghActRtr,
     +              HessAdj,GradObFctAdj)
c
c*********************************************************************
c    CALCULATION OF THE QUADRATIC FUNCTION GRADIENT g=Hx+b IN THE
c    FULL MINIMIZATION SPACE AND IN THE ACTUAL MINIMIZATION SPACE
c*********************************************************************
c
     call FstGradQuad(NProjVarMax,NProjVar,ProjVar,Hess,
     +                   GradObFct,LowActRtr,HghActRtr,
     +             GradQuadAdj,GradQuad)
c
c*********************************************************************
c    CALCULATION OF THE CHOPED GRADIENT
c*********************************************************************
c
   call ChpGradCalc(NProjVarMax,NProjVar,LowActRtr,
     +              HghActRtr,GradQuad,
     +              ChpGrad)
c
c*********************************************************************
c    CALCULATION OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
     call NegPrjGradCalc(NProjVarMax,NProjVar,LowActRtr,
     +               HghActRtr,GradQuad,
     +               NegPrjGrad)

c*********************************************************************
c    NORM OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
     GradNorm = dnrm2(NProjVar, NegPrjGrad, 1)
c
c*********************************************************************
c    NORM OF THE CHOPED GRADIENT
c*********************************************************************
c
```

```
      ChpGradNorm = dnrm2(NProjVar,ChpGrad, 1)
c
c*********************************************************************
c     IF NORM OF THE CHOPED GRADIENT GREATER THAN NETA*(NORM OF THE
c     NEGATIVE PROJECTED GRADIENT), MINIMIZE ON CHOPED GRADIENT DIRECTION
c*********************************************************************
c
  500         if (ChpGradNorm.gt.(neta*GradNorm))then
c
      call GoChpGradDir(NProjVarMax,NProjVar,Hess,ChpGrad,
     +              ChpGradNorm,ProjVar,LowBound,
     +              HighBound,theta,MinBoundRange,
     +              LowActRtr,HghActRtr,ProjVarNew,iterTotal)
c
    goto 1000
c
    else        ! CHOPED GRADIENT IS NOT GRATER THAN NETA*PROJECTED GRAD
c
c*********************************************************************
c     DIRECTION OF THE FIRST MINIMIZATION  d0=-g0
c*********************************************************************
c
      do 10 iProjVar=1,NProjVar
c
    StepDir(iProjVar)=-GradQuadAdj(iProjVar,1)
c
  10          continue
c
      iter=0
c
c*********************************************************************
c      CALCULATION OF -dk'g TO BE USED ON DETERMINATION OF STEPLENGHT
c*********************************************************************
c
    dkTGrad=0
c
      do 15 iProjVar = 1,NProjVar
c
    dkTGrad=dkTGrad+
     +       (-1)*StepDir(iProjVar)*GradQuad(iProjVar,1)
c
  15  continue
c
c*********************************************************************
c     CALCULATION OF dk'Hdk: IF POSITIVE,PROCEED TO CONJUGATE GRADIENT
```

```
c    METHOD, IF NOT, PROCEED IN dk DIRECTION UNTIL REACH BOUNDARY
c*********************************************************************
c
     call DirCurvature(NProjVarMax,NProjVar,Hess,StepDir,
    +                  StepTHessStep)
c
c*********************************************************************
c    BEGIN OF LOOP - CONJUGATE GRADIENT METHOD
c*********************************************************************
c
     do while ((GradNorm>0.000001).and.(StepTHessStep>0).and.
    +         (iter.lt.10))
c
   StepLenght=dkTGrad/(StepTHessStep)
c
   dkTGradNew=0
c
c*********************************************************************
c    CALCULATION OF x(k+1)=x(k)+a*dk
c*********************************************************************
c
     do 20 iProjVar=1,NProjVar
c
   ProjVarNew(iProjVar)=ProjVar(iProjVar)+
    +                       StepLenght*StepDir(iProjVar)
c
  20           continue
c
               iterTotal=iterTotal+1
c
c*********************************************************************
c    CHECK IF x(k+1) IS INSIDE THE BONDARIES, IF NOT, PROJECT IT ON THE
c    BONDARIES
c*********************************************************************
c
     call BoundChk(NProjVarMax,NProjVar,LowBound,HighBound,
    +              ProjVarNew,
    +              OutBound)
c
     if (OutBound.eq.1) then

     call PrjOnBound(NProjVarMax,NProjVar,Hess,GradObFct,
    +                LowBound,HighBound,
    +                ProjVar,ProjVarNew,iterTotal,yes)
c
```

```
      end if
c
c*********************************************************************
c     UPDATE ACTIVE RESTRICTIONS
c*********************************************************************
c
      call ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,ProjVarNew,
     +            LowActRtrNew,HghActRtrNew)
c
c*********************************************************************
c     CHECK IF ACTIVE RESTRICTIONS CHANGED, IF NOT, CONTINUE MINIMIZATION
c     IN THE SAME SUBSPACE, IF YES, CHANGE MINIMIZATION SUBSPACE
c*********************************************************************
c
      call ActRestrChg(NProjVarMax,NProjVar,LowActRtr,HghActRtr,
     +            LowActRtrNew,HghActRtrNew,
     +            RtrChgFlag)
c
    if (RtrChgFlag.eq.1) then
c
      do 25 jProjVar=1,NProjVar
c
      ProjVar(jProjVar)=ProjVarNew(jProjVar)
    LowActRtr(jProjVar)=LowActRtrNew(jProjVar)
    HghActRtr(jProjVar)=HghActRtrNew(jProjVar)
c
   25         continue
c
    if (yes.eq.1) then
c
    iterTotal=iterTotal+1
c
    end if
c
    yes=0
c
    goto 1000
c
    end if
c
c*********************************************************************
c     CALCULATION OF g(k+1)=H*x(k+1)+b IN THE FULL MINIMIZATION
c     SPACE
c*********************************************************************
c*********************************************************************
```

```
c    CALCULATION OF g(k+1)=g(k)+a*H*dk IN THE ACTUAL MINIMIZATION
c    SUBSPACE
c*********************************************************************
c
     call GradQuadNew(NProjVarMax,NProjVar,ProjVarNew,Hess,HessAdj,
    +              StepDir,StepLenght,GradObFct,
    +              GradQuadAdj,GradQuad,dkTGradNew)
c
c*********************************************************************
c
   BetaStepLenght=dkTGradNew/dkTGrad
c
   dkTGrad=dkTGradNew
c
c*********************************************************************
c    NEW DIRECTION TO MINIMIZATION d(k+1)=-g(k+1)+B*dk
c*********************************************************************

     do 30 iProjVar=1,NProjVar
c
   StepDirNew(iProjVar)=-GradQuadAdj(iProjVar,2)+
    +                    BetaStepLenght*StepDir(iProjVar)
c
 30          continue
c
c*********************************************************************
c    VARIABLES UPDATE FOR THE NEXT ITERACTION
c*********************************************************************
c
     do 35 iProjVar=1,NProjVar
c
   StepDir(iProjVar)=StepDirNew(iProjVar)
     ProjVar(iProjVar)=ProjVarNew(iProjVar)
   GradQuad(iProjVar,1)=GradQuad(iProjVar,2)
             GradQuadAdj(iProjVar,1)=GradQuadAdj(iProjVar,2)
c
 35          continue
c
c*********************************************************************
c    CALCULATION OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
     call NegPrjGradCalc(NProjVarMax,NProjVar,LowActRtr,
    +              HghActRtr,GradQuad,
    +              NegPrjGrad)
```

```
c
c*********************************************************************
c    CALCULATION OF THE CHOPED GRADIENT
c*********************************************************************
c
  call ChpGradCalc(NProjVarMax,NProjVar,LowActRtr,
   +              HghActRtr,GradQuad,
   +              ChpGrad)
c
c*********************************************************************
c    NORM OF THE NEGATIVE PROJECTED GRADIENT
c*********************************************************************
c
    GradNorm = dnrm2(NProjVar,NegPrjGrad, 1)
c
c*********************************************************************
c    NORM OF THE CHOPED GRADIENT
c*********************************************************************
c
    ChpGradNorm = dnrm2(NProjVar,ChpGrad, 1)
c
c*********************************************************************
c    IF NORM OF THE CHOPED GRADIENT GREATER THAN NETA*(NORM OF THE
c    NEGATIVE PROJECTED GRADIENT), MINIMIZE ON CHOPED GRADIENT DIRECTION
c*********************************************************************
c
  if (ChpGradNorm.gt.(neta*GradNorm))then
c
    call GoChpGradDir(NProjVarMax,NProjVar,Hess,ChpGrad,
   +              ChpGradNorm,ProjVar,LowBound,
   +              HighBound,theta,MinBoundRange,
   +              LowActRtr,HghActRtr,ProjVarNew,iterTotal)
c
  goto 1000
c
  else        ! CHOPED GRADIENT IS NOT GRATER THAN NETA*PROJECTED GRAD
c
c*********************************************************************
c    CALCULATION OF dk'Hdk: IF POSITIVE, CONTINUE ON CONJUGATE GRADIENT
c    METHOD, IF NOT, PROCEED IN dk DIRECTION UNTIL REACH BOUNDARY
c*********************************************************************
c
    call DirCurvature(NProjVarMax,NProjVar,Hess,StepDir,
   +              StepTHessStep)
c
```

```
   iter=iter+1
c
   endif
c
     end do
c
c**********************************************************************
c    END OF LOOP - CONJUGATE GRADIENT METHOD
c**********************************************************************
c
     end if ! END IF 500
c
c**********************************************************************
c    IF CURVATURE IS NEGATIVE, GO IN dk DIRECTION UNTIL REACH THE
c    BOUNDARY
c**********************************************************************
c
     if ((StepTHessStep.le.0).and.(GradNorm.gt.0.000001)) then
c
c**********************************************************************
c    CALCULATION OF x(k+1)=x(k)+factor*t*dk. THE FACTOR IS TO MAKE
c    SURE THAT x(k+1) WILL BE OUT OF THE BOUNDARIES
c**********************************************************************
c
     StepDirNorm = dnrm2(NProjVar,StepDir, 1)
   NPv=NProjvar
c
     do 55 iProjVar=1,NProjVar
c
   NegCurvStpLength=MaxBoundRange/StepDirNorm
c
     ProjVarNew(iProjVar)=ProjVar(iProjVar)+
    +                    sqrt(NPv)*NegCurvStpLength*StepDir(iProjVar)
c
 55          continue
c
   iterTotal=iterTotal+1
c
c**********************************************************************
c    CHECK IF x(k+1) IS INSIDE THE BONDARIES, IF NOT, PROCEED LINEAR
c    SEARCH TO BRING IT INSIDE BONDARIES
c**********************************************************************
c
   OutBound=1
c
```

```
   do while (OutBound.eq.1)
c
     call BoundChk(NProjVarMax,NProjVar,LowBound,HighBound,
    +            ProjVarNew,
    +            OutBound)
c
c*********************************************************************
c    LINEAR SEARCH TO BRING x(k+1) INSIDE BONDARIES
c*********************************************************************
c
   if (OutBound.eq.1) then
c
     call LinSearch(NProjVarMax,NProjVar,ProjVar,OutBound,
    +             ProjVarNew,theta)
c
   end if
c
     end do
c
c*********************************************************************
c   ONE STEP BACK TO BRING x(k+1) SLIGHTLY OUT OF THE BOUNDARIES
c*********************************************************************
c
     do 60 iProjVar=1,NProjVar
c
   ProjVarNew(iProjVar)=ProjVar(iProjVar)+
    +                   (ProjVarNew(iProjVar)-ProjVar(iProjVar))/theta
c
  60           continue
c
     call PrjOnBound(NProjVarMax,NProjVar,Hess,GradObFct,
    +              LowBound,HighBound,
    +              ProjVar,ProjVarNew,iterTotal,yes)
c
c*********************************************************************
c    UPDATE ACTIVE RESTRICTIONS
c*********************************************************************
c
     call ActRestr(NProjVarMax,NProjVar,LowBound,HighBound,
    +            ProjVarNew,
    +            LowActRtr,HghActRtr)
c
c*********************************************************************
c    UPDATE PROJECT VARIABLES
c*********************************************************************
```

```
c
      do 65 iProjVar=1,NProjVar
c
      ProjVar(iProjVar)=ProjVarNew(iProjVar)
c
 65           continue
c
      iterTotal=iterTotal+1
c
      goto 1000
c
      end if
c
c**********************************************************************
c     END OF x(k+1) CALCULATION
c**********************************************************************
c
c**********************************************************************
c
      endsubroutine
c
c**** QuadBoundMin ends here ***********************************************
c**********************************************************************



c**********************************************************************
      subroutine QuadValue (NProjVarMax,NProjVar,Hess,GradObFct,
     +                  QuadProjVar,ObjFctIni,
     +                  Quad)
c**********************************************************************
c     BEGIN PROLOGUE  QuadValue
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer:Eduardo Perez
c     Purpose: Value of the aproximating quadratic function on x(k+1)
c  q(x(k+1)-xk)=(1/2)*(x(k+1)-xk)T*H*(x(k+1)-xk)+bT*(x(k+1)-xk)+f(xk)                *
c-----------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer iProjVar,jProjVar,NProjVar
c
      real*8 Hess(NProjVarMax,NProjVarMax),
     +      GradObFct(NProjVarMax),ProjVarTHessProjVar,
     +      GradObFctTProjVar,HessProjVar(NProjVarMax),Quad,
```

```
     +      QuadProjVar(NProjVarMax),ObjFctIni
c
c-----------------------------------------------------------------
c     RESET VARIABLES
c-----------------------------------------------------------------
c
      do 1 iProjVar=1,NProjVar
c
   HessProjVar(iProjVar)=0
c
  1   continue
c
c**********************************************************************
c     CALCULATION OF THE VALUE OF THE QUADRATIC FUNCTION ON x(k+1)
c**********************************************************************
c
   ProjVarTHessProjVar=0
   GradObFctTProjVar=0
c
      do 5 iProjVar=1,NProjVar
c
   do 10 jProjVar=1,NProjVar
c
   HessProjVar(iProjVar)=Hess(iProjVar,jProjVar)*
     +               QuadProjVar(jProjVar)+HessProjVar(iProjVar)
c
  10   continue
c
   ProjVarTHessProjVar=ProjVarTHessProjVar +
     +                QuadProjVar(iProjVar)*HessProjVar(iProjVar)
c
   GradObFctTProjVar=GradObFctTProjVar+GradObFct(iProjVar)*
     +               QuadProjVar(iProjVar)
c
  5   continue
c
c-----------------------------------------------------------------
c
   Quad=(0.5)*ProjVarTHessProjVar+GradObFctTProjVar+ObjFctIni
c
c-----------------------------------------------------------------
c
      end subroutine
c
c**** QuadValue ends here ***************************************************
```

```
c*******************************************************************

c*******************************************************************
      subroutine TrtRegBound (NProjVarMax,NProjVar,ProjVar,TrtRegSize,
     +                        TrtRegLowBound,TrtRegHighBound)
c*******************************************************************
c     BEGIN PROLOGUE  TrtRegBound
c     Date Written:  18 Oct 2007
c     Revision Date:
c     Programmer: Eduardo Perez
c     Purpose: Determine the trust region boundaries
c-------------------------------------------------------------------------------------------------------
c
c     INTERNAL & EXTERNAL ARRAYS
c
      integer NProjVar,iProjVar
c
      real*8  TrtRegLowBound(NProjVarMax),TrtRegHighBound(NProjVarMax),
     +        ProjVar(NProjVarMax),TrtRegSize
c
c-----------------------------------------------------------------------
c
c*******************************************************************
c     DETERMINATION OF THE BOUNDARIES OF THE TRUST REGION
c*******************************************************************
c
      do 1 iProjVar=1,NProjVar
c
                TrtRegLowBound(iProjVar)=ProjVar(iProjVar)-TrtRegSize
                TrtRegHighBound(iProjVar)=ProjVar(iProjVar)+TrtRegSize
c
  1             continue
c
      end subroutine
c
c**** TrtRegBound ends here ***********************************************
c*******************************************************************

c*******************************************************************
c     Include file UserFctInfo
c*******************************************************************
c     BEGIN PROLOGUE
c     Date Written:  27 Oct 2007
```

```
c     Revision Date: 23 Sep 2008
c     Programmer:
c           Eduardo Perez
c     Purpose:
c-----------------------------------------------------------------
c     DESCRIPTION
c     USER routine
c     It is problem dependent.
c
c     Evaluate the an analitic objective function on xK
c
c     REFERENCES                                    *
c                                                   *
c     END PROLOGUE START                            *
c**********************************************************************
c
c**********************************************************************
c     ATRIBUTE VARIABLES
c**********************************************************************
c
c     X1=ProjVar(1)
c    X2=ProjVar(2)
c    X3=ProjVar(3)
c    X4=ProjVar(4)
c    X5=ProjVar(5)
c
c**********************************************************************
c     OBJECT FUNCTION DEFINITION
c**********************************************************************
c
c     F=X1**3+(X2+1)**2+X1*X2
c
c**********************************************************************
c     MANUAL INPUT
c**********************************************************************
c
      F=0.1311
c
c**********************************************************************
c**** UserFctInfo ends here ***************************************************
c**********************************************************************


c**********************************************************************
c     Include file UserGradFctInfo
```

```
c*********************************************************************
c     BEGIN PROLOGUE
c     Date Written:  27 Oct 2007
c     Revision Date:
c     Programmer:
c           Eduardo Perez
c     Purpose:
c---------------------------------------------------------------------------------------------
c     DESCRIPTION
c     USER routine
c     It is problem dependent.
c

c     Evaluate the gradient of the object function on xK
c     REFERENCES
c     END PROLOGUE START
c*********************************************************************
c
c*********************************************************************
c     ATRIBUTE VARIABLES
c*********************************************************************
c     X1=ProjVar(1)
c    X2=ProjVar(2)
c    X3=ProjVar(3)
c    X4=ProjVar(4)
c    X5=ProjVar(5)
c
c*********************************************************************
c     OBJECT FUNCTION DEFINITION
c*********************************************************************
c    GradObFct(1)=3*(X1**2)+X2
c    GradObFct(2)=2*(X2+1)+X1
c
c*********************************************************************
c     MANUAL INPUT
c*********************************************************************
     GradObFct(1)=-9.42d-4
     GradObFct(2)=2.25d-2
c
c*********************************************************************
c**** UserGradFctInfo ends here ***********************************************
c*********************************************************************


c*********************************************************************
```

```
c     Include file UserHessFctInfo
c*************************************************************************
c     BEGIN PROLOGUE
c     Date Written:  27 Oct 2007
c     Revision Date:
c     Programmer:
c           Eduardo Perez
c     Purpose:
c-------------------------------------------------------------------
c     DESCRIPTION
c     USER routine
c     It is problem dependent.
c
c     Evaluate the Hess of the object function on xK
c     REFERENCES
c     END PROLOGUE START
c*************************************************************************
c
c*************************************************************************
c     ATRIBUTE VARIABLES
c*************************************************************************
c     X1=ProjVar(1)
c    X2=ProjVar(2)
c    X3=ProjVar(3)
c    X4=ProjVar(4)
c    X5=ProjVar(5)
c
c*************************************************************************
c     HESSIAN EVALUATION
c*************************************************************************
c     Hess(1,1)=6*X1
c    Hess(1,2)=1
c    Hess(2,1)=1
c    Hess(2,2)=2
c
c*************************************************************************
c     MANUAL INPUT
c*************************************************************************
      Hess(1,1)=0.533d-0
     Hess(1,2)=0.754d-0
     Hess(2,1)=0.754d-0
     Hess(2,2)=1.00d-0
c
c*************************************************************************
c**** UserHessFctInfo ends here ***************************************
```

```
c*******************************************************************************
c*******************************************************************************
c     Include file UserFctInfo
c*******************************************************************************
c     BEGIN PROLOGUE
c     Date Written:  27 Oct 2007
c     Revision Date: 23 Sep 2008
c     Programmer:
c           Eduardo Perez
c     Purpose:
c-------------------------------------------------------------------
c     DESCRIPTION
c     USER routine
c     It is problem dependent.
c     Define manually # of project variables and initial guess.
c     REFERENCES
c
c     END PROLOGUE START
c*******************************************************************************
c
c*******************************************************************************
c     OPTIMIZATION FLAG (0-MANUAL INPUT 1-CFD TIED)
c*******************************************************************************
      OptFlag=0
c
c*******************************************************************************
c     # OF PROJECT VARIABLES
c*******************************************************************************
c
      NProjVar = 2
c
c*******************************************************************************
c     LOW AND HIGH LIMIT FOR THE PROJECT VARIABLES (ONLY POSITIVE #)
c*******************************************************************************

      LowBound(1)=1.2
      LowBound(2)=0.16
c      LowBound(3)=-1
c      LowBound(4)=-1
c      LowBound(5)=-1

      HighBound(1)=4.4
      HighBound(2)=4.333
c      HighBound(3)=1
c      HighBound(4)=1
```

```
c      HighBound(5)=1
c
c*************************************************************************
c     INITIAL GUESS
c*************************************************************************
      ProjVar(1)=2.653
      ProjVar(2)=1.0
c      ProjVar(3)=2
c      ProjVar(4)=-1
c      ProjVar(5)=1
c
c*************************************************************************
c**** UserFctInfo ends here ********************************************
c*************************************************************************
```