```
/* KKT.mac
    Maxima software for Economic Analysis
    Ted Woollett, Aug. 5, 2021
    http://web.csulb.edu/~woollett/
    http://web.csulb.edu/~woollett/eam.html

    Copyright (C) 2021 Edwin L. Woollett  <woollett@charter.net>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU GENERAL PUBLIC LICENSE, Version 2, June 1991,
    as published by the Free Software Foundation.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details. You should have received a
    copy of the GNU General Public License along with this program.
    If not, see http://www.fsf.org/licensing/ .
*/



/* KKT (f,max, g) calls KKTmax(f,g)
   KKT (f,min, g) calls KKTmin(f,g)
   KKT (f,max, g1,g2) calls KKTmax2(f,g1,g2)
   KKT (f,min, g1,g2) calls KKTmin2(f,g1,g2)
*/




KKT([%v]) :=
 block([%lv, %lm, %lvr],

    %lv : length (%v),
    %lm : %v[2],
    %lvr : cons (%v[1], rest (%v,2)),
    /* display (%v, %lv,%v[1],rest(%v,2), %lm, %lvr), */

    if (%lv = 3 or %lv = 4) and (%lm = min or %lm = max) then
        (if %lv = 3 then
             if %lm = max then apply ('KKTmax, %lvr)
               else apply ('KKTmin, %lvr)
        else if %lv = 4 then
             if %lm = max then apply ('KKTmax2, %lvr)
               else apply ('KKTmin2, %lvr))
    else (print ("KKT(f,minmax,g) or KKT (f,minmax,g1,g2)"),
          print (" where minmax is either max, with g's >= 0"),
          print (" or minmax = min, with g's <= 0"),
          done))$


/*****************************************/


/* multiple, Dsingle, and single help distinuish different types of lists
   returned by solve */

multiple(%X) :=
 (if length(%X) > 1 and part(%X,1,0) = "[" then true else false)$

Dsingle(%X) := (if length(%X) = 1 and part(%X, 1, 0) = "[" then true else false)$

single (%X) :=
 ( if length(%X) > 1 and part(%X, 1,0) = "=" then true else false)$
```

```
/*
KKTmax(f, g) , both args assumed to depend on (x,y),
      assumes f(x,y) is the objective function to be maximized
      such that g(x,y) >= 0
*/

KKTmax(func, %g) :=
    block ([%n:0, lam, LF, LF1, LF2,LF3, solns, objsub, fx0, fy0,
            F10, F20, F30, F10x, F20x,F10y, F20y,asoln, gsoln,
             lam0 : [lam = 0],lvf],

        lvf :listofvars(func),
        /* print("lvf = ",lvf), */
         if (lfreeof(lvf,x) and lfreeof(lvf,y)) then
           ( print (" the objective  and constraint expressions should
                        depend only on (x,y)"),
             return (done)),

        fx0 : subst(0, x, func),
        fy0 : subst(0, y, func),
        /* display(fx0, fy0), */
        LF : func + lam*%g,
     /*    display (LF), */
        LF1 : diff(LF,x),
        LF2 : diff(LF,y),
        LF3 : diff(LF,lam),
    /*   display(LF1,LF2, LF3),  */

/* lam = 0 cases   */

       /* print ("---------------------------------"),
        print (" case lam = 0 "), */
        F10 : subst(0,lam,LF1),
        F20 : subst(0,lam,LF2),
        /* display (F10, F20),   */
        if (numberp(F10) and F10 > 0) or (numberp(F20) and F20 > 0) then %n : %n + 1
        else (
/* lam = 0, x = 0, y > 0 */
            /* print ("case lam = 0, x = 0, y > 0"), */
            if numberp(fx0) and abs(float (fx0)) < 1e-10 then %n : %n + 1
            /* func close to zero at x = 0; no solution */

              else (
                 F10x : subst(0, x, F10),  /* both lam and x are zero here */
                 F20x : subst(0, x, F20),
                 if (numberp(F10x) and F10x > 0) or (numberp(F20x) and F20x > 0) then %n : %n + 1

                 else (
                       solns : solve (F20x, y),  /* since y > 0, we enforce F20x = 0 */
                    /*    display (solns), */

                        if multiple (solns) then
                              for j thru length(solns) do (
                              asoln : solns[j],
                              gsoln : at (at (%g, asoln), x = 0),
                           /*  display (gsoln), */
                             if at(y, asoln) > 0 and gsoln >= 0 and at(F10x, asoln) <= 0 then (
                                   asoln : flatten ([lam0,[x = 0], asoln]),
                                   print ("constraint is non-binding"),
                                   print ("soln = ", asoln,"  g(x,y) at soln =
                                   ",gsoln),
                                   objsub : at( at(func,asoln), x = 0),
                                   print ("objsub = ",objsub),
                                    print ("soln = ", float (asoln),"  objsub = ", float
                                     (objsub)))),

                        if Dsingle (solns) or single (solns) then (
```

```
                        if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                        gsoln : at( at(%g,asoln), x = 0),
                     /*   display (gsoln), */
                        if at(y, asoln) > 0 and gsoln >= 0 and at(F10x, asoln) <= 0 then
                        (
                            asoln : flatten ([lam0,[x = 0], asoln]),
                            print ("constraint is non-binding"),
                            print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                            objsub : at( at(func,asoln), x = 0),
                            print ("objsub = ",objsub),
                            print ("soln = ", float (asoln),"  objsub = ",float
                            (objsub)))))),
/* lam = 0, x > 0, y = 0 */
          /* print ("case lam = 0, x > 0, y = 0"), */
          if numberp(fy0) and abs(float (fy0)) < 1e-10 then %n : %n + 1
              /* func close to zero at y = 0; no solution */


          else (
               F10y : subst(0, y, F10),  /* both lam and y are zero here */
               F20y : subst(0, y, F20),
               if (numberp(F10y) and F10y > 0) or (numberp(F20y) and F20y > 0) then %n : %n + 1

               else (
                     solns : solve (F10y, x), /* since x > 0, enforce F10y = 0 */
                   /*  display (solns), */

                     if multiple (solns) then
                            for j thru length(solns) do (
                            asoln : solns[j],
                            gsoln : at(at (%g, asoln), y = 0),
                          /*  display (gsoln), */
                            if at(x,asoln) > 0 and gsoln >= 0 and at(F20y,asoln) <= 0 then (
                                 asoln : flatten ([lam0,asoln, [y = 0]]),
                                 print ("constraint is non-binding"),
                                 print ("soln = ", asoln,"  g(x,y) at soln =
                                 ",gsoln),
                                 objsub : at( at(func,asoln), y = 0),
                                 print ("objsub = ",objsub),
                                 print ("soln = ", float (asoln),"  objsub = ", float
                                 (objsub)))),

                     if Dsingle (solns) or single (solns) then (
                            if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                            gsoln : at( at(%g,asoln), y = 0),
                          /*  display (gsoln), */
                            if at(x,asoln) > 0 and gsoln >= 0 and at(F20y,asoln) <= 0 then (
                                asoln : flatten ([lam0,asoln, [y = 0]]),
                                print ("constraint is non-binding"),
                                print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                                objsub : at( at(func,asoln), y = 0),
                                print ("objsub = ",objsub),
                                print ("soln = ", float (asoln),"  objsub = ",float
                                (objsub)))))),



/* lam = 0, x > 0, y > 0 */

               /* print ("case lam = 0, x > 0, y > 0"), */
               solns : solve ([F10,F20], [x, y]),  /* since x>0,y>0, enforce F10=0,F20=0 */
               /* display (solns), */

               if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
                       gsoln : at (%g, asoln),
```

```
                    if gsoln >= 0 and at(x,asoln) > 0
                              and at(y,asoln) > 0 then (
                         asoln : flatten ([lam0,asoln]),
                         print ("constraint is non-binding"),
                         print ("soln = ", asoln,"  g(x,y) at soln =
                         ",gsoln),
                         objsub : at (func, asoln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

               if Dsingle (solns) or single (solns) then (
                   if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                   gsoln : at (%g, asoln),
                   if gsoln >= 0 and at(x,asoln) > 0
                                 and at (y,asoln) > 0 then (
                        asoln : flatten ([lam0,asoln]),
                        print ("constraint is non-binding"),
                        print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                        objsub : at (func, asoln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub)))))),

/*  lam > 0,  x = 0, y > 0  */

       /* print ("-------------------------------------"), */
       /* print (" case lam > 0, x = 0, y > 0  "), */
       F10 : subst (0,x,LF1),
       F20 : subst (0,x,LF2),  /* only x is zero here, redefining F10, F20 */
       F30 : subst (0,x,LF3),
       /* display (F20, F30), */
       if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
       else if numberp (F10) and F10 > 0 then %n : %n + 1
       else if numberp (F20) and F20 > 0 then %n : %n + 1
       else if numberp (F30) and F30 < 0 then %n : %n + 1

       else (
             solns : solve ( [F20, F30], [y, lam]),    /* since y>0,lam>0, enforce F20=0,F30=0
             */
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      gsoln : at ( at(%g, asoln), x = 0),
                      if at(y,asoln) > 0 and at(lam, asoln) > 0 and  gsoln >= 0
                               and at(F10,asoln) <= 0  then (
                           asoln : flatten ([[x = 0],asoln]),
                           print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                           objsub : at( at(func, asoln), x = 0),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  gsoln : at( at(%g, asoln), x = 0),
                  if at(y,asoln) > 0 and at(lam,asoln) > 0 and gsoln >= 0
                          and at(F10,asoln) <= 0  then (
                        asoln : flatten ([[x = 0],asoln]),
                        objsub : at(at(func, asoln), x = 0),
                        print ("soln = ", asoln,"  g(x,y) at soln =
                        ",gsoln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/*  lam > 0, x > 0, y = 0 */

       /* print ("-------------------------------------"), */
```

```
        /* print (" case  lam > 0, x > 0, y = 0 "), */
        F10 : subst (0,y,LF1),
        F20 : subst (0,y,LF2),  /* only y is zero here, redefining F10,F20,F30 */
        F30 : subst (0,y,LF3),
        /* display (F10, F30), */

        if numberp (fy0) and abs (float (fy0)) < 1e-10 then %n : %n + 1

        else if numberp (F10) and F10 > 0 then %n : %n + 1
        else if numberp (F20) and F20 > 0 then %n : %n + 1

        else if numberp (F30) and F30 < 0 then %n : %n + 1

        else (
              solns : solve ( [F10, F30], [x, lam]),    /*x>0,lam>0, enforce F10=0,F30=0 */
             /* display (solns),   */

              if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
                       gsoln : at(at(%g,asoln), y = 0),
                       if at(x,asoln) > 0 and at(lam,asoln) > 0 and  gsoln >= 0
                           and  at(F20,asoln) <= 0  then(
                            asoln : flatten ([asoln, [y = 0]]),
                            objsub : at( at(func,asoln), y = 0),
                            print ("soln = ", asoln,"  g(x,y) at soln =
                            ",gsoln),
                            print ("objsub = ",objsub),
                            print ("soln = ", float (asoln),"  objsub = ", float
                            (objsub)))),

              if Dsingle(solns) or single(solns) then (
                    if Dsingle(solns) then asoln : solns[1] else asoln : solns,
                    gsoln : at( at(%g, asoln), y =
                    0),
                    if at(x,asoln) > 0 and at(lam,asoln) > 0 and gsoln >= 0
                              and at(F20,asoln) <= 0    then (
                         asoln : flatten ([asoln, [y = 0]]),
                         objsub : at( at(func,asoln), y = 0),
                         print ("soln = ", asoln,"  g(x,y) at soln =
                         ",gsoln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam > 0,  x > 0, y > 0  */

        /* print ("-----------------------------------------"),     */
        /* print (" case   lam > 0, x > 0, y > 0 "), */
        solns : solve ([LF1, LF2, LF3],[x, y, lam]),
          /* display (solns), */

        if multiple (solns) then
                for j thru length(solns) do (
                       asoln : solns[j],
                       gsoln : at (%g, asoln),
                       if gsoln >= 0 and at(x, asoln) > 0 and at(y,asoln) > 0 and
                                       at(lam,asoln) > 0  then
                                        (
                            objsub : at (func, asoln),
                            print ("soln = ", asoln,"  g(x,y) at soln =
                            ",gsoln),
                            print ("objsub = ",objsub),
                            print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

        if Dsingle(solns) or single(solns) then (
                if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                gsoln : at (%g,asoln),
```

```
              if  gsoln >= 0 and  at(x,asoln) > 0 and at(y,asoln) > 0 and
                                  at(lam,asoln) > 0  then (
                              objsub : at (func,asoln),
                              print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                              print ("objsub = ", objsub),
                              print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),
        done )$

/*  end of KKTmax (f,g)  */




/*
KKTmin(f, g) , both args assumed to depend on (x,y),
      assumes f(x,y) is the objective function to be minimized
      such that g(x,y) <= 0
*/

KKTmin(func, %g) :=
    block ([%n:0, lam, LF, LF1, LF2,LF3, solns, objsub, fx0, fy0,
            F10, F20, F30, F10x, F20x,F10y, F20y,asoln, gsoln,
             lam0 : [lam = 0]],

        if listofvars(func) # [x,y] or listofvars(%g) # [x,y] then
           (print (" the objective  and constraint expressions should
                        depend only on (x,y)"),
            return (done)),

        fx0 : subst(0, x, func),
        fy0 : subst(0, y, func),
        /* display(fx0, fy0), */
        LF : func + lam*%g,
     /*    display (LF), */
        LF1 : diff(LF,x),
        LF2 : diff(LF,y),
        LF3 : diff(LF,lam),
    /*   display(LF1,LF2, LF3),  */

/* lam = 0 cases   */

       /* print ("---------------------------------"),
        print (" case lam = 0 "), */
        F10 : subst(0,lam,LF1),
        F20 : subst(0,lam,LF2),
        /* display (F10, F20),   */
        if (numberp(F10) and F10 < 0) or (numberp(F20) and F20 < 0) then %n : %n + 1
        else (
/* lam = 0, x = 0, y > 0 */
            /* print ("case lam = 0, x = 0, y > 0"), */
            if numberp(fx0) and abs(float (fx0)) < 1e-10 then %n : %n + 1
            /* func close to zero at x = 0; no solution */

              else (
                 F10x : subst(0, x, F10),  /* both lam and x are zero here */
                 F20x : subst(0, x, F20),
                 if (numberp(F10x) and F10x < 0) or (numberp(F20x) and F20x < 0) then %n : %n + 1

                 else (
                       solns : solve (F20x, y),  /* since y > 0, we enforce F20x = 0 */
                    /*    display (solns), */

                        if multiple (solns) then
                              for j thru length(solns) do (
                              asoln : solns[j],
                              gsoln : at (at (%g, asoln), x = 0),
                           /*  display (gsoln), */
```

```
                        if at(y, asoln) > 0 and gsoln <= 0 and at(F10x, asoln) >= 0 then (
                             asoln : flatten ([lam0,[x = 0], asoln]),
                             print ("constraint is non-binding"),
                             print ("soln = ", asoln,"  g(x,y) at soln =
                             ",gsoln),
                             objsub : at( at(func,asoln), x = 0),
                             print ("objsub = ",objsub),
                              print ("soln = ", float (asoln),"  objsub = ", float
                              (objsub)))),

                      if Dsingle (solns) or single (solns) then (
                            if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                            gsoln : at( at(%g,asoln), x = 0),
                         /*   display (gsoln), */
                            if at(y, asoln) > 0 and gsoln <= 0 and at(F10x, asoln) >= 0 then
                            (
                                asoln : flatten ([lam0,[x = 0], asoln]),
                                print ("constraint is non-binding"),
                                print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                                objsub : at( at(func,asoln), x = 0),
                                print ("objsub = ",objsub),
                                print ("soln = ", float (asoln),"  objsub = ",float
                                (objsub)))))),
/* lam = 0, x > 0, y = 0 */
           /* print ("case lam = 0, x > 0, y = 0"), */
           if numberp(fy0) and abs(float (fy0)) < 1e-10 then %n : %n + 1
               /* func close to zero at y = 0; no solution */


           else (
                F10y : subst(0, y, F10),  /* both lam and y are zero here */
                F20y : subst(0, y, F20),
                if (numberp(F10y) and F10y < 0) or (numberp(F20y) and F20y < 0) then %n : %n + 1

                else (
                      solns : solve (F10y, x), /* since x > 0, enforce F10y = 0 */
                    /*  display (solns), */

                      if multiple (solns) then
                             for j thru length(solns) do (
                             asoln : solns[j],
                             gsoln : at(at (%g, asoln), y = 0),
                          /*  display (gsoln), */
                             if at(x,asoln) > 0 and gsoln <= 0 and at(F20y,asoln) >= 0 then (
                                  asoln : flatten ([lam0,asoln, [y = 0]]),
                                  print ("constraint is non-binding"),
                                  print ("soln = ", asoln,"  g(x,y) at soln =
                                  ",gsoln),
                                  objsub : at( at(func,asoln), y = 0),
                                  print ("objsub = ",objsub),
                                  print ("soln = ", float (asoln),"  objsub = ", float
                                  (objsub)))),

                      if Dsingle (solns) or single (solns) then (
                             if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                             gsoln : at( at(%g,asoln), y = 0),
                          /*  display (gsoln), */
                             if at(x,asoln) > 0 and gsoln <= 0 and at(F20y,asoln) >= 0 then (
                                 asoln : flatten ([lam0,asoln, [y = 0]]),
                                 print ("constraint is non-binding"),
                                 print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                                 objsub : at( at(func,asoln), y = 0),
                                 print ("objsub = ",objsub),
                                 print ("soln = ", float (asoln),"  objsub = ",float
                                 (objsub))))))),
```

```
/* lam = 0, x > 0, y > 0 */

                /* print ("case lam = 0, x > 0, y > 0"), */
                solns : solve ([F10,F20], [x, y]),  /* since x>0,y>0, enforce F10=0,F20=0 */
                /* display (solns), */

                if multiple (solns) then
                    for j thru length(solns) do (
                        asoln : solns[j],
                        gsoln : at (%g, asoln),
                        if gsoln <= 0 and at(x,asoln) > 0
                                  and at(y,asoln) > 0 then (
                             asoln : flatten ([lam0,asoln]),
                             print ("constraint is non-binding"),
                             print ("soln = ", asoln,"  g(x,y) at soln =
                             ",gsoln),
                             objsub : at (func, asoln),
                             print ("objsub = ",objsub),
                             print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

                if Dsingle (solns) or single (solns) then (
                    if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                    gsoln : at (%g, asoln),
                    if gsoln <= 0 and at(x,asoln) > 0
                                  and at (y,asoln) > 0 then (
                         asoln : flatten ([lam0,asoln]),
                         print ("constraint is non-binding"),
                         print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                         objsub : at (func, asoln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))))),

/*  lam > 0,  x = 0, y > 0  */

       /* print ("-------------------------------------"), */
       /* print (" case lam > 0, x = 0, y > 0  "), */
       F10 : subst (0,x,LF1),
       F20 : subst (0,x,LF2),  /* only x is zero here, redefining F10, F20 */
       F30 : subst (0,x,LF3),
       /* display (F10, F20, F30), */
       if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
       else if numberp (F10) and F10 < 0 then %n : %n + 1
       else if numberp (F20) and F20 < 0 then %n : %n + 1
       else if numberp (F30) and F30 > 0 then %n : %n + 1

       else (
             solns : solve ( [F20, F30], [y, lam]),    /* since y>0,lam>0, enforce F20=0,F30=0
             */
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      gsoln : at ( at(%g, asoln), x = 0),
                      if at(y,asoln) > 0 and at(lam, asoln) > 0 and  gsoln <= 0
                               and at(F10,asoln) >= 0  then (
                           asoln : flatten ([[x = 0],asoln]),
                           print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                           objsub : at( at(func, asoln), x = 0),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  gsoln : at( at(%g, asoln), x = 0),
                  if at(y,asoln) > 0 and at(lam,asoln) > 0 and gsoln <= 0
```

```
                        and at(F10,asoln) >= 0  then (
                      asoln : flatten ([[x = 0],asoln]),
                      objsub : at(at(func, asoln), x = 0),
                      print ("soln = ", asoln,"  g(x,y) at soln =
                      ",gsoln),
                      print ("objsub = ",objsub),
                      print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/*  lam > 0, x > 0, y = 0 */

       /* print ("---------------------------------------"), */
       /* print (" case  lam > 0, x > 0, y = 0 "), */
       F10 : subst (0,y,LF1),
       F20 : subst (0,y,LF2),  /* only y is zero here, redefining F10,F20,F30 */
       F30 : subst (0,y,LF3),
       /* display (F10, F20, F30), */

       if numberp (fy0) and abs (float (fy0)) < 1e-10 then %n : %n + 1

       else if numberp (F10) and F10 < 0 then %n : %n + 1
       else if numberp (F20) and F20 < 0 then %n : %n + 1

       else if numberp (F30) and F30 > 0 then %n : %n + 1

       else (
             solns : solve ( [F10, F30], [x, lam]),    /*x>0,lam>0, enforce F10=0,F30=0 */
            /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      gsoln : at(at(%g,asoln), y = 0),
                      if at(x,asoln) > 0 and at(lam,asoln) > 0 and  gsoln <= 0
                          and  at(F20,asoln) >= 0  then(
                           asoln : flatten ([asoln, [y = 0]]),
                           objsub : at( at(func,asoln), y = 0),
                           print ("soln = ", asoln,"  g(x,y) at soln =
                           ",gsoln),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float
                           (objsub)))),

             if Dsingle(solns) or single(solns) then (
                   if Dsingle(solns) then asoln : solns[1] else asoln : solns,
                   gsoln : at( at(%g, asoln), y =
                   0),
                   if at(x,asoln) > 0 and at(lam,asoln) > 0 and gsoln <= 0
                             and at(F20,asoln) >= 0    then (
                        asoln : flatten ([asoln, [y = 0]]),
                        objsub : at( at(func,asoln), y = 0),
                        print ("soln = ", asoln,"  g(x,y) at soln =
                        ",gsoln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam > 0,  x > 0, y > 0  */

       /* print ("----------------------------------------"),     */
       /* print (" case   lam > 0, x > 0, y > 0 "), */
       solns : solve ([LF1, LF2, LF3],[x, y, lam]),
         /* display (solns), */

       if multiple (solns) then
               for j thru length(solns) do (
                      asoln : solns[j],
                      gsoln : at (%g, asoln),
                      if gsoln <= 0 and at(x, asoln) > 0 and at(y,asoln) > 0 and
```

```
                                        at(lam,asoln) > 0  then
                                        (
                                objsub : at (func, asoln),
                                print ("soln = ", asoln,"  g(x,y) at soln =
                                ",gsoln),
                                print ("objsub = ",objsub),
                                print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

        if Dsingle(solns) or single(solns) then (
                if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                gsoln : at (%g,asoln),
                if  gsoln <= 0 and  at(x,asoln) > 0 and at(y,asoln) > 0 and
                                    at(lam,asoln) > 0  then (
                                objsub : at (func,asoln),
                                print ("soln = ", asoln,"  g(x,y) at soln = ",gsoln),
                                print ("objsub = ", objsub),
                                print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),
        done )$

/* end of KKTmin (f,g)  */


/*
 KKTmax2(f,g1,g2)
         maximizes f(x,y) subject to g1(x,y) >=0 and g2(x,y) >= 0
*/

KKTmax2(func, %g1, %g2) :=

    block ([%n:0, lam1,lam2, LF, LF1, LF2,LF3,LF4, solns, objsub, fx0, fy0,
            F10, F20, F30,F40, F10x, F20x,F10y, F20y,asoln, g1soln, g2soln,
             lam10 : [lam1 = 0], lam20 : [lam2 = 0],lvf ],

       /* display (func, %g1, %g2), */

            lvf : listofvars (func),
            if lfreeof (lvf,x) and lfreeof(lvf,y) then
           ( print (" the objective  and constraint expressions should
                        depend only on (x,y)"),
             return (done)),

         fx0 : subst(0, x, func),
         fy0 : subst(0, y, func),
          /*  display(fx0, fy0),  */
         LF : func + lam1*%g1 + lam2*%g2,
     /*    display (LF), */
         LF1 : diff(LF,x),
         LF2 : diff(LF,y),
         LF3 : diff(LF,lam1),
         LF4 : diff(LF,lam2),
    /*   display(LF1,LF2, LF3, LF4),  */

/* lam1 = 0, lam2 = 0 cases   */

       /* print ("---------------------------------"),
        print (" case lam1 = 0, lam2 = 0 "), */
        F10 : subst([lam1=0,lam2=0],LF1),
        F20 : subst([lam1=0,lam2=0],LF2),
        /* display (F10, F20),   */
        if (numberp(F10) and F10 > 0) or (numberp(F20) and F20 > 0) then %n : %n + 1
        else (
/* lam1 = 0, lam2 = 0,  x = 0, y > 0 */
            /* print ("case lam1 = 0,lam2 = 0, x = 0, y > 0"), */
            if numberp(fx0) and abs(float (fx0)) < 1e-10 then %n : %n + 1
            /* func close to zero at x = 0; no solution */

              else (
```

```
              F10x : subst(0, x, F10),  /* lam1, lam2, and x are zero here */
              F20x : subst(0, x, F20),
              if (numberp(F10x) and F10x > 0) or (numberp(F20x) and F20x > 0) then %n : %n + 1

              else (
                    solns : solve (F20x, y),  /* since y > 0, we enforce F20x = 0 */
                 /*    display (solns), */

                     if multiple (solns) then
                           for j thru length(solns) do (
                           asoln : solns[j],
                           g1soln : at (at (%g1, asoln), x = 0),
                           g2soln : at (at (%g2, asoln), x = 0),
                        /*  display (g1soln, g2soln), */
                          if at(y, asoln) > 0 and g1soln >= 0 and
                              g2soln >= 0 and at(F10x, asoln) <= 0 then (
                                asoln : flatten ([lam10, lam20, [x = 0], asoln]),
                                print ("constraint is non-binding"),
                                print ("soln = ", asoln,"  g1= ",g1soln,
                                     "   g2 = ", g2soln),
                                objsub : at( at(func,asoln), x = 0),
                                print ("objsub = ",objsub),
                                 print ("soln = ", float (asoln),"  objsub = ", float
                                  (objsub)))),

                     if Dsingle (solns) or single (solns) then (
                           if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                           g1soln : at (at (%g1, asoln), x = 0),
                           g2soln : at (at (%g2, asoln), x = 0),
                        /*   display (g1soln, g2soln), */
                           if at(y, asoln) > 0 and g1soln >= 0 and
                                 g2soln >= 0 and at(F10x, asoln) <= 0 then (
                               asoln : flatten ([lam10, lam20, [x = 0], asoln]),
                               print ("constraint is non-binding"),
                               print ("soln = ", asoln,"  g1= ",g1soln,
                                     "   g2 = ", g2soln),
                               objsub : at( at(func,asoln), x = 0),
                               print ("objsub = ",objsub),
                               print ("soln = ", float (asoln),"  objsub = ",float
                                (objsub))))))),
/* lam1 = 0, lam2 = 0,  x > 0, y = 0 */
            /* print ("case lam1 = 0, lam2 = 0, x > 0, y = 0"), */
            if numberp(fy0) and abs(float (fy0)) < 1e-10 then %n : %n + 1
                /* func close to zero at y = 0; no solution */


            else (
                 F10y : subst(0, y, F10),  /* lam1, lam2 and y are zero here */
                 F20y : subst(0, y, F20),
                 if (numberp(F10y) and F10y > 0) or (numberp(F20y) and F20y > 0) then %n : %n + 1

                 else (
                       solns : solve (F10y, x), /* since x > 0, enforce F10y = 0 */
                     /*  display (solns), */

                       if multiple (solns) then
                             for j thru length(solns) do (
                             asoln : solns[j],
                             g1soln : at(at (%g1, asoln), y = 0),
                             g2soln : at(at (%g2, asoln), y = 0),
                          /*  display (g1soln, g2soln), */
                             if at(x,asoln) > 0 and g1soln >= 0 and
                               g2soln >= 0 and at(F20y,asoln) <= 0 then (
                                  asoln : flatten ([lam10,lam20,asoln, [y = 0]]),
                                  print ("constraint is non-binding"),
                                  print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
```

```
                                objsub : at( at(func,asoln), y = 0),
                                print ("objsub = ",objsub),
                                print ("soln = ", float (asoln),"  objsub = ", float
                                (objsub)))),

                        if Dsingle (solns) or single (solns) then (
                               if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                               g1soln : at(at (%g1, asoln), y = 0),
                               g2soln : at(at (%g2, asoln), y = 0),
                              /*  display (g1soln, g2soln), */
                               if at(x,asoln) > 0 and g1soln >= 0 and
                                     g2soln >= 0 and at(F20y,asoln) <= 0 then (
                                   asoln : flatten ([lam10,lam20, asoln, [y = 0]]),
                                   print ("constraint is non-binding"),
                                   print ("soln = ", asoln,"  g1= ",g1soln,
                                         "   g2 = ", g2soln),
                                   objsub : at( at(func,asoln), y = 0),
                                   print ("objsub = ",objsub),
                                   print ("soln = ", float (asoln),"  objsub = ",float
                                   (objsub)))))),



/* lam1 = 0, lam2 = 0,  x > 0, y > 0 */

                /* print ("case lam1 = 0,lam2 = 0, x > 0, y > 0"), */
                solns : solve ([F10,F20], [x, y]),  /* since x>0,y>0, enforce F10=0,F20=0 */
                /* display (solns), */

                if multiple (solns) then
                    for j thru length(solns) do (
                        asoln : solns[j],
                        g1soln : at (%g1, asoln),
                        g2soln : at (%g2, asoln),
                        if g1soln >= 0 and at(x,asoln) > 0 and g2soln >= 0
                                  and at(y,asoln) > 0 then (
                             asoln : flatten ([lam10, lam20,asoln]),
                             print ("constraint is non-binding"),
                             print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                             objsub : at (func, asoln),
                             print ("objsub = ",objsub),
                             print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

                if Dsingle (solns) or single (solns) then (
                    if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                    g1soln : at (%g1, asoln),
                    g2soln : at (%g2, asoln),
                    if g1soln >= 0 and at(x,asoln) > 0 and g2soln >= 0
                                  and at (y,asoln) > 0 then (
                         asoln : flatten ([lam10, lam20, asoln]),
                         print ("constraint is non-binding"),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                         objsub : at (func, asoln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam1 > 0, lam2 = 0 section  */


/*  lam1 > 0, lam2 = 0,  x = 0, y > 0  */

        /* print ("-------------------------------------"), */
        /* print (" case lam1 > 0, lam2 = 0, x = 0, y > 0  "), */
        F10 : subst ([lam2=0,x=0],LF1),
        F20 : subst ([lam2=0,x=0],LF2),  /* lam2 and x are zero here, redefining F10,
```

```
       F20,F30,F40 */
       F30 : subst ([lam2=0,x=0],LF3),
       F40 : subst ([lam2=0,x=0],LF4),
       /* display (F10, F20, F30,F40), */
       if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
       else if numberp (F10) and F10 > 0 then %n : %n + 1
       else if numberp (F20) and F20 > 0 then %n : %n + 1
       else if numberp (F30) and F30 < 0 then %n : %n + 1
       else if numberp (F40) and F40 < 0 then %n : %n + 1

       else (
             solns : solve ( [F20, F30], [y, lam1]),    /* since y>0,lam1>0, enforce
             F20=0,F30=0 */
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at(at (%g1, asoln), x = 0),
                      g2soln : at(at (%g2, asoln), x = 0),
                      if at(y,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln >= 0 and
                                  g2soln >= 0 and at(F10,asoln) <= 0  then (
                           asoln : flatten ([lam20,[x = 0],asoln]),
                           print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                           objsub : at( at(func, asoln), x = 0),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  g1soln : at(at (%g1, asoln), x = 0),
                  g2soln : at(at (%g2, asoln), x = 0),
                  if at(y,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln >= 0 and
                                  g2soln >= 0 and at(F10,asoln) <= 0  then (
                        asoln : flatten ([lam20,[x = 0],asoln]),
                        objsub : at(at(func, asoln), x = 0),
                        print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 > 0, lam2 = 0, x > 0, y = 0  */

       /* print ("-------------------------------------"), */
       /* print (" case lam1 > 0, lam2 = 0 x > 0, y = 0  "), */
       F10 : subst ([lam2=0,y=0],LF1),
       F20 : subst ([lam2=0,y=0],LF2),  /* lam2 and y  zero here, redefining F10,F20,F30,F40 */
       F30 : subst ([lam2=0,y=0],LF3),
       F40 : subst ([lam2=0,y=0],LF4),
       /* display (F10, F20, F30,F40), */
       if numberp (fy0) and abs (float (fy0)) < 1e-10 then  %n : %n + 1
       else if numberp (F10) and F10 > 0 then %n : %n + 1
       else if numberp (F20) and F20 > 0 then %n : %n + 1
       else if numberp (F30) and F30 < 0 then %n : %n + 1
       else if numberp (F40) and F40 < 0 then %n : %n + 1

       else (
             solns : solve ( [F10, F30], [x, lam1]),    /* since x>0,lam1>0, enforce
             F10=0,F30=0 */
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at(at (%g1, asoln), y = 0),
                      g2soln : at(at (%g2, asoln), y = 0),
```

```
                    if at(x,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln >= 0 and
                                g2soln >= 0 and at(F20,asoln) <= 0 then (
                         asoln : flatten ([lam20,[y = 0],asoln]),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                     "   g2 = ", g2soln),
                         objsub : at( at(func, asoln), y = 0),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

               if Dsingle(solns) or single(solns) then (
                    if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                    g1soln : at(at (%g1, asoln), y = 0),
                    g2soln : at(at (%g2, asoln), y = 0),
                    if at(x,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln >= 0 and
                                g2soln >= 0 and at(F20,asoln) <= 0 then (
                         asoln : flatten ([lam20,[y = 0],asoln]),
                         objsub : at(at(func, asoln), y = 0),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                     "   g2 = ", g2soln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 > 0, lam2 = 0, x > 0, y > 0 */

       /* print ("-------------------------------------"), */
        /* print (" case lam1 > 0, lam2 = 0, x > 0, y > 0  "), */
       F10 : subst (0,lam2,LF1),
       F20 : subst (0,lam2,LF2),  /* only lam2 is zero here, redefining F10,F20  */

       /* display (F10, F20),       */
       if numberp (F10) and F10 > 0 then %n : %n + 1
       else if numberp (F20) and F20 > 0 then %n : %n + 1

       else (   /* since x>0, y>0, lam1>0, enforce F10=0,F20=0, LF3=0 */
             solns : solve ( [F10,F20,LF3], [x,y,lam1]),
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at (%g1, asoln),
                      g2soln : at (%g2, asoln),
                      if at(x,asoln) > 0 and at(y,asoln) > 0 and
                              at(lam1, asoln) > 0 and  g1soln >= 0 and
                                   g2soln >= 0  then (
                         asoln : flatten ([lam20,asoln]),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                     "   g2 = ", g2soln),
                         objsub :  at(func, asoln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  g1soln : at (%g1, asoln),
                  g2soln : at (%g2, asoln),
                  if at(x,asoln) > 0 and at(y,asoln) > 0 and
                              at(lam1, asoln) > 0 and  g1soln >= 0 and
                                   g2soln >= 0  then (
                       asoln : flatten ([lam20,asoln]),
                       objsub :  at(func, asoln),
                       print ("soln = ", asoln,"  g1= ",g1soln,
                                   "   g2 = ", g2soln),
                       print ("objsub = ",objsub),
                       print ("soln = ", float (asoln),"  objsub = ", float(objsub)))),
```

```
/* lam1 = 0, lam2 > 0 section */

/* lam1 = 0, lam2 > 0, x = 0, y > 0 */


        /* print ("-------------------------------------"), */
        /* print (" case lam1 = 0, lam2 > 0, x = 0, y > 0  "), */
        F10 : subst ([lam1=0,x=0],LF1),
        F20 : subst ([lam1=0,x=0],LF2),  /* lam1 and x  zero here, redefining F10, F20, F30,
        F40 */
        F30 : subst ([lam1=0,x=0],LF3),
        F40 : subst ([lam1=0,x=0],LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
        else if numberp (F10) and F10 > 0 then %n : %n + 1
        else if numberp (F20) and F20 > 0 then %n : %n + 1
        else if numberp (F30) and F30 < 0 then %n : %n + 1
        else if numberp (F40) and F40 < 0 then %n : %n + 1

        else (
              solns : solve ( [F20, F40], [y, lam2]),    /* since y>0,lam2>0, enforce
              F20=0,F40=0 */
              /* display (solns),   */

              if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
                       g1soln : at(at (%g1, asoln), x = 0),
                       g2soln : at(at (%g2, asoln), x = 0),
                       if at(y,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln >= 0 and
                                    g2soln >= 0 and at(F10,asoln) <= 0  then (
                            asoln : flatten ([lam10,[x = 0],asoln]),
                            print ("soln = ", asoln,"  g1= ",g1soln,
                                         "   g2 = ", g2soln),
                            objsub : at( at(func, asoln), x = 0),
                            print ("objsub = ",objsub),
                            print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

              if Dsingle(solns) or single(solns) then (
                   if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                   g1soln : at(at (%g1, asoln), x = 0),
                   g2soln : at(at (%g2, asoln), x = 0),
                   if at(y,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln >= 0 and
                                    g2soln >= 0 and at(F10,asoln) <= 0  then (
                         asoln : flatten ([lam10,[x = 0],asoln]),
                         objsub : at(at(func, asoln), x = 0),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))))),

/* lam1 = 0, lam2 > 0, x > 0, y = 0 */

        /* print ("-------------------------------------"), */
        /* print (" case lam1 = 0, lam2 > 0 x > 0, y = 0  "), */
        F10 : subst ([lam1=0,y=0],LF1),
        F20 : subst ([lam1=0,y=0],LF2),  /* lam1 and y  zero here, redefining F10,F20,F30,F40 */
        F30 : subst ([lam1=0,y=0],LF3),
        F40 : subst ([lam1=0,y=0],LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fy0) and abs (float (fy0)) < 1e-10 then  %n : %n + 1
        else if numberp (F10) and F10 > 0 then %n : %n + 1
        else if numberp (F20) and F20 > 0 then %n : %n + 1
        else if numberp (F30) and F30 < 0 then %n : %n + 1
        else if numberp (F40) and F40 < 0 then %n : %n + 1

        else (
```

```
        solns : solve ( [F10, F40], [x, lam2]),    /* since x>0,lam2>0, enforce
        F10=0,F40=0 */
        /* display (solns),   */

        if multiple (solns) then
             for j thru length(solns) do (
                 asoln : solns[j],
                 g1soln : at(at (%g1, asoln), y = 0),
                 g2soln : at(at (%g2, asoln), y = 0),
                 if at(x,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln >= 0 and
                             g2soln >= 0 and at(F20,asoln) <= 0  then (
                      asoln : flatten ([lam10,[y = 0],asoln]),
                      print ("soln = ", asoln,"  g1= ",g1soln,
                                  "   g2 = ", g2soln),
                      objsub : at( at(func, asoln), y = 0),
                      print ("objsub = ",objsub),
                      print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

        if Dsingle(solns) or single(solns) then (
             if Dsingle (solns) then asoln : solns[1] else asoln : solns,
             g1soln : at(at (%g1, asoln), y = 0),
             g2soln : at(at (%g2, asoln), y = 0),
             if at(x,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln >= 0 and
                             g2soln >= 0 and at(F20,asoln) <= 0  then (
                   asoln : flatten ([lam10,[y = 0],asoln]),
                   objsub : at(at(func, asoln), y = 0),
                   print ("soln = ", asoln,"  g1= ",g1soln,
                                  "   g2 = ", g2soln),
                   print ("objsub = ",objsub),
                   print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 = 0, lam2 > 0, x > 0, y > 0 */

       /* print ("-------------------------------------"), */
        /* print (" case lam1 = 0, lam2 > 0, x > 0, y > 0  "), */
       F10 : subst (0,lam1,LF1),
       F20 : subst (0,lam1,LF2),  /* only lam1 is zero here, redefining F10,F20  */

       /* display (F10, F20),       */
       if numberp (F10) and F10 > 0 then %n : %n + 1
       else if numberp (F20) and F20 > 0 then %n : %n + 1

       else (   /* since x>0, y>0, lam2>0, enforce F10=0,F20=0,LF4=0 */
             solns : solve ( [F10,F20,LF4], [x,y,lam2]),
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at (%g1, asoln),
                      g2soln : at (%g2, asoln),
                      if at(x,asoln) > 0 and at(y,asoln) > 0 and
                              at(lam2, asoln) > 0 and  g1soln >= 0 and
                                   g2soln >= 0  then (
                           asoln : flatten ([lam10,asoln]),
                           print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                           objsub :  at(func, asoln),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  g1soln : at (%g1, asoln),
                  g2soln : at (%g2, asoln),
                  if at(x,asoln) > 0 and at(y,asoln) > 0 and
                              at(lam2, asoln) > 0 and  g1soln >= 0 and
```

```
                              g2soln >= 0  then (
                        asoln : flatten ([lam10,asoln]),
                        objsub :  at(func, asoln),
                        print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam1 > 0, lam2 > 0 section */

/* lam1 > 0, lam2 > 0, x = 0, y > 0  */

        /* print ("-------------------------------------"), */
        /* print (" case lam1 > 0, lam2 > 0, x = 0, y > 0  "), */
        F10 : subst (0,x,LF1),
        F20 : subst (0,x,LF2),  /* only x is zero here, redefining F10, F20, F30, F40 */
        F30 : subst (0,x,LF3),
        F40 : subst (0,x,LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
        else if numberp (F10) and F10 > 0 then %n : %n + 1
        else if numberp (F20) and F20 > 0 then %n : %n + 1
        else if numberp (F30) and F30 < 0 then %n : %n + 1
        else if numberp (F40) and F40 < 0 then %n : %n + 1

        else (       /* since y>0,lam1>0,lam2>0, enforce F20=0, F30=0, F40=0 */
              solns : solve ( [F20, F30, F40], [y,lam1,lam2]),
              /* display (solns),   */

              if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
                       g1soln : at(at (%g1, asoln), x = 0),
                       g2soln : at(at (%g2, asoln), x = 0),
                       if at(y,asoln) > 0 and at(lam1, asoln) > 0
                       and at(lam2, asoln) > 0 and  g1soln >= 0 and
                                   g2soln >= 0 and at(F10,asoln) <= 0  then (
                            asoln : flatten ([[x = 0],asoln]),
                            print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                            objsub : at( at(func, asoln), x = 0),
                            print ("objsub = ",objsub),
                            print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

              if Dsingle(solns) or single(solns) then (
                   if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                   g1soln : at(at (%g1, asoln), x = 0),
                   g2soln : at(at (%g2, asoln), x = 0),
                   if at(y,asoln) > 0 and at(lam1, asoln) > 0 and at(lam2,asoln) > 0
                   and g1soln >= 0 and g2soln >= 0 and at(F10,asoln) <= 0  then (
                         asoln : flatten ([[x = 0],asoln]),
                         objsub : at(at(func, asoln), x = 0),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam1 > 0, lam2 > 0, x > 0, y = 0 */

        /* print ("-------------------------------------"), */
        /* print (" case lam1 > 0, lam2 > 0, x > 0, y = 0  "), */
        F10 : subst (0,y,LF1),
        F20 : subst (0,y,LF2),  /* only y is zero here, redefining F10, F20, F30, F40 */
        F30 : subst (0,y,LF3),
        F40 : subst (0,y,LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fy0) and abs (float (fy0)) < 1e-10 then  %n : %n + 1
```

```
      else if numberp (F10) and F10 > 0 then %n : %n + 1
      else if numberp (F20) and F20 > 0 then %n : %n + 1
      else if numberp (F30) and F30 < 0 then %n : %n + 1
      else if numberp (F40) and F40 < 0 then %n : %n + 1

      else (       /* since x>0,lam1>0,lam2>0, enforce F10=0, F30=0, F40=0 */
            solns : solve ( [F10, F30, F40], [x,lam1,lam2]),
            /* display (solns),   */

            if multiple (solns) then
                 for j thru length(solns) do (
                     asoln : solns[j],
                     g1soln : at(at (%g1, asoln), y = 0),
                     g2soln : at(at (%g2, asoln), y = 0),
                     if at(x,asoln) > 0 and at(lam1, asoln) > 0
                     and at(lam2, asoln) > 0 and  g1soln >= 0 and
                                 g2soln >= 0 and at(F20,asoln) <= 0  then (
                          asoln : flatten ([[y = 0],asoln]),
                          print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                          objsub : at( at(func, asoln), y = 0),
                          print ("objsub = ",objsub),
                          print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

            if Dsingle(solns) or single(solns) then (
                 if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                 g1soln : at(at (%g1, asoln), y = 0),
                 g2soln : at(at (%g2, asoln), y = 0),
                 if at(x,asoln) > 0 and at(lam1, asoln) > 0 and at(lam2,asoln) > 0
                 and g1soln >= 0 and g2soln >= 0 and at(F20,asoln) <= 0  then (
                       asoln : flatten ([[y = 0],asoln]),
                       objsub : at(at(func, asoln), y = 0),
                       print ("soln = ", asoln,"  g1= ",g1soln,
                                   "   g2 = ", g2soln),
                       print ("objsub = ",objsub),
                       print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 > 0, lam2 > 0, x > 0, y > 0
*/


      /* print ("-----------------------------------------"),     */
      /* print (" case lam1 > 0, lam2 > 0, x > 0, y > 0 "), */
      solns : solve ([LF1, LF2, LF3, LF4],[x, y, lam1, lam2]),
        /* display (solns), */

      if multiple (solns) then
              for j thru length(solns) do (
                     asoln : solns[j],
                     g1soln : at(%g1, asoln),
                     g2soln : at (%g2, asoln),
                     if g1soln >= 0 and  g2soln >= 0 and at(x, asoln) > 0
                         and at(y,asoln) > 0 and at(lam1,asoln) > 0
                                 and at(lam2,asoln) > 0  then
                                  (
                          objsub : at (func, asoln),
                          print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                          print ("objsub = ",objsub),
                          print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

      if Dsingle(solns) or single(solns) then (
              if Dsingle (solns) then asoln : solns[1] else asoln : solns,
              g1soln : at(%g1, asoln),
              g2soln : at (%g2, asoln),
              if g1soln >= 0 and  g2soln >= 0 and at(x, asoln) > 0
                        and at(y,asoln) > 0 and at(lam1,asoln) > 0
```

```
                                    and at(lam2,asoln) > 0  then (
                                objsub : at (func,asoln),
                                print ("soln = ", asoln,"  g1= ",g1soln,
                                          "   g2 = ", g2soln),
                                print ("objsub = ", objsub),
                                print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),
        done )$

/* end of KKTmax2 (f,g1,g2) */



/*
      KKTmin2(f,g1,g2)
       minimizes f(x,y) subject to g1(x,y) <=0 and g2(x,y) <= 0
*/


KKTmin2(func, %g1, %g2) :=

    block ([%n:0, lam1,lam2, LF, LF1, LF2,LF3,LF4, solns, objsub, fx0, fy0,
            F10, F20, F30,F40, F10x, F20x,F10y, F20y,asoln, g1soln, g2soln,
             lam10 : [lam1 = 0], lam20 : [lam2 = 0],lvf ],

            lvf : listofvars (func),
            if lfreeof (lvf,x) and lfreeof(lvf,y) then
           ( print (" the objective  and constraint expressions should
                        depend only on (x,y)"),
             return (done)),

          fx0 : subst(0, x, func),
          fy0 : subst(0, y, func),
          /* display(fx0, fy0), */
          LF : func + lam1*%g1 + lam2*%g2,
     /*    display (LF), */
          LF1 : diff(LF,x),
          LF2 : diff(LF,y),
          LF3 : diff(LF,lam1),
          LF4 : diff(LF,lam2),
    /*   display(LF1,LF2, LF3, LF4),  */

/* lam1 = 0, lam2 = 0 cases   */

         /* print ("---------------------------------"),
          print (" case lam1 = 0, lam2 = 0 "), */
          F10 : subst([lam1=0,lam2=0],LF1),
          F20 : subst([lam1=0,lam2=0],LF2),
          /* display (F10, F20),   */
          if (numberp(F10) and F10 < 0) or (numberp(F20) and F20 < 0) then %n : %n + 1
          else (
/* lam1 = 0, lam2 = 0,  x = 0, y > 0 */
             /* print ("case lam1 = 0,lam2 = 0, x = 0, y > 0"), */
             if numberp(fx0) and abs(float (fx0)) < 1e-10 then %n : %n + 1
             /* func close to zero at x = 0; no solution */

               else (
                  F10x : subst(0, x, F10),  /* lam1, lam2, and x are zero here */
                  F20x : subst(0, x, F20),
                  if (numberp(F10x) and F10x < 0) or (numberp(F20x) and F20x < 0) then %n : %n + 1

                  else (
                        solns : solve (F20x, y),  /* since y > 0, we enforce F20x = 0 */
                     /*    display (solns), */

                         if multiple (solns) then
                              for j thru length(solns) do (
                              asoln : solns[j],
```

```
                        g1soln : at (at (%g1, asoln), x = 0),
                        g2soln : at (at (%g2, asoln), x = 0),
                       if at(y, asoln) > 0 and g1soln <= 0 and
                           g2soln <= 0 and at(F10x, asoln) >= 0 then (
                             asoln : flatten ([lam10, lam20, [x = 0], asoln]),
                             print ("constraint is non-binding"),
                             print ("soln = ", asoln,"  g1= ",g1soln,
                                  "   g2 = ", g2soln),
                             objsub : at( at(func,asoln), x = 0),
                             print ("objsub = ",objsub),
                              print ("soln = ", float (asoln),"  objsub = ", float
                              (objsub)))),

                    if Dsingle (solns) or single (solns) then (
                          if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                          g1soln : at (at (%g1, asoln), x = 0),
                          g2soln : at (at (%g2, asoln), x = 0),
                          if at(y, asoln) > 0 and g1soln <= 0 and
                                g2soln <= 0 and at(F10x, asoln) >= 0 then (
                              asoln : flatten ([lam10, lam20, [x = 0], asoln]),
                              print ("constraint is non-binding"),
                              print ("soln = ", asoln,"  g1= ",g1soln,
                                    "   g2 = ", g2soln),
                              objsub : at( at(func,asoln), x = 0),
                              print ("objsub = ",objsub),
                              print ("soln = ", float (asoln),"  objsub = ",float
                              (objsub))))))),

/* lam1 = 0, lam2 = 0,  x > 0, y = 0 */

           /* print ("case lam1 = 0, lam2 = 0, x > 0, y = 0"), */
           if numberp(fy0) and abs(float (fy0)) < 1e-10 then %n : %n + 1
               /* func close to zero at y = 0; no solution */


           else (
                F10y : subst(0, y, F10),  /* lam1, lam2 and y are zero here */
                F20y : subst(0, y, F20),
                if (numberp(F10y) and F10y < 0) or (numberp(F20y) and F20y < 0) then %n : %n + 1

                else (
                      solns : solve (F10y, x), /* since x > 0, enforce F10y = 0 */
                     /*  display (solns), */

                      if multiple (solns) then
                             for j thru length(solns) do (
                             asoln : solns[j],
                             g1soln : at(at (%g1, asoln), y = 0),
                             g2soln : at(at (%g2, asoln), y = 0),
                             if at(x,asoln) > 0 and g1soln <= 0 and
                               g2soln <= 0 and at(F20y,asoln) >= 0 then (
                                  asoln : flatten ([lam10,lam20,asoln, [y = 0]]),
                                  print ("constraint is non-binding"),
                                  print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                                  objsub : at( at(func,asoln), y = 0),
                                  print ("objsub = ",objsub),
                                  print ("soln = ", float (asoln),"  objsub = ", float
                                  (objsub)))),

                      if Dsingle (solns) or single (solns) then (
                             if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                             g1soln : at(at (%g1, asoln), y = 0),
                             g2soln : at(at (%g2, asoln), y = 0),
                            /*  display (g1soln, g2soln), */
                             if at(x,asoln) > 0 and g1soln <= 0 and
                                   g2soln <= 0 and at(F20y,asoln) >= 0 then (
```

```
                                asoln : flatten ([lam10,lam20, asoln, [y = 0]]),
                                print ("constraint is non-binding"),
                                print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                                objsub : at( at(func,asoln), y = 0),
                                print ("objsub = ",objsub),
                                print ("soln = ", float (asoln),"  objsub = ",float
                                (objsub))))))),



/* lam1 = 0, lam2 = 0,  x > 0, y > 0 */

                /* print ("case lam1 = 0,lam2 = 0, x > 0, y > 0"), */
                solns : solve ([F10,F20], [x, y]),  /* since x>0,y>0, enforce F10=0,F20=0 */
                /* display (solns), */

                if multiple (solns) then
                    for j thru length(solns) do (
                        asoln : solns[j],
                        g1soln : at (%g1, asoln),
                        g2soln : at (%g2, asoln),
                        if g1soln <= 0 and at(x,asoln) > 0 and g2soln <= 0
                                 and at(y,asoln) > 0 then (
                             asoln : flatten ([lam10, lam20,asoln]),
                             print ("constraint is non-binding"),
                             print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                             objsub : at (func, asoln),
                             print ("objsub = ",objsub),
                             print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

                if Dsingle (solns) or single (solns) then (
                    if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                    g1soln : at (%g1, asoln),
                    g2soln : at (%g2, asoln),
                    if g1soln <= 0 and at(x,asoln) > 0 and g2soln <= 0
                                  and at (y,asoln) > 0 then (
                         asoln : flatten ([lam10, lam20, asoln]),
                         print ("constraint is non-binding"),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                         objsub : at (func, asoln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam1 > 0, lam2 = 0 section */

/*  lam1 > 0, lam2 = 0,  x = 0, y > 0  */

        /* print ("--------------------------------------"), */
        /* print (" case lam1 > 0, lam2 = 0 x = 0, y > 0  "), */
        F10 : subst ([lam2=0,x=0],LF1),
        F20 : subst ([lam2=0,x=0],LF2),  /* lam2 and x  zero here, redefining F10, F20 */
        F30 : subst ([lam2=0,x=0],LF3),
        F40 : subst ([lam2=0,x=0],LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
        else if numberp (F10) and F10 < 0 then %n : %n + 1
        else if numberp (F20) and F20 < 0 then %n : %n + 1
        else if numberp (F30) and F30 > 0 then %n : %n + 1
        else if numberp (F40) and F40 > 0 then %n : %n + 1

        else (
              solns : solve ( [F20, F30], [y, lam1]),    /* since y>0,lam1>0, enforce
              F20=0,F30=0 */
              /* display (solns),   */
```

```
        if multiple (solns) then
             for j thru length(solns) do (
                 asoln : solns[j],
                 g1soln : at(at (%g1, asoln), x = 0),
                 g2soln : at(at (%g2, asoln), x = 0),
                 if at(y,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln <= 0 and
                             g2soln <= 0 and at(F10,asoln) >= 0  then (
                      asoln : flatten ([lam20,[x = 0],asoln]),
                      print ("soln = ", asoln,"  g1= ",g1soln,
                                  "   g2 = ", g2soln),
                      objsub : at( at(func, asoln), x = 0),
                      print ("objsub = ",objsub),
                      print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

        if Dsingle(solns) or single(solns) then (
             if Dsingle (solns) then asoln : solns[1] else asoln : solns,
             g1soln : at(at (%g1, asoln), x = 0),
             g2soln : at(at (%g2, asoln), x = 0),
             if at(y,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln <= 0 and
                             g2soln <= 0 and at(F10,asoln) >= 0  then (
               asoln : flatten ([lam20,[x = 0],asoln]),
               objsub : at(at(func, asoln), x = 0),
               print ("soln = ", asoln,"  g1= ",g1soln,
                             "   g2 = ", g2soln),
               print ("objsub = ",objsub),
               print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 > 0, lam2 = 0, x > 0, y = 0  */

       /* print ("--------------------------------------"), */
       /* print (" case lam1 > 0, lam2 = 0 x > 0, y = 0  "), */
       F10 : subst ([lam2=0,y=0],LF1),
       F20 : subst ([lam2=0,y=0],LF2),  /* lam2 and y  zero here, redefining F10,F20,F30,F40 */
       F30 : subst ([lam2=0,y=0],LF3),
       F40 : subst ([lam2=0,y=0],LF4),
       /* display (F10, F20, F30,F40), */
       if numberp (fy0) and abs (float (fy0)) < 1e-10 then  %n : %n + 1
       else if numberp (F10) and F10 < 0 then %n : %n + 1
       else if numberp (F20) and F20 < 0 then %n : %n + 1
       else if numberp (F30) and F30 > 0 then %n : %n + 1
       else if numberp (F40) and F40 > 0 then %n : %n + 1

       else (
             solns : solve ( [F10, F30], [x, lam1]),    /* since x>0,lam1>0, enforce
             F10=0,F30=0 */
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at(at (%g1, asoln), y = 0),
                      g2soln : at(at (%g2, asoln), y = 0),
                      if at(x,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln <= 0 and
                                  g2soln <= 0 and at(F20,asoln) >= 0 then (
                           asoln : flatten ([lam20,[y = 0],asoln]),
                           print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                           objsub : at( at(func, asoln), y = 0),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  g1soln : at(at (%g1, asoln), y = 0),
                  g2soln : at(at (%g2, asoln), y = 0),
                  if at(x,asoln) > 0 and at(lam1, asoln) > 0 and  g1soln <= 0 and
```

```
                                g2soln <= 0 and at(F20,asoln) >= 0 then (
                        asoln : flatten ([lam20,[y = 0],asoln]),
                        objsub : at(at(func, asoln), y = 0),
                        print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub))))),

/* lam1 > 0, lam2 = 0, x > 0, y > 0  */

        /* print ("-------------------------------------"), */
         /* print (" case lam1 > 0, lam2 = 0, x > 0, y > 0  "), */
        F10 : subst (0,lam2,LF1),
        F20 : subst (0,lam2,LF2),  /* only lam2 is zero here, redefining F10,F20  */

        /* display (F10, F20),       */
        if numberp (F10) and F10 < 0 then %n : %n + 1
        else if numberp (F20) and F20 < 0 then %n : %n + 1

        else (   /* since x>0, y>0, lam1>0, enforce F10=0,F20=0, LF3=0 */
              solns : solve ( [F10,F20,LF3], [x,y,lam1]),
              /* display (solns),   */

              if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
                       g1soln : at (%g1, asoln),
                       g2soln : at (%g2, asoln),
                       if at(x,asoln) > 0 and at(y,asoln) > 0 and
                               at(lam1, asoln) > 0 and  g1soln <= 0 and
                                    g2soln <= 0  then (
                           asoln : flatten ([lam20,asoln]),
                           print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                           objsub :  at(func, asoln),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

              if Dsingle(solns) or single(solns) then (
                   if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                   g1soln : at (%g1, asoln),
                   g2soln : at (%g2, asoln),
                   if at(x,asoln) > 0 and at(y,asoln) > 0 and
                               at(lam1, asoln) > 0 and  g1soln <= 0 and
                                    g2soln <= 0  then (
                         asoln : flatten ([lam20,asoln]),
                         objsub :  at(func, asoln),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ",
                         float(objsub)))),


/* lam1 = 0, lam2 > 0 section */


/* lam1 = 0, lam2 > 0, x = 0, y > 0 */


        /* print ("-------------------------------------"), */
        /* print (" case lam1 = 0, lam2 > 0, x = 0, y > 0  "), */
        F10 : subst ([lam1=0,x=0],LF1),
        F20 : subst ([lam1=0,x=0],LF2),  /* lam1 and x  zero here, redefining F10, F20, F30,
        F40 */
        F30 : subst ([lam1=0,x=0],LF3),
        F40 : subst ([lam1=0,x=0],LF4),
```

```
      /* display (F10, F20, F30,F40), */
      if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
      else if numberp (F10) and F10 < 0 then %n : %n + 1
      else if numberp (F20) and F20 < 0 then %n : %n + 1
      else if numberp (F30) and F30 > 0 then %n : %n + 1
      else if numberp (F40) and F40 > 0 then %n : %n + 1

      else (
            solns : solve ( [F20, F40], [y, lam2]),    /* since y>0,lam2>0, enforce
            F20=0,F40=0 */
            /* display (solns),   */

            if multiple (solns) then
                 for j thru length(solns) do (
                     asoln : solns[j],
                     g1soln : at(at (%g1, asoln), x = 0),
                     g2soln : at(at (%g2, asoln), x = 0),
                     if at(y,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln <= 0 and
                                 g2soln <= 0 and at(F10,asoln) >= 0  then (
                          asoln : flatten ([lam10,[x = 0],asoln]),
                          print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                          objsub : at( at(func, asoln), x = 0),
                          print ("objsub = ",objsub),
                          print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

            if Dsingle(solns) or single(solns) then (
                 if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                 g1soln : at(at (%g1, asoln), x = 0),
                 g2soln : at(at (%g2, asoln), x = 0),
                 if at(y,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln <= 0 and
                                 g2soln <= 0 and at(F10,asoln) >= 0  then (
                       asoln : flatten ([lam10,[x = 0],asoln]),
                       objsub : at(at(func, asoln), x = 0),
                       print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                       print ("objsub = ",objsub),
                       print ("soln = ", float (asoln),"  objsub = ", float (objsub)))))),

/* lam1 = 0, lam2 > 0, x > 0, y = 0 */

      /* print ("-------------------------------------"), */
      /* print (" case lam1 = 0, lam2 > 0, x > 0, y = 0  "), */
      F10 : subst ([lam1=0,y=0],LF1),
      F20 : subst ([lam1=0,y=0],LF2),  /* lam1 and y  zero here, redefining F10,F20,F30,F40 */
      F30 : subst ([lam1=0,y=0],LF3),
      F40 : subst ([lam1=0,y=0],LF4),
      /* display (F10, F20, F30,F40), */
      if numberp (fy0) and abs (float (fy0)) < 1e-10 then  %n : %n + 1
      else if numberp (F10) and F10 < 0 then %n : %n + 1
      else if numberp (F20) and F20 < 0 then %n : %n + 1
      else if numberp (F30) and F30 > 0 then %n : %n + 1
      else if numberp (F40) and F40 > 0 then %n : %n + 1

      else (
            solns : solve ( [F10, F40], [x, lam2]),    /* since x>0,lam2>0, enforce
            F10=0,F40=0 */
            /* display (solns),   */

            if multiple (solns) then
                 for j thru length(solns) do (
                     asoln : solns[j],
                     g1soln : at(at (%g1, asoln), y = 0),
                     g2soln : at(at (%g2, asoln), y = 0),
                     if at(x,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln <= 0 and
                                 g2soln <= 0 and at(F20,asoln) >= 0  then (
                          asoln : flatten ([lam10,[y = 0],asoln]),
```

```
                        print ("soln = ", asoln,"  g1= ",g1soln,
                                    "   g2 = ", g2soln),
                        objsub : at( at(func, asoln), y = 0),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

            if Dsingle(solns) or single(solns) then (
                 if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                 g1soln : at(at (%g1, asoln), y = 0),
                 g2soln : at(at (%g2, asoln), y = 0),
                 if at(x,asoln) > 0 and at(lam2, asoln) > 0 and  g1soln <= 0 and
                                 g2soln <= 0 and at(F20,asoln) >= 0  then (
                       asoln : flatten ([lam10,[y = 0],asoln]),
                       objsub : at(at(func, asoln), y = 0),
                       print ("soln = ", asoln,"  g1= ",g1soln,
                                    "   g2 = ", g2soln),
                       print ("objsub = ",objsub),
                       print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 = 0, lam2 > 0, x > 0, y > 0 */

       /* print ("-------------------------------------"), */
        /* print (" case lam1 = 0, lam2 > 0, x > 0, y > 0  "), */
       F10 : subst (0,lam1,LF1),
       F20 : subst (0,lam1,LF2),  /* only lam1 is zero here, redefining F10,F20  */

       /* display (F10, F20),       */
       if numberp (F10) and F10 < 0 then %n : %n + 1
       else if numberp (F20) and F20 < 0 then %n : %n + 1

       else (   /* since x>0, y>0, lam2>0, enforce F10=0,F20 = 0, LF4=0 */
             solns : solve ( [F10,F20,LF4], [x,y,lam2]),
             /* display (solns),   */

             if multiple (solns) then
                  for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at (%g1, asoln),
                      g2soln : at (%g2, asoln),
                      if at(x,asoln) > 0 and at(y,asoln) > 0 and
                              at(lam2, asoln) > 0 and  g1soln <= 0 and
                                   g2soln <= 0  then (
                          asoln : flatten ([lam10,asoln]),
                          print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                          objsub :  at(func, asoln),
                          print ("objsub = ",objsub),
                          print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

             if Dsingle(solns) or single(solns) then (
                  if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                  g1soln : at (%g1, asoln),
                  g2soln : at (%g2, asoln),
                  if at(x,asoln) > 0 and at(y,asoln) > 0 and
                              at(lam2, asoln) > 0 and  g1soln <= 0 and
                                   g2soln <= 0  then (
                        asoln : flatten ([lam10,asoln]),
                        objsub :  at(func, asoln),
                        print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                        print ("objsub = ",objsub),
                        print ("soln = ", float (asoln),"  objsub = ", float
                        (objsub)))),


/* lam1 > 0, lam2 > 0, x = 0, y > 0  */
```

```
        /* print ("-------------------------------------"), */
        /* print (" case lam1 > 0, lam2 > 0, x = 0, y > 0  "), */
        F10 : subst (0,x,LF1),
        F20 : subst (0,x,LF2),  /* only x is zero here, redefining F10, F20, F30, F40 */
        F30 : subst (0,x,LF3),
        F40 : subst (0,x,LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fx0) and abs (float (fx0)) < 1e-10 then  %n : %n + 1
        else if numberp (F10) and F10 < 0 then %n : %n + 1
        else if numberp (F20) and F20 < 0 then %n : %n + 1
        else if numberp (F30) and F30 > 0 then %n : %n + 1
        else if numberp (F40) and F40 > 0 then %n : %n + 1

        else (       /* since y>0,lam1>0,lam2>0, enforce F20=0, F30=0, F40=0 */
              solns : solve ( [F20, F30, F40], [y,lam1,lam2]),
              /* display (solns),   */

              if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
                       g1soln : at(at (%g1, asoln), x = 0),
                       g2soln : at(at (%g2, asoln), x = 0),
                       if at(y,asoln) > 0 and at(lam1, asoln) > 0
                       and at(lam2, asoln) > 0 and  g1soln <= 0 and
                                  g2soln <= 0 and at(F10,asoln) >= 0  then (
                            asoln : flatten ([[x = 0],asoln]),
                            print ("soln = ", asoln,"  g1= ",g1soln,
                                        "   g2 = ", g2soln),
                            objsub : at( at(func, asoln), x = 0),
                            print ("objsub = ",objsub),
                            print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

              if Dsingle(solns) or single(solns) then (
                   if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                   g1soln : at(at (%g1, asoln), x = 0),
                   g2soln : at(at (%g2, asoln), x = 0),
                   if at(y,asoln) > 0 and at(lam1, asoln) > 0 and at(lam2,asoln) > 0
                   and g1soln <= 0 and g2soln <= 0 and at(F10,asoln) >= 0  then (
                         asoln : flatten ([[x = 0],asoln]),
                         objsub : at(at(func, asoln), x = 0),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                       "   g2 = ", g2soln),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))))),

/* lam1 > 0, lam2 > 0, x > 0, y = 0 */

        /* print ("-------------------------------------"), */
        /* print (" case lam1 > 0, lam2 > 0, x > 0, y = 0  "), */
        F10 : subst (0,y,LF1),
        F20 : subst (0,y,LF2),  /* only y is zero here, redefining F10, F20, F30, F40 */
        F30 : subst (0,y,LF3),
        F40 : subst (0,y,LF4),
        /* display (F10, F20, F30,F40), */
        if numberp (fy0) and abs (float (fy0)) < 1e-10 then  %n : %n + 1
        else if numberp (F10) and F10 < 0 then %n : %n + 1
        else if numberp (F20) and F20 < 0 then %n : %n + 1
        else if numberp (F30) and F30 > 0 then %n : %n + 1
        else if numberp (F40) and F40 > 0 then %n : %n + 1

        else (       /* since x>0,lam1>0,lam2>0, enforce F10=0, F30=0, F40=0 */
              solns : solve ( [F10, F30, F40], [x,lam1,lam2]),
              /* display (solns),   */

              if multiple (solns) then
                   for j thru length(solns) do (
                       asoln : solns[j],
```

```
                    g1soln : at(at (%g1, asoln), y = 0),
                    g2soln : at(at (%g2, asoln), y = 0),
                    if at(x,asoln) > 0 and at(lam1, asoln) > 0
                    and at(lam2, asoln) > 0 and  g1soln <= 0 and
                              g2soln <= 0 and at(F20,asoln) >= 0  then (
                         asoln : flatten ([[y = 0],asoln]),
                         print ("soln = ", asoln,"  g1= ",g1soln,
                                    "   g2 = ", g2soln),
                         objsub : at( at(func, asoln), y = 0),
                         print ("objsub = ",objsub),
                         print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

            if Dsingle(solns) or single(solns) then (
                 if Dsingle (solns) then asoln : solns[1] else asoln : solns,
                 g1soln : at(at (%g1, asoln), y = 0),
                 g2soln : at(at (%g2, asoln), y = 0),
                 if at(x,asoln) > 0 and at(lam1, asoln) > 0 and at(lam2,asoln) > 0
                 and g1soln <= 0 and g2soln <= 0 and at(F20,asoln) >= 0  then (
                       asoln : flatten ([[y = 0],asoln]),
                       objsub : at(at(func, asoln), y = 0),
                       print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                       print ("objsub = ",objsub),
                       print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

/* lam1 > 0, lam2 > 0, x > 0, y > 0
*/


       /* print ("-----------------------------------------"),     */
       /* print (" case lam1 > 0, lam2 > 0, x > 0, y > 0 "), */
       solns : solve ([LF1, LF2, LF3, LF4],[x, y, lam1, lam2]),
         /* display (solns), */

       if multiple (solns) then
               for j thru length(solns) do (
                      asoln : solns[j],
                      g1soln : at(%g1, asoln),
                      g2soln : at (%g2, asoln),
                      if g1soln <= 0 and  g2soln <= 0 and at(x, asoln) > 0
                          and at(y,asoln) > 0 and at(lam1,asoln) > 0
                                 and at(lam2,asoln) > 0  then
                                  (
                           objsub : at (func, asoln),
                           print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                           print ("objsub = ",objsub),
                           print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),

       if Dsingle(solns) or single(solns) then (
               if Dsingle (solns) then asoln : solns[1] else asoln : solns,
               g1soln : at(%g1, asoln),
               g2soln : at (%g2, asoln),
               if g1soln <= 0 and  g2soln <= 0 and at(x, asoln) > 0
                          and at(y,asoln) > 0 and at(lam1,asoln) > 0
                                 and at(lam2,asoln) > 0  then (
                             objsub : at (func,asoln),
                             print ("soln = ", asoln,"  g1= ",g1soln,
                                      "   g2 = ", g2soln),
                             print ("objsub = ", objsub),
                             print ("soln = ", float (asoln),"  objsub = ", float (objsub)))),
       done )$

/* end of KKTmin2(f,g1,g2)  */
```