Mathematica Project 6 Help

Part 1: Critical Points

We want to find critical points for the function of three variables defined below. To do that, we set the gradient of the function equal to the zero vector. We store the critical points in the variable cpLst so that we can access them in Part 2. Notice that we limit the Solve command to the Reals to avoid picking up any complex roots.

In[ ]:=
```
f[x_, y_, z_] := x^3 - x*y - y^2 - x*z - y*z - z^2
cpLst = Solve[D[f[x, y, z], {{x, y, z}, 1}] == {0, 0, 0}, {x, y, z}, Reals]
```

Out[ ]= $\left\{\left\{x \to -\frac{2}{9}, y \to \frac{2}{27}, z \to \frac{2}{27}\right\}, \{x \to 0, y \to 0, z \to 0\}\right\}$

Notice that f has two critical points which are stored as replacement lists. We can access the individual critical points by asking Mathematica to return one of the two components. The code for that is as follows.

In[ ]:=
```
cpLst[[1]]
```

Out[ ]= $\left\{x \to -\frac{2}{9}, y \to \frac{2}{27}, z \to \frac{2}{27}\right\}$

Notice the double brackets! Also notice that Mathematica numbers the items in a list starting at 1. If you've taken much programming, you know that most languages use 0 to refer to the first item in a list. Mathematica is no different, but it turns out the 0-th item in any object is a description of what that item is.

In[ ]:=
```
cpLst[[0]]
```

Out[ ]=
```
List
```

Almost every object in Mathematica has a descriptor as its 0-th item.

Part 2: Classification of Critical Points

To classify the two equilibria we found above, we need to find the eigenvalues of the Hessian matrix. We begin by creating a function that will do just that.

In[ ]:=
```
eVals[P_] := N[Eigenvalues[D[f[x, y, z], {{x, y, z}, 2}] /. P]]
```

The outer N[...] function just instructs Mathematica to return numerical (i.e. approximate) values for the eigenvalues. Obviously, Eigenvalues[...] computes the eigenvalues of a given matrix. The complicated D[f[x,y,z],{{x,y,z},2}] command actually returns the Hessian. That leaves us with the /.P notation. What eVals[P] is expecting is a set of replacement rules for the point (x, y, z) - exactly the sort of thing returned to us from solving for the critical points. We can get the eigenvalues for each of our two critical points easily now!

```
In[ ]:= eVals[cpLst[[1]]]
Out[ ]= {-3.80814, -1., -0.52519}
```

The first critical point in our list from Part 1 is clearly a local maximum since all eigenvalues are negative.

```
In[ ]:= eVals[cpLst[[2]]]
Out[ ]= {-3.56155, -1., 0.561553}
```

The second critical point is a saddle point since we have a mixture of positive and negative eigenvalues.