```
(*^

::[   frontEndVersion = "Microsoft Windows Mathematica Notebook Front End
Version 2.2";
      microsoftWindowsStandardFontEncoding;
      fontset = title, "Times New Roman", 24, L0, center, nohscroll,
bold;
      fontset = subtitle, "Times New Roman", 18, L0, center, nohscroll,
bold;
      fontset = subsubtitle, "Times New Roman", 14, L0, center,
nohscroll, bold;
      fontset = section, "Times New Roman", 14, L0, bold, grayBox;
      fontset = subsection, "Times New Roman", 12, L0, bold, blackBox;
      fontset = subsubsection, "Times New Roman", 10, L0, bold, whiteBox;
      fontset = text, "Times New Roman", 12, L0;
      fontset = smalltext, "Times New Roman", 10, L0;
      fontset = input, "Courier New", 12, L0, nowordwrap, bold;
      fontset = output, "Courier New", 8, L0, nowordwrap;
      fontset = message, "Courier New", 10, L0, nowordwrap, R65535;
      fontset = print, "Courier New", 10, L0, nowordwrap;
      fontset = info, "Courier New", 10, L0, nowordwrap;
      fontset = postscript, "Courier New", 8, L0, nowordwrap;
      fontset = name, "Times New Roman", 10, L0, nohscroll, italic,
B65535;
      fontset = header, "Times New Roman", 10, L0, right, nohscroll;
      fontset = footer, "Times New Roman", 10, L0, right, nohscroll;
      fontset = help, "Times New Roman", 10, L0, nohscroll;
      fontset = clipboard, "Times New Roman", 12, L0, nohscroll;
      fontset = completions, "Times New Roman", 12, L0, nowordwrap,
nohscroll;
      fontset = graphics, "Courier New", 10, L0, nowordwrap, nohscroll;
      fontset = special1, "Times New Roman", 12, L0, nowordwrap,
nohscroll;
      fontset = special2, "Times New Roman", 12, L0, center, nowordwrap,
nohscroll;
      fontset = special3, "Times New Roman", 12, L0, right, nowordwrap,
nohscroll;
      fontset = special4, "Times New Roman", 12, L0, nowordwrap,
nohscroll;
      fontset = special5, "Times New Roman", 12, L0, nowordwrap,
nohscroll;
      fontset = leftheader, "Times New Roman", 12, L0, nowordwrap,
nohscroll;
      fontset = leftfooter, "Times New Roman", 12, L0, nowordwrap,
nohscroll;
      fontset = reserved1, "Courier New", 10, L0, nowordwrap, nohscroll;]
:[font = section; inactive; backColorRed = 65535; backColorGreen = 65535;
backColorBlue = 65535; fontColorRed = 0; fontColorGreen = 0;
fontColorBlue = 0; bold; fontName = "Times New Roman"; fontSize = 14; ]


Package VariableMetricAlgorithm
:[font = input; initialization; nowordwrap; backColorRed = 65535;
backColorGreen = 65535; backColorBlue = 65535; fontColorRed = 0;
```

```
fontColorGreen = 0; fontColorBlue = 0; bold; fontName = "Courier New";
fontSize = 12; ]
*)
(* :Title: VariableMetricAlgorithm *)

(* :Context: Nonsmooth Optimization *)

(* :Author: Dr. Stanislav Uryasev    *)

(* :Affiliation:  University of Florida,
474 Weil Hall, Gainesville, FL 32611-6595, USA; E-mail:
Uryasev@ise.ufl.edu     *)

(* :Summary: This package implements a variable metric algorithm
 for optimizing nonlinear (possibly nonsmooth) functions in the
 Euclidean space *)

(* :History:  In December 1989 a FORTRAN 77 subroutine is coded by
 S. Uryasev at the International  Institute for Applied Systems
 Analysis. This subroutine was translated to MATHEMATICA language
 in August 1996 *)

(* :Mathematica Version: 2.2 *)

(* :Keywords: nonsmooth optimization, nondifferentiable optimization,
  nonlinear programming, objective function,  minimize, optimize  *)

(*: Method: This package implements an adaptive variable metric
 algorithm for minimization of nonlinear (possibly nonsmooth)
 functions. In this method,two subgradient algorithms work
 simultaneously, the first in the main space, the second with respect
 to  the matrix improving the convergence speed. The method is
 described in S. Uryasev, "New  Variable-Metric Algorithms for
 Nondifferentiable Optimization Problems." Journal of Optimization
 Theory and Applications  Vol. 71, No. 2, (1991), 359-388. *)


(* :Problem statement: The variable metric algorithm solves the
 problem
                  f(x) -> min,                    (1)
 where f(x) is a nonlinear (possibly nonsmooth) function, and
 x is a vector in the Euclidean n-dimensional space.
 The current version does not include constraints (directly).
 However, they can be included, for example, with nonsmooth penalty
 function. The minimization problem (1) with constraints
             f1(x) <= 0 , ... , fI(x) <= 0,
 is equivalent to the problem without constraints
             f(x) + C Max[0,f1(x),...,fI(x)] -> min,
 where C is a sufficiently large constant.*)

(* :Discussion: This program formally does not check for optimality of
 the returned vector which should thus be considered just an
 approximation of the local extremum *)
```

```
BeginPackage["NonsmoothOptimization`VariableMetricAlgorithm`"]

ison = Off[General::spell1,General::spell]

VariableMetricAlgorithm::usage = "VariableMetricAlgorithm[x0,opts] finds
a local minimum of a nonlinear (possibly nonsmooth) function.
To apply VariableMetricAlgorithm, the following two functions should be
defined :
   fun[x] := performance function depending upon the array x of real
         variables;
   grad[x]:= gradient (or generalized gradient if fun[x] is nonsmooth)
         of the performance function.
For example,
   fun[{x1_,x2_}]:= Abs[x1] + Abs[10*x2];
   grad[{x1_,x2_}]:= {Sign[x1],Sign[x2]*10}.
Algorithm has four stop criteria:
   1. ArgumentPrecision = desired accuracy on the argument x;
   2. GradientPrecision = desired value of the norm of the gradient
      (subgradient of function is nonsmooth);
   3. NumberIterations = maximal number of iterations to be performed;
   4. ConvergenceGoal = desired value of performance function;
By default: ArgumentPrecision->10.^-20, GradientPrecision->10.^-6,
   NumberIterations -> 100, ConvergenceGoal -> -10.^600 .
Initial stepsize can be set with the option InitialStepsize.
By default, InitialStepsize->0.1. With the option PrintFrequency->k,
algorithm prints results of calculations each  k-th iteration.
By default, PrintFrequency->0. OutputList option specifies parameters
to be printed. By default, OutputList->{Function,Approximation}.
The output can include
  Function = performance function value,
  Approximation = vector of variables,
  Gradient = gradient vector,
  Iteration = iteration number,
  InternalIteration = internal iteration number calculating total
    number of performance function calls";

GradientPrecision::usage = "GradientPrecision is an option to stop
VariableMetricAlgorithm; it specifies a minimal value for the
gradient (subgradient) Euclidean norm. The default setting is
GradientPrecision->10.^-20";


ArgumentPrecision::usage = "ArgumentPrecision is an option to stop
VariableMetricAlgorithm; it specifies the minimal Euclidean distance
between successive approximations. This parameter can be set
to desirable accuracy with respect to the argument of the
performance function. The default setting is ArgumentPrecision->10^-20";

NumberIterations::usage = "NumberIterations is an option to stop
VariableMetricAlgorithm. It specifies  the maximum number of main
iterations. The default setting is NumberIterations -> 100";

ConvergenceGoal::usage = "ConvergenceGoal is an option to stop
```

```
VariableMetricAlgorithm when the value of performance
function is less than ConvergenceGoal. The default setting is
ConvergenceGoal -> -10.^600";

InitialStepsize::usage = "InitialStepsize is an option of
VariableMetricAlgorithm specifying initial stepsize for the
algorithm. The default setting is InitialStepsize->0.1";

PrintFrequency::usage = "PrintFrequency is an option of
VariableMetricAlgorithm specifying print frequency of the
algorithm. Algorithm prints results of calculations on each
k-th iteration with option: PrintFrequency->k.
The default setting is PrintFrequency->0, i.e., algorithm prints
only the final result and does not print intermediate
approximations. OutputList option specifies parameters to be printed.
By default, OutputList->{Function,Approximation}. The output
can include
  Function = performance function value,
  Approximation = vector of variables,
  Gradient = gradient vector,
  Iteration = iteration number,
  InternalIteration = internal iteration number calculating total
    number of performance function calls";

OutputList::usage = "OutputList is an option of
VariableMetricAlgorithm specifying print outputs of the
algorithm. Algorithm prints results of calculations on each
k-th iteration with option: PrintFrequency->k. The default setting is
PrintFrequency->0, i.e., algorithm prints only the final result
and does not print intermediate approximations. OutputList option
specifies parameters to be printed. By default,
OutputList->{Function,Approximation}. The output can include
  Function = performance function value,
  Approximation = vector of variables,
  Gradient = gradient vector,
  Iteration = iteration number,
  InternalIteration = internal iteration number calculating total
       number of performance function calls";

VariableMetricAlgorithm::lstop0=
"Value of the objective function achieved ConvergenceGoal";

VariableMetricAlgorithm::lstop1=
"Norm of subgradient achieved GradientPrecision";

VariableMetricAlgorithm::lstop2=
"Distance between two approximations achieved ArgumentPrecision";

VariableMetricAlgorithm::lstop3=
"Algorithm performed specified NumberIterations";

Options[VariableMetricAlgorithm] = {
ArgumentPrecision->10.^-20,InitialStepsize->0.1,
GradientPrecision->10.^-6,
```

```
NumberIterations -> 100,ConvergenceGoal -> -10.^600,PrintFrequency->0,
OutputList->{Function,Approximation}
                                                                    };

VariableMetricAlgorithm[x0_,opts___]:=
Module[{x,xpr,b,gr,grold,grb,grbb,vsold,vsnew,xvector,grvector,
 shift,fnpr,fn,outlist,n=Length[x0],nk,bstep,rodecr,roincr,lstop,
 nouv,induv,s,sfull,smax,grbnrm,grnorm,shnorm,noprin,sprint,grprint},

{dxstop, ro, grstop, smax, flevel,outlist,noprin} =
 {ArgumentPrecision, InitialStepsize, GradientPrecision,
  NumberIterations,ConvergenceGoal,OutputList,PrintFrequency}
  /.  {opts} /.  Options[VariableMetricAlgorithm];

PrintOptions={Function->Function->fn,
Approximation->Approximation->x,
Iteration-> Iteration->sprint,
Gradient->Gradient->gr,
InternalIteration-> InternalIteration->sfull};

(* METHOD PARAMETERS *)
      x=x0;
      nk=n; (* nk can be decreased (nk <= n) to reduce,
               if necessary, the size of the matrix b *)
      bstep=0.55;
      rodecr=0.8;
      roincr=1.25;
      lstop=3;

(* INITIALIZATION *)
      nouv=1;
      lstep=0;
      induv=1;
      sfull=1;
      s=1;
      sprint=1;
      grbnrm=1.;
      lenoutlist=Length[outlist];
      grprint=False;
      Do[If[outlist[[i]]==Gradient,grprint=True] ,{i, lenoutlist}];

(* FUNCTION CALCULATION *)
      fn=fun[x] //N;

(* SUBGRADIENT CALCULATION *)
     gr=grad[x] //N;

(* FUNCTION VALUE STOP *)
      If[flevel >= fn,
         lstop=0;
         Message[VariableMetricAlgorithm::lstop0];
         Return[outlist  /. PrintOptions ]
      ];
```

```
(* PRINT *)
 If[noprin!=0, Print[outlist  /. PrintOptions ]];

(* CALCULATION OF SUBGRADIENT NORM *)
      grnorm=Sqrt[gr.gr];

(* SUBGRADIENT NORM STOP *)
      If[grnorm<=grstop,
         lstop=1;
         Message[VariableMetricAlgorithm::lstop1];
          Return[outlist  /. PrintOptions]
      ];

(* SUBGRADIENT NORMALIZATION *)
      gr/=grnorm;

(*     BEGINNING OF MAIN CYCLE *)

Do[
sprint=s;

(* GROLD MEMORY *)
      grold=gr;

(* CALCULATION OF GRB,GRBB *)
      If[s==1,
         grbb=gr; grbnrm=Max[Sqrt[gr.gr],10^-20];
         grb=Table[1.,{j,nk}];
         b=Table[If[i==j,1.,0.],{i,n},{j,nk}],
         (* following two Do operators perform
            matrix operations
               grb=Transpose[b].gr;
               grbb=b.grb;
           in matrix form, sometimes, these operators
           do not work because of unknown reasons *)
         Do[
            grb[[i]]=0;
            Do[
               grb[[i]]=grb[[i]]+b[[j,i]]*gr[[j]]
           ,{j,n}]
         ,{i,nk}];
         Do[
            grbb[[i]]=0.;
            Do[
               grbb[[i]]=grbb[[i]]+b[[i,j]]*grb[[j]]
             ,{j,nk}]
         ,{i,n}];


(*       CALCULATION OF GRB NORM *)
         grbnrm=Max[Sqrt[grb.grb],10^-20];

       ]; (* END OF IF *)
```

```
(* MINIMUM SEARCH ON DIRECTION *)
fnpr=fn+1;

While[fn <=  fnpr,
(* XPR,FNPR MEMORY *)
          xpr=x;
          fnpr=fn;

(* MOTION IN SPACE OF X  *)
          shift=grbb*ro/grbnrm;
          x=x-shift;

(* FUNCTION  CALCULATION *)
          fn=fun[x] //N;

(* CALCULATION OF SHIFT NORM *)
          shnorm=Sqrt[shift.shift];

(* SHIFT NORM STOP *)
          If[shnorm<=dxstop,
             lstop=2;
             fn=fnpr;
              x=xpr;
             Message[VariableMetricAlgorithm::lstop2];
              If[grprint,gr = grad[x]];
             Return[outlist  /. PrintOptions]
          ];


(* FUNCTION VALUE STOP *)
          If[flevel>=fn,
            lstop=0;
            Message[VariableMetricAlgorithm::lstop0];
             If[grprint,gr = grad[x]];
            Return[outlist  /. PrintOptions  ]
          ];

(* INCREASE OF SFULL  *)
          sfull+=1;

(* RO CONTROL *)
          If[fn<=fnpr,
              lstep+=1;
              If[s==1,ro*=1.5];
              If[lstep>nouv && s>1, ro*=roincr]
          ];

]; (* END OF WHILE *)
      If[lstep==0, ro*=rodecr];

(* SUBGRADIENT CALCULATION *)
      gr=grad[x] //N;

(* CALCULATION OF SUBGRADIENT NORM *)
```

```
      grnorm=Sqrt[gr.gr];

(* SUBGRADIENT NORM STOP *)
      If[grnorm<=grstop,
         fn=fnpr;
         x=xpr;
         lstop=1;
         Message[VariableMetricAlgorithm::lstop1];
         If[grprint,gr = grad[x]];
         Return[outlist  /. PrintOptions ]
      ];

(* SUBGRADIENT NORMALIZATION *)
         gr/=grnorm;

(* MATRIX B CALCULATION *)
      vsold= grold.b;
      vsnew= gr.b;
      Do[
         Do[
          b[[i,j]]=b[[i,j]]+bstep*
                   (gr[[i]]*vsold[[j]]+grold[[i]]*vsnew[[j]])
         ,{j,nk}]
      ,{i,n}];

(* GR,X,FN RESET *)
      x=xpr;
      fn=fnpr;
      If[lstep==0, gr=grold];

(* LSTEP RESET *)
      lstep=0;

(* PRINT *)
 If[noprin!=0,
    grold=gr;
    If[grprint,gr = grad[x]];
    If[Ceiling[s/noprin]==s/noprin,
                    Print[outlist  /.   PrintOptions]];
   gr=grold;
   ];

(* END OF MAIN CYCLE *)
,{s,smax}];

 If[lstop==3,Message[VariableMetricAlgorithm::lstop3]];
 If[grprint,gr = grad[x]];
 Return[outlist    /.  PrintOptions ]
 ]

On[ison]

EndPackage[]
(*
```

```
:[font = section; inactive; ]


Examples of VariableMetricAlgorithm  use
:[font = input; nowordwrap; ]


(* Examples of VariableMetricAlgorithm use *)
(* The VariableMetricAlgorithm solves the
 problem
                  f(x) -> min,                    (1)
 where f(x) is a nonlinear (possibly nonsmooth) function, and
 x is a vector in the Euclidean n-dimensional space.
 The current version does not include constraints. However,
 they can be included, for example, with nonsmooth penalty
 function. The minimization problem (1) with constraints
            f1(x) <= 0 , ... , fI(x) <= 0,
 is equivalent to the problem without constraints
            f(x) + C Max[0,f1(x),...,fI(x)] -> min,
 where C is a sufficiently large constant.*)

(* VariableMetricAlgorithm[x0,opts] finds a local minimum. To
  apply it, the following two functions should be defined:
   fun[x] := performance function depending upon
             the array x of real variables;
   grad[x]:= gradient (or generalized gradient if fun[x]
             is nonsmooth) of the performance function.
  For example,
   fun[{x1_,x2_}]:= Abs[x1] + Abs[10*x2];
   grad[{x1_,x2_}]:= {Sign[x1],Sign[x2]*10}.
  Algorithm has four stop criteria:
   1. ArgumentPrecision = desired accuracy on the argument x;
   2. GradientPrecision = desired value of the norm of the
      gradient subgradient of function is nonsmooth);
   3. NumberIterations = maximal number of iterations to be
      performed;
   4. ConvergenceGoal = desired value of performance
      function;
  By default: ArgumentPrecision->10.^-20,
              GradientPrecision->10.^-6,
              NumberIterations -> 100,
              ConvergenceGoal -> -10.^600 .
  Initial stepsize can be set with the option
  InitialStepsize. By default, InitialStepsize->0.1.
  With the option PrintFrequency->k, algorithm prints
  results of calculations each k-th iteration.
  By default, PrintFrequency->0.
  OutputList option specifies parameters to be printed.
  By default, OutputList->{Function,Approximation}.
  The output can include
    Function = performance function value,
    Approximation = vector of variables,
    Gradient = gradient vector,
    Iteration = iteration number,
```

```
    InternalIteration = internal iteration number
     calculating total number of performance function calls *)


:[font = input; startGroup; nowordwrap; backColorRed = 65535;
backColorGreen = 65535; backColorBlue = 65535; fontColorRed = 0;
fontColorGreen = 0; fontColorBlue = 0; bold; fontName = "Courier New";
fontSize = 12; ]

fun[{x1_,x2_}]:= Abs[x1] + Abs[10*x2];
grad[{x1_,x2_}]:= {Sign[x1],Sign[x2]*10};

x0={10,2};
VariableMetricAlgorithm[x0,NumberIterations -> 200,
ArgumentPrecision->10.^-6,PrintFrequency->20,
OutputList -> {Function,Iteration,Approximation}]
:[font = print; inactive; formatted; output; nowordwrap; ]
{Function -> 30., Iteration -> 1, Approximation -> {10, 2}}
{Function -> 0.199917, Iteration -> 20, Approximation -> {-0.157332,
0.00425856}}
{Function -> 0.0200816, Iteration -> 40, Approximation -> {0.0142812,
0.000580041}}
{Function -> 0.0075678, Iteration -> 60, Approximation -> {0.00180915,
0.000575865}}

-6
{Function -> 0.000122647, Iteration -> 80, Approximation -> {-
0.0000832797, -3.93669 10  }}
                         -6
-6            -8
{Function -> 3.97612 10  , Iteration -> 100, Approximation -> {-3.58701
10  , 3.89113 10  }}



;[o]
{Function -> 30., Iteration -> 1, Approximation -> {10, 2}}
{Function -> 0.199917, Iteration -> 20, Approximation -> {-0.157332,
0.00425856}}
{Function -> 0.0200816, Iteration -> 40, Approximation -> {0.0142812,
0.000580041}}
{Function -> 0.0075678, Iteration -> 60, Approximation -> {0.00180915,
0.000575865}}

-6
{Function -> 0.000122647, Iteration -> 80, Approximation -> {-
0.0000832797, -3.93669 10  }}
                         -6
-6            -8
{Function -> 3.97612 10  , Iteration -> 100, Approximation -> {-3.58701
10  , 3.89113 10  }}
:[font = message; inactive; formatted; output; nowordwrap; ]
No Input Form was generated for this Cell
```

```
;[o]
VariableMetricAlgorithm::lstop2: Distance between two approximations
achieved ArgumentPrecision
:[font = output; inactive; formatted; output; endGroup; nowordwrap; ]
{Function -> 5.374996040638463*10^-7, Iteration -> 110,
  Approximation -> {-(4.251502666366882*10^-7), 1.123493374271581*10^-8}}



;[o]
                      -7                                                -7
-8
{Function -> 5.375 10  , Iteration -> 110, Approximation -> {-4.2515 10
, 1.12349 10  }}
:[font = input; nowordwrap; ]


:[font = input; startGroup; nowordwrap; backColorRed = 65535;
backColorGreen = 65535; backColorBlue = 65535; fontColorRed = 0;
fontColorGreen = 0; fontColorBlue = 0; bold; fontName = "Courier New";
fontSize = 12; ]

(*  f(x) = (x1-10)^2+x2^2 -> min
            Abs[x1+x2] <= 1 *)

fun[{x1_,x2_}]:=
Module[{fn,PenaltyConstant,Constraint,ObjectiveFunction},
PenaltyConstant=10;
Constraint=Max[Abs[x1+x2] -1,0];
ObjectiveFunction=(x1-10)^2+x2^2;
fn= ObjectiveFunction + PenaltyConstant * Constraint;
Return[fn]
];

grad[{x1_,x2_}]:=
Module[{gr,PenaltyConstant,GrConstraint,GrObjectiveFunction,
        Constraint},
PenaltyConstant=10;

If[Abs[x1+x2]-1 <0,GrConstraint={0,0},
       GrConstraint= Sign[x1+x2]*{1,1}];
GrObjectiveFunction={2*(x1-10),2*x2};
gr= GrObjectiveFunction + PenaltyConstant  * GrConstraint;
Return[gr]
];

x0={10,2};
VariableMetricAlgorithm[x0,NumberIterations -> 200,
ArgumentPrecision->10.^-6,PrintFrequency->30,
OutputList -> {Function,Iteration,Approximation,InternalIteration}]
:[font = print; inactive; formatted; output; nowordwrap; ]
```

```
{Function -> 114., Iteration -> 1, Approximation -> {10, 2},
InternalIteration -> 1}
{Function -> 41.1573, Iteration -> 30, Approximation -> {5.9277, -
4.45533}, InternalIteration -> 50}
{Function -> 41.1534, Iteration -> 60, Approximation -> {5.91753, -
4.44185}, InternalIteration -> 106}
{Function -> 40.5012, Iteration -> 90, Approximation -> {5.50505, -
4.50385}, InternalIteration -> 169}
{Function -> 40.5001, Iteration -> 120, Approximation -> {5.49548, -
4.49538}, InternalIteration -> 218}
{Function -> 40.5, Iteration -> 150, Approximation -> {5.49942, -
4.49939}, InternalIteration -> 284}
{Function -> 40.5, Iteration -> 180, Approximation -> {5.50054, -
4.50053}, InternalIteration -> 330}



;[o]
{Function -> 114., Iteration -> 1, Approximation -> {10, 2},
InternalIteration -> 1}
{Function -> 41.1573, Iteration -> 30, Approximation -> {5.9277, -
4.45533}, InternalIteration -> 50}
{Function -> 41.1534, Iteration -> 60, Approximation -> {5.91753, -
4.44185}, InternalIteration -> 106}
{Function -> 40.5012, Iteration -> 90, Approximation -> {5.50505, -
4.50385}, InternalIteration -> 169}
{Function -> 40.5001, Iteration -> 120, Approximation -> {5.49548, -
4.49538}, InternalIteration -> 218}
{Function -> 40.5, Iteration -> 150, Approximation -> {5.49942, -
4.49939}, InternalIteration -> 284}
{Function -> 40.5, Iteration -> 180, Approximation -> {5.50054, -
4.50053}, InternalIteration -> 330}
:[font = message; inactive; formatted; output; nowordwrap; ]
No Input Form was generated for this Cell



;[o]
VariableMetricAlgorithm::lstop3: Algorithm performed specified
NumberIterations
:[font = output; inactive; formatted; output; endGroup; nowordwrap; ]
{Function -> 40.50000000628499, Iteration -> 200, Approximation ->
{5.499982341413129, -4.499982335751996},
  InternalIteration -> 364}



;[o]
{Function -> 40.5, Iteration -> 200, Approximation -> {5.49998, -
4.49998}, InternalIteration -> 364}
^*)
```