13.10 Project
Critical Point Investigations

Here we use the function

$$f(x,y) = 10\exp\left(-x^2 - \tfrac{1}{2}xy - \tfrac{1}{2}y^2\right)\sin x \sin y \qquad (1)$$

is used to illustrate computer algebra system techniques for the location and classification of critical points of 2-variable functions, as follows:

- First, a surface graph shows the "big picture" that we want to investigate in detail. In Fig. 13.10.20 in the text we see two peaks and two pits, plus an apparent saddle point.

- Next, a contour plot reveals the approximate location of each of these critical points. Fig. 13.10.21 indicates that the approximate locations of the four local extrema are (0.6, 0.75), (–0.6, –0.75), (–0.75, 1), and (0.75, –1), and that the saddle point is at the origin.

- Then we set up the equations $f_x = 0,\ f_y = 0$ and use a CAS **`solve`** command to approximate the critical points accurately — with the known approximate location of each critical point providing an initial guess for its calculation.

- Finally, we compute automatically the second-derivative information need to apply Theorem 2 to classify each critical point. And a contour plot in a small neighborhood of a critical point (a la Figs. 13.10.3–13.10.5 in the text) can provide satisfying visual corroboration of our results.

You can follow this program to investigate a function such as

$$f(x,y) = (ax^2 + 2bxy + cy^2)\exp(-x^2 - y^2) \qquad (2)$$

where a, b, and c are selected integers, or the more exotic function

$$f(x,y) = 10\left(x^3 + y^5 \pm \frac{x}{p}\right)\exp(-x^2 - y^2) + \frac{1}{3}\exp\left(-(x-1)^2 - y^2\right) \qquad (3)$$

where p is a small positive integer. With the plus sign in (3) you are likely to see a half-dozen critical points, but with the minus sign you can expect to see more (as in Fig. 13.10.22 where $p = 5$ and there appears to be some "action" near the origin, in addition to the pairs of pits, peaks, passes that are more clearly visible).

Using Maple

First we define the function in (1),

```
f := (x,y) ->
10*exp(-x^2-0.5*x*y-0.5*y^2)*sin(x)*sin(y):
```

Then the command

```
plot3d(f(x,y), x=-3..3, y=-3..3,
     orientation=[-20,60], style=patch, axes=boxed);
```

plots a surface graph like Fig. 13.10.20 (in the text) where we see two local maxima, two local minima, and a saddle point (evidently at the origin).

The critical points we seek must satisfy the two equations obtained by setting the two first partial derivatives equal to zero.

```
diff(f(x,y),x);

 10(-2x-.5y)e^(-x^2-.5x y-.5y^2) sin(x) sin(y) +
      10 e^(-x^2-.5x y-.5y^2) cos(x) sin(y)
```

Note that when we set $\partial f / \partial x = 0$, we can cancel the (non-zero) exponential throughout. Doing this mentally, the first critical equation we need to solve is

```
eq1 :=
cos(x)*sin(y) + (-2*x-0.5*y)*sin(x)*sin(y) = 0:
```

Similarly,

```
diff(f(x,y),y);

10(-.5x-1.0y)e^(-x^2-.5x y-.5y^2) sin(x) sin(y) +
      10 e^(-x^2-.5x y-.5y^2) sin(x) cos(y)
```

Again canceling the exponential throughout, the second critical equation to be solved is

```
eq2 :=
cos(y)*sin(x) + (-0.5*x-y)*sin(x)*sin(y) = 0:
```

To solve these equations, we need initial estimates of the solutions. The command

```
with(plots):
contourplot(f(x,y),x=-Pi..Pi, y=-Pi..Pi,
          contours=10, coloring=[red,blue]);
```

generates a plot where we can see the four local extrema — one in each quadrant — encircled with level curves. We can "grab" the coordinates of an apparent critical point by pointing to it and clicking the mouse. For instance, the first-quadrant critical point has approximate coordinates (0.6, 0.75). Certainly both coordinates are between 0.5 and 1. We therefore use the "floating point solve" command

```
point1 :=
fsolve({eq1,eq2}, {x,y}, {x=0.5..1.0,y=0.5..1.0});

        point1 := {y = .761116, x = .577336}
```

Thus our first critical point has coordinates

```
x1 := rhs(point1[2]);
y1 := rhs(point1[1]);
                  x1 := .577336
                  y1 := .761116
```

We now proceed to calculate the values of the second derivatives of f at **`point1`**:

```
A := evalf(subs(x=x1,y=y1,diff(f(x,y),x,x)));

                 A1 := -8.68269

B := evalf(subs(x=x1,y=y1,diff(f(x,y),x,y))):
C := evalf(subs(x=x1,y=y1,diff(f(x,y),y,y))):
```

The value of the discriminant Δ of f at **`point1`** is then

```
A*C-B^2;
                      42.9989
```

Since $A < 0$ and $\Delta > 0$, we see that

```
f(x1,y1);
                      1.62083
```

is a local maximum value. You can proceed in the same way to locate and classify the other four critical points.

Using Mathematica

First we define the function in (1),

```
f[x_,y_] := 10 Exp[-x^2-0.5x*y -0.5y^2] Sin[x] Sin[y]
```

Then the command

```
     Plot3D[ f[x, y], {x,-3,3}, {y,-3,3},
             PlotPoints -> {30,30}, PlotRange -> {-3,2}];
```

plots a surface graph like Fig. 13.10.20 (in the text) where we see two local maxima, two local minima, and a saddle point (evidently at the origin).

The critical points we seek must satisfy the two equations obtained by setting the two first partial derivatives equal to zero.

```
     D[f[x,y], x]

     10 E^(-x^2-0.5x y-0.5y^2) Cos[x] Sin[y] +
     10 E^(-x^2-0.5x y-0.5 y^2) (-2x - 0.5y) Sin[x] Sin[y]
```

Note that when we set $\partial f / \partial x = 0,$ we can cancel the (non-zero) exponential throughout. Doing this mentally, the first critical equation we need to solve is

```
     eq1 = Cos[x] Sin[y] + (-2x-0.5y) Sin[x] Sin[y] == 0;
```

Similarly,

```
     D[f[x,y], y]

     10 E^(-x^2-0.5x y - 0.5y^2) Cos[y] Sin[x] +
     10 E^(-x^2-0.5x y-0.5y^2) (-0.5x-1. y) Sin[x] Sin[y]
```

Again canceling the exponential throughout, the second critical equation to be solved is

```
     eq2 = Cos[y] Sin[x] + (-0.5x - y) Sin[x] Sin[y] == 0;
```

To solve these equations, we need initial estimates of the solutions. The command

```
ContourPlot[f[x,y], {x,-3,3}, {y,-3,3},
PlotPoints -> {30, 30}];
```

generates a plot where we can see the four local extrema — one in each quadrant — encircled with level curves. After selecting the figure, we can "grab" the coordinates of an apparent critical point by pointing to it with the cross-hairs while pressing the control (or command key). For instance, the first-quadrant critical point has approximate coordinates $(0.6, 0.75)$. We therefore use the "numerical root-finder" command

```
     point1 = FindRoot[{eq1,eq2}, {x,0.6}, {y,0.75}]
     {x -> 0.577336, y -> 0.761116}
```

So these are the coordinates x_1 and y_1 of our first critical point,

```
x1 = point1[[1,2]]; y1 = point1[[2,2]];
```

We now proceed to calculate the values of the second derivatives of f at **`point1`**:

```
A1 = D[f[x,y], x,x] /. {x -> x1, y -> y1}
-8.68269

B1 = D[f[x,y], x,y] /. {x -> x1, y -> y1};
C1 = D[f[x,y], y,y] /. {x -> x1, y -> y1};
```

The value of the discriminant of f at **`point1`** is then

```
A1*C1 - B1^2
42.9989
```

Since $A < 0$ and $\Delta > 0$, we see that

```
f[x1,y1]
1.62083
```

is a local maximum value. You can proceed in the same way to locate and classify the other four critical points.

Using MATLAB

First we define the function in (1) as a symbolic expression,

```
syms x y
f = 10*exp(-x^2-0.5*x*y-0.5*y^2)*sin(x)*sin(y);
```

First we plot the surface to see what it looks like. For this purpose it's more convenient to use numeric vectors than symbolic expressions.

```
x = -3 : 1/5 : 3;
y = x;
[x,y] = meshgrid(x,y);
```

These commands set up a 31-by-31 grid of *xy*-points.

```
z = 10*exp(-x.^2-1/2*x.*y-1/2*y.^2).*sin(x).*sin(y);
```

Here we have copy-pasted the above formula for $f(x, y)$, then "vectorized" the expression by inserting dots; **`z`** is the corresponding 31-by-31 matrix of *z*-values. It remains only to plot the surface.

```
surf(x,y,z),  view(15,40)
```

This gives a surface graph like Fig. 13.10.20 (in the text) where we see two local maxima, two local minima, and a saddle point (evidently at the origin). In order to solve the critical point equations, we will need initial estimates of the coordinates of the critical points. The command

```
contour(x,y,z,'b'), grid on, axis('square')
```

generates a plot where we can see the four local extrema — one in each quadrant — encircled with level curves. We can "grab" the coordinates of an apparent critical point by entering the **ginput** command, locating the point with the cross-hairs, and selecting it by pressing Enter. For instance,

```
ginput
ans =
    0.5789    0.7544
```

so the first-quadrant critical point has approximate coordinates $(0.6, 0.75)$. Certainly both coordinates are between 0.5 and 1.

The critical points we seek must satisfy the two equations obtained by setting the two first partial derivatives equal to zero.

```
syms x y
f_x = diff(f,x)

f_x =
10*(-2*x-1/2*y)*exp(-x^2-1/2*x*y-1/2*y^2)*sin(x)*sin(y)
+10*exp(-x^2-1/2*x*y-1/2*y^2)*cos(x)*sin(y)
```

Note that when we set $\partial f / \partial x = 0,$ we can cancel the (non-zero) exponential throughout. Doing this mentally, the first critical equation we need to solve is

```
eq1 = 10*(-2*x-1/2*y)*sin(x)*sin(y)+10*cos(x)*sin(y);
```

(Actually, we copy-pasted the derivative and simply deleted the exponential throughout; for MATLAB's purposes we need not bother to append the **=0** that would make it a "real" equation.) Similarly

```
f_y = diff(f,y)

f_y =
10*(-1/2*x-y)*exp(-x^2-1/2*x*y-1/2*y^2)*sin(x)*sin(y)
+10*exp(-x^2-1/2*x*y-1/2*y^2)*sin(x)*cos(y)
```

Again canceling the exponential throughout, the second critical equation to be solved is

```
eq2 = 10*(-1/2*x-y)*sin(x)*sin(y)+10*sin(x)*cos(y);
```

We must call on the Maple toolbox function **`fsolve`** to solve our critical point equations.

```
syms x y
maple('Digits := 6');

point1 =
maple('fsolve(
{(-20*x-5*y)*sin(x)*sin(y) +10*cos(x)*sin(y),
 (-5*x-10*y)*sin(x)*sin(y)+10*sin(x)*cos(y)},
{x,y}, {x=0.5..1.0,y=0.5..1.0})')

point1 =
{x = .577336, y = .761116}
```

So these are the coordinates x_1 and y_1 of our first critical point,

```
x1 = .577336; y1 = .761116;
```

We now proceed to calculate the values of the second derivatives of f at **`point1`**. For this purpose we must substitute our implicit "double precision" values x_1 and y_1 for the symbolic variables x and y in the definition of f.

```
A1 = double(subs(diff(f_x,x),{x,y},{x1,y1}))
A1 =
   -8.6827

B1 = double(subs(diff(f_x,y),{x,y},{x1,y1}));
C1 = double(subs(diff(f_y,y),{x,y},{x1,y1}));
```

The value of the discriminant of f at **`point1`** is then

```
A1*C1 - B1^2
ans =
    42.9989
```

Thus

```
double(subs(f,{x,y},{x1,y1}))
ans =
    1.6208
```

is a local maximum value. You can proceed in the same way to locate and classify the other four critical points.