MATH1072

Practical 3: Constrained and Unconstrained Optimization

Motivation: Optimization is a large branch of numerics that has applications in energy methods, ODE and PDE solving, deep learning, and more. You have been exploring constrained and unconstrained optimization in lecture using analytic expressions. As you can imagine, solving more complex optimization problems by hand can be incredibly time-consuming (or impossible)!
Often in unconstrained optimization, we either have many critical points offering many solutions to our proposed problem, or no solution at all. To address this issue, we constrain our problem to a region or function. For example, suppose we wanted to maximize $y(x) = x$, the solution would be infinity; however, the same problem in a constrained space, say on $x \leq 0$ immediately gives us the solution $y(x) = 0$ as the maximum.

1 Exploring Unconstrained Optimization

You will explore solutions to the following unconstrained optimization problem:

$$\min z(x, y)$$

(a) Type the following into MATLAB to define the function $z(x, y)$ known as the 'peaks' function in MATLAB.

```
z = @(x) 3*(1-x(1)).^2.*exp(-(x(1).^2) - (x(2)+1).^2) -
   10*(x(1)/5 - x(1).^3 - x(2).^5).*exp(-x(1).^2-x(2).^2)
   - 1/3*exp(-(x(1)+1).^2 - x(2).^2);
```

Notice that, while in past practicals we have defined these functions according to individual variables, we have specifically chosen to define our function for this part according to a single, vector variable. This is extremely important for the `fminsearch` setup!

(b) Perform the `fminsearch` by calling the function with a starting value of $(-1, 1)$.

```
xymin = fminsearch() %complete the argument
```

The output of the `fminsearch` function is a vector of (x, y) values where the local minimum of the function $z(x, y)$ has occurred, according to `fminsearch` function.

(c) Use the `fminsearch` function to find the local maximum of the function $z(x, y)$. To do this, note that minimizing $-z(x, y)$ is the same as maximizing $z(x, y)$.

```
negz = @(x) %define the negative of z(x,y), implicitly
xymax = fminsearch() %complete the argument
```

(d) Use the `meshgrid` and `surf` functions to plot $z(x, y)$ as a surface on $[-2, 2] \times [-2, 2]$.

```
Z = @(x,y) 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
   - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
   - 1/3*exp(-(x+1).^2 - y.^2); %defined Z implicitly as
      you have done in previous practicals
[X,Y] = meshgrid(); %complete the argument
surf() %complete the argument
```

(e) Plot the local minima and local maxima found by `fminsearch` on the same plot as (d) using the `plot3` function. Make the point markers filled red circles of size 8.

```
hold on %use the same plot
plot3() %complete the argument
plot3() %complete the argument
```

(f) BONUS: Use the surface plot to choose different initial conditions for finding other local minima and maxima from `fminsearch`.

2 Exploring Constrained Optimization

You will explore solutions to the following constrained optimization problem:

$$\min z(x, y)$$
$$\text{subject to: } 2x + y = 1$$

(a) Create a contour plot of the function $z(x, y) : \mathbb{R}^2 \to \mathbb{R}$ over the region $[-2, 2] \times [-2, 2]$. You can use the predefined X and Y variables from part 1(d).

```
hold off %to stop using the previous figure
contour() %complete the argument
hold on %to start using this figure
```

(b) Plot the constraint function on this plot using the function `fimplicit(@(x,y) func, interval)`. `fimplicit` requires the `func` definition to equal zero on the right hand side.

```
fimplicit(@(x,y) func, interval) %replace "func" and "
    interval" with the correct inputs
```

(c) Perform constrained minimization using the `fmincon` function with the initial condition $(x, y) = (0, 0)$. Plot the solution on your graph. The complete MATLAB code is provided below. Describe the output of `fmincon`.

```
xyminc = fmincon(z,[0,0],[],[],[2,1],1);
plot(xyminc(1),xyminc(2),'ro','MarkerFaceColor','r','
    MarkerSize',8)
```

The inputs of `fmincon` include: the function, the initial condition, inequality constraints, and equality constraints. We will use the `[]` command to avoid inequality constraints.

(d) Perform constrained *maximization* of the same problem using the `fmincon` function with initial condition $(x, y) = (-1, 1)$. Plot the solution on your graph.

```
xymaxc = fmincon(); %complete the argument
plot() %complete the argument
```