13.9 Project
Numerical Solution of Lagrange Multiplier Systems

Here we illustrate the use of computer algebra for the numerical solution of Lagrange multiplier systems by revisiting the solution of Example 4 in this section. There we needed to solve the system

$$\begin{aligned} x + y + z &= 12 \\ z &= x^2 + y^2 \\ \lambda + 2\mu x &= 0 \\ \lambda + 2\mu y &= 0 \\ \lambda - \mu &= 1 \end{aligned} \qquad (1)$$

of five equations in the five unknowns $x, y, z,$ and λ, μ. It's simply a matter of knowing how to enter such a system and the appropriate command for its solution.

Using Maple

First we enter our equations and unknowns.

```
eq1 :=   x + y + z = 12:
eq2 :=   z = x^2 + y^2:
eq3 :=   lambda + 2*mu*x = 0:
eq4 :=   lambda + 2*mu*y = 0:
eq5 :=   lambda - mu = 1:

equations :=   {eq1,eq2,eq3,eq4,eq5}:
unknowns  :=  {x,y,z,lambda,mu}:
```

This is a relatively simple polynomial system that Maple's exact **`solve`** command can handle.

```
solve( equations, unknowns );
```

$$\{\mu = \frac{1}{5}, y = -3, x = -3, \lambda = \frac{6}{5}, z = 18\},\ \{\mu = \frac{-1}{5}, y = 2, x = 2, \lambda = \frac{4}{5}, z = 8\}$$

We see here the low point (2, 2, 8) and the high point (–3,–3, 18) on our ellipse.

In case **`solve`** does not yield an exact solution, we can use Maple's **`fsolve`** command for approximate solutions (read **`fsolve`** as "floating point solve"). This

requires that we provide initial guesses in terms of intervals bracketing an expected solution. For the present problem, we can see in Fig. 13.9.8 that there surely is one solution with $0 < x, y < 5$ and another with $-5 < x, y < 0$. Certainly $0 < z < 25$ in both cases. Frequently it's unnecessary to estimate the multipliers accurately. So we proceed with

```
guess1 :=
{x=0..5, y=0..5, z=0..25, lambda=-10..10, mu=-10..10}:
fsolve( equations, unknowns, guess1 );
```

{mu = -.2000000000, x = 2.000000000, y = 2.000000000,
z = 8.000000000, lambda = .8000000000}

```
guess2 :=
{x=-5..0, y=-5..0, z=0..25,lambda=-10..10,mu=-10..10}:
fsolve( equations, unknowns, guess2 );
```

{lambda = 1.200000000, z = 18.00000000, y = -3.000000000,
x = -3.000000000, mu = .2000000000}

and get the same two solutions as before. Evidently the crudity of our guesses did not matter. But in case it's not so easy to make successful guesses, you just "do what you've got to do." Experiment until something works.

Using Mathematica

First we enter our equations and unknowns.

```
eq1 = x + y + z == 12;
eq2 = z == x^2 + y^2;
eq3 = lambda + 2*mu*x == 0;
eq4 = lambda + 2*mu*y == 0;
eq5 = lambda - mu == 1;

equations =   {eq1,eq2,eq3,eq4,eq5};
unknowns  =  {x,y,z,lambda,mu};
```

This is a relatively simple polynomial system that Mathematica's exact **`Solve`** command can handle.

```
Solve[equations, unknowns]
```

$$\{\{z \to 8,\ \lambda \to \frac{4}{5},\ \mu \to -\frac{1}{5},\ x \to 2,\ y \to 2,\},$$
$$\{z \to 18,\ \lambda \to \frac{6}{5},\ \mu \to \frac{1}{5},\ x \to -3,\ y \to -3\}\}$$

We see here the low point (2, 2, 8) and the high point (–3,–3, 18) on our ellipse.

In case **`Solve`** does not yield an exact solution, we can use Mathematica's **`FindRoot`** command for approximate solutions. This requires that we provide initial guesses for the values of the unknowns. For the present problem, we can see in Fig. 13.9.8 that there surely is one solution with $0 < x, y < 5$ and another with $-5 < x, y < 0$. It seems likely that $0 < z < 20$, so we'll start with $z = 10$ in both cases. Frequently it's unnecessary to estimate the multipliers accurately. So we proceed with

```
FindRoot[equations, {x,5}, {y,5}, {z,10},
                    {lambda,0}, {mu,0} ]
```

$\{x \to 2.,\ y \to 2.,\ z \to 8.,\ \text{lambda} \to 0.8,\ \text{mu} \to -0.2\}$

for the first-quadrant solution we saw above, and with

```
FindRoot[equations, {x,-5}, {y,-5}, {z,10},
                    {lambda,0}, {mu,0} ]
```

$\{x \to -3.,\ y \to -3.,\ z \to 18.,\ \text{lambda} \to 1.2,\ \text{mu} \to 0.2\}$

for the same third-quadrant solution. Evidently the crudity of our guesses did not matter. But in case it's not so easy to make successful guesses, you just "do what you've got to do." Experiment until something works.

Using MATLAB

We want to solve five equations of the form $g = 0,\ h = 0,\ p = 0,\ q = 0,\ r = 0$ corresponding to the equations in (1). However, the student edition of MATLAB does not include a function for the solution of systems of equations. Our strategy is therefore to use the MATLAB function **`fminsearch`** to minimize the function

$$F = g^2 + h^2 + p^2 + q^2 + r^2.$$

This function is defined as a function of the vector **`v = [x;y;z;lambda;mu]`** by the m-file **`F.m`** consisting of the lines

```
function w = F(v)
x = v(1);  y = v(2);   z = v(3);
lambda = v(4);   mu = v(5);
```

```
g = x + y + z - 12;
h = x^2 + y^2 - z;
p = lambda + 2*mu*x;
q = lambda + 2*mu*y;
r = lambda - mu - 1;
w = g^2 + h^2 + p^2 + q^2 + r^2;
```

that you can copy/paste into the MATLAB editor and save on your disk. Evidently a minimal point where $F(\mathbf{v}) = 0$ will be a solution to our original system.

Use of **fminsearch** requires that we provide initial guesses for the values of the intervals. For the present problem, we can see in Fig. 13.9.8 that there surely is one solution with $0 < x, y < 5$ and another with $-5 < x, y < 0$. It seems likely that $0 < z < 20,$ so we'll start with $z = 10$. We also note from the last equation in (1) that λ is 1 greater than μ. Frequently it's unnecessary to estimate the multipliers accurately. So we proceed with

```
[soln,w] = fminsearch('F',[2.5;2.5;10;1;0])

Exiting: Maximum number of function evaluations has
been exceeded.  Current function value: 0.671397

soln =
   -2.4952
    2.4238
   12.0843
    0.3301
    0.0030
w =
    0.6714
```

This doesn't look so good, but we can simply apply **fminsearch** again, using the result of our first iteration as the initial guess for the second iteration.

```
» [soln,w] = fminsearch('F',soln)

Optimization terminated successfully:

soln =
    2.0000
    2.0000
    8.0000
    0.8000
   -0.2000
```

```
w =

  2.1869e-009
```

We see here the first-quadrant solution $x = 2,\ y = 2,\ z = 8.$ This is obviously the low point on our ellipse. To find the third-quadrant solution, we start afresh with negative xy-values.

```
[soln,w] = fminsearch('F',[-2.5;-2.5;10;1;0])

Exiting: Maximum number of function evaluations has
been exceeded. Current function value: 0.632715

soln =
   -3.8133
    0.7310
   15.0881
    0.3674
    0.0169
w =
    0.6327

[soln,w] = fminsearch('F',soln)

Optimization terminated successfully:

soln =
   -3.0000
   -3.0000
   18.0000
    1.2000
    0.2000
w =
  1.7355e-009
```

Now we see the third-quadrant solution $x = -3,\ y = -3,\ z = 18$ that provides the high point on the ellipse.

A Two-Ladder Moat Problem

The figure below shows two ladders L_1 and L_2 leaning across two walls of given heights $H_1 = 10$ and $H_2 = 15$ bordering an alligator-filled moat of given width $W = 50.$ The lower ends of the ladders are placed at distances p and q from the walls, so that their upper ends meet at a point (x, y) above the water. The two ladders are supported

in place by each other as well as by the two walls. Let u and v denote the horizontal distances of the point (x, y) from the two walls.

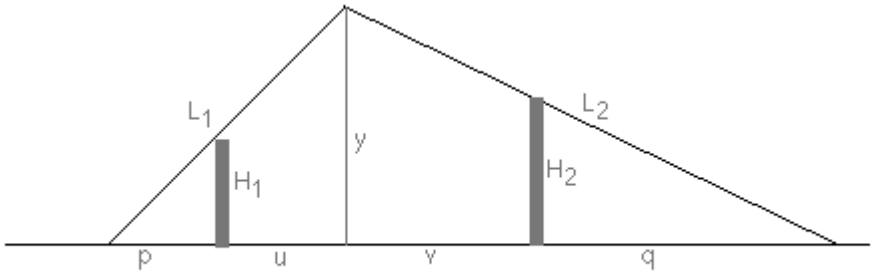

Figure 3

Geometric intuition suggests the existence of a minimum possible value of the sum $L_1 + L_2$ of the lengths of the two ladders. Our problem is to find this minimal sum of ladder lengths. From the figure above we read off the five constraint equations

$$\begin{aligned} u + v &= 50, \\ (p+u)^2 + y^2 &= L_1^2, \\ (q+v)^2 + y^2 &= L_2^2, \\ \frac{y}{p+u} &= \frac{10}{p}, \\ \frac{y}{q+v} &= \frac{15}{q}. \end{aligned}$$

The first of these equations simply records the given width of the moat. The second and third equations are the Pythagorean relations for the two larger right triangles in the figure, while the last two equations are proportionality relations for the two pairs of similar triangles.

We see that our problem involves the seven variables p, q, y, u, v, L_1, and L_2, and we seek to minimize the value of the function

$$F(p, q, y, u, v, L_1, L_2) = L_1 + L_2$$

subject to the five listed constraints. A Mathematica package for the solution of such problems is discussed in C. Henry Edwards, Ladders, Moats, and Lagrange Multipliers, *The Mathematica Journal* 4 (Winter 1994), 48–52. This issue can be downloaded from

http://www.mathsource.com/Content/Publications/Periodicals/TheMathematicaJournal/0207-391