```
function [p,jh,l,h,ic]=solnp(pb,ib,op,l,h)
%
if exist('pb')<=0.5,
  disp('Syntax error')
  return
end;
%
om=['SOLNP--> ';'         '];
%
[np,n]=size(pb); lpb=[1 0];
if     n==1,
  p=pb; pb=0; lpb(1)=0;
elseif n==2,
  p=(pb(:,1)+pb(:,2))/2;
elseif n==3,
  p=pb(:,1); pb=pb(:,2:3);
else,
  disp([om(1,:) 'Parameter array must have three columns or less']);
  return
end;
%
if lpb(1)>=0.5,
  if min(pb(:,2)-pb(:,1))<=0,
    disp([om(1,:) 'The lower bounds of the parameter constraints ';...
          om(2,:) 'must be strictly less than the upper bounds.  ']);
    return
  elseif min([p-pb(:,1);pb(:,2)-p])<=0,
    disp([om(1,:) 'Initial parameter values must be within the bounds']);
    return
  end;
end;
%
if exist('ib')<=0.5,...
  ic=0; nic=0;...
else
  [nic, n]=size(ib);
  if n==3,
    ib0=ib(:,1);ib=ib(:,[2:3]);
    if min([ib0-ib(:,1);ib(:,2)-ib0])<=0,
      disp([om(1,:) 'Initial inequalities must be within the bounds']);
      return
    end;
  elseif n==2,
    if min(ib(:,2)-ib(:,1))<=0,
      disp([om(1,:) 'The lower bounds of the inequality constraints';...
            om(2,:) 'must be strictly less than the upper bounds.  ']);
      return
    end;
    ib0=(ib(:,1)+ib(:,2))/2;
  elseif n==1,
    ic=0; nic=0;
  else
    disp([om(1,:) 'Inequality constraints must have 2 or 3 columns.']);
    return
  end;
  if nic>=0.5,
    if lpb(1)>=0.5,
      pb=[ib; pb];
    else
      pb=ib;
```

```
    end;
    p=[ib0; p];
  end;
end;
clear ib ib0
%
if lpb(1)+nic>=0.5,
  lpb(2)=1;
end;
%
opd=[1 10 10 1.0e-5 1.0e-4];  % default optimization parameters
%
if exist('op')>=0.5,
  [m,n]=size(op);
  if m>1,
    disp([om(1,:) 'Control variables must be a vector'])
    return
  end;
  if n<5,
    op=[op(1:n) opd(n+1:5)];
  end;
else
  op=opd;
end;
op(1)=max(op(1),0);
for i=2:5,
  if op(i)<=0,
    op(i)=opd(i);
  end;
end;
rho=op(1); maxit=op(2); minit=op(3); delta=op(4); tol=op(5);
clear op opd
%
ob=cost(p(nic+1:nic+np),0);
[m,n]=size(ob);
if n>1.5,
  disp([om(1,:) 'Cost function must return a column vector'])
  return
end;
if m<nic+1,
  disp([om(1,:) 'The number of constraints in your COST function does';...
        om(2,:) 'not match the number specified in the call to SOLNP.']);
  return
end;
nec=m-1-nic;
nc=m-1;
clear m n
%
j=ob(1),
jh=j; t=0*ones(3,1);
%
if nc>0.5,
  if exist('l') <= 0.5,
    l=0*ones(nc,1);
  end;
  constraint=ob(2:nc+1)
  if nic>0.5,
    if min([constraint(nec+1:nc)-pb(1:nic,1);...
            pb(1:nic,2)-constraint(nec+1:nc)]) > 0,
      p(1:nic)=constraint(nec+1:nc);
```

```
    end;
    constraint(nec+1:nc)=constraint(nec+1:nc)-p(1:nic);
  end;
  t(2)=norm(constraint);
  if max([t(2)-10*tol, nic])<=0,
    rho=0;
  end;
else
  l=0;
end;
%
if exist('h') <= 0.5,
  h=eye(np+nic);
end;
%
mu=np; iteration=0;
while iteration<maxit,
  iteration=iteration+1,
  op=[rho minit delta tol nec nic np lpb];
  [p,l,h,mu]=subnp(p,op,l,ob,pb,h,mu);
  disp('Updated parameters:')
  p(nic+1:nic+np),
  ob=cost(p(nic+1:nic+np),iteration);
  t(1)=(j-ob(1))/max(abs(ob(1)),1);
  j=ob(1),
  if nc>0.5,
    constraint=ob(2:nc+1),
    if nic>0.5,
      if min([constraint(nec+1:nc)-pb(1:nic,1);...
              pb(1:nic,2)-constraint(nec+1:nc)]) > 0,
        p(1:nic)=constraint(nec+1:nc);
      end;
      constraint(nec+1:nc)=constraint(nec+1:nc)-p(1:nic);
    end;
    t(3)=norm(constraint);
    if t(3)<10*tol,
      rho=0; mu=min(mu, tol);
    end
    if t(3)<5*t(2),
      rho=rho/5;
    elseif t(3)>10*t(2),
      rho=5*max(rho, sqrt(tol));
    end;
    if max([tol+t(1), t(2)-t(3)]) <= 0,
      l=0*l; h=diag(diag(h));
    end;
    t(2)=t(3);
  end;
  if norm([t(1) t(2)])<=tol,
    maxit=iteration;
  end;
  jh=[jh j];
end;
%
if nic>0.5,
  ic=p(1:nic);
end;
p=p(nic+1:nic+np);
%
if norm([t(1) t(2)])<=tol,
```

```
  disp([om(1,:) 'Completed in']);
  iteration
  disp([om(2,:) 'iterations']);
else
  disp([om(1,:) 'Exiting after maximum number of iterations']);
  disp([om(2,:) 'Tolerance not achieved']);
end;
return
%
%-------------------------------------------------------------------------
%  Variable Glossary:
%-------------------------------------------------------------------------
%OB(1)      value of the cost objective function
%CONSTRAINT:vector of constraint values
%IB:        on input, contains the inequality constraint bounds + optionally
%           the values of the inequality constraints.  Gets converted to a
%           NIC x 2 matrix of inequality constraint bounds.
%IC:        NIC x 1 vector of inequality constraints
%ITERATION: index for major iterations
%J:         previous value of the cost objective function
%JH:        history of the cost function
%LPB (2):   vector flag which indicates the presence of parameter bounds
%           and or inequality constraints.
%             LPB(1) refers to parameter bounds, it is 0 if there are
%                    none, 1 if there are one or more.
%             LPB(2) refers to constraints of either type.
%M:         number of rows of a matrix, usually temporary
%msg:       long error message string
%N:         number of columns of a matrix, usually temporary
%NC:        total number of constraints (=NEC+NIC)
%NEC:       number of equality constraints
%NIC:       number of inequality constraints
%NP:        number of parameters
%OPD:       vector of default optimization control variables
%om:        string for optimization messages
%OP:        vector of control variables.
%             It is passed in as:
%               [RHO MAJIT MINIT DELTA TOL] (all optional)
%             It is passed to ISISUBOPT as:
%               [RHO MAJIT MINIT DELTA TOL NEC NIC NP LPB(1) LPB(2)]
%P:         On input, contains the parameters to be optimized.  During the
%           optimization this vector contains: [PIC;P]  where PIC contains
%           pseudo parameters corresponding to the inequality constraints.
%PB:        on input, optionally contains the parameter bounds + optionally
%           the values of the parameters (one or the other or both can be
%           specified).  Gets converted to a NPB x 2 matrix of parameter bounds.
%T (3):     vector of computed tolerances during optimization.
%             T(1) is the difference of the objective values between two
%                  consecutive iterations
%             T(2) is NORM(CONSTRAINT) before a major iteration
%             T(3) is NORM(CONSTRAINT) after a major iteration
%
%-------------------------------------------------------------------------
%  DOCUMENTATION:
%-------------------------------------------------------------------------
%
%The Function SOLNP solves nonlinear programs in standard form:
%
%        minimize              J(P)
%        subject to            EC(P)  =0
```

```
%                   IB(:,1)<=  IC(P)  <=IB(:,2)
%                   PB(:,1)<=    P    <=PB(:,2).
%where
%
%  J       : Cost objective scalar function
%  EC      : Equality constraint vector function
%  IC      : Inequality constraint vector function
%  P       : Decision parameter vector
%  IB, PB  : lower and upper bounds for IC and P.
%
% [P,L,JH,IC]=SOLNP(PB,IB,OP)
%
%  Output P      : optimal decision parameters
%         L      : optimal lagrangian multipliers
%         JH     : objective value's history of iterations
%         IC     : optimal values of inequality constraints
%
%  Input  PB = [P {PB}]
%
%            P   : any initial parameter within the parameter bound
%            PB  : optional parameter bound
%                  PB(:,1) is the lower bound for P, and
%                  PB(:,2) is the upper bound for P.
%
%         IB = [{IC} IB] (optional)
%
%            IC  : (optional) best approximation values for inequality
%                  constraints (IC) within the inequality constraint bound
%            IB  : inequality constraint bound
%                  IB(:,1) is the lower bound for IC, and
%                  IB(:,2) is the upper bound for IC.
%
%         OP=[RHO,MAJIT,MINIT,DELTA,TOL] (all optional with defaults)
%
%           RHO  : penalty parameter
%           MAJIT: maximum number of major iterations
%           MINIT: maximum number of minor iterations
%           DELTA: relative step size in forward difference evaluation
%           TOL  : tolerance on feasibility and optimality
%
%User-Defined Input function
%
%         OB : [OB]=COST(P,IT)
%
%           OB(1)        : cost objective function J value
%           OB(2:NEC+NIC): constraint function values of EC and IC
%           IT           : iteration number
%                          >0 - major iteration
%                          >0 - minor iteration
%                           0 - any other call
```