AMATH 301 - Spring 2018

Homework #6

Due on Thursday, May 10, 2018

1. The function

$$f(x, y) = (a - x)^2 + b(y - x^2)^2$$

is called Rosenbrock's banana function. It is often used as a benchmarking test for optimization algorithms because it is easy to find the minimum by hand but often very difficult to find numerically. Throughout the problem, we will use the values $a = 2$ and $b = 10$. You can plot this function using the following code:

```
x = -3:0.1:3;
y = -10:0.2:10;
[X,Y] = meshgrid(x,y);
Z = (2-X).^2 + 10*(Y-X.^2).^2;
surf(X,Y,Z)
```

   (a) Write a Matlab function that computes f using only one input (i.e. one variable). Your input will need to be a vector of length 2. Use this function to calculate $f(1, 10)$ and save the result in `A1.dat`.

   (b) Use `fminsearch` to find the minimum of f and save it as a 2×1 column vector `[x; y]` in the file `A2.dat`. Use an initial guess of `[1; 10]`.

   Things to think about: Can you find the minimum of f by hand by taking the partial derivatives and setting them equal to 0? Does this match with what you got from `fminsearch`? Try timing `fminsearch`. How long does it take to find the minimum?

   (c) We will now explore a method called **gradient descent** to find the minimum. To start, find a formula for the gradient $\nabla f(x, y)$. Recall from Calculus that the gradient is defined as the 2×1 vector

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x}(x, y) \\ \frac{\partial f}{\partial y}(x, y) \end{pmatrix}.$$

Calculate $\nabla f(1, 10)$ and save the resulting 2×1 vector in `A3.dat`.

Remember from Calculus that the gradient of f is zero at a minimum. Find the infinity norm of $\nabla f(1, 10)$ and confirm that it is not particularly close to zero. Save the result in `A4.dat`.

(d) The vector $-\nabla f(x, y)$ points from `[x,y]` in the direction where f decreases the fastest. In other words, it points in the direction of steepest descent. Write a function in Matlab defined by

$$\phi(t) = \begin{pmatrix} 1 \\ 10 \end{pmatrix} - t\nabla f(1, 10) = \begin{pmatrix} 1 - t\frac{\partial f}{\partial x}(1, 10) \\ 10 - t\frac{\partial f}{\partial y}(1, 10) \end{pmatrix}$$

Calculate $\phi(0.1)$ and save the resulting 2×1 vector in `A5.dat`. Calculate $f(\phi(0.1))$ and save the resulting number in `A6.dat`.

(e) The function $f(\phi(t))$ is decreasing for small values of t, but that at some point it starts to increase again. This means that $f(\phi(t))$ has a minimum at some $t > 0$. We could implement the golden section search or Newton's method to find this minimum, but we can make things easy by just using the built-in Matlab function `fminbnd`. Use `fminbnd` to minimize $f(\phi(t))$ on the interval $0 < t < 0.1$. Save the resulting t in `A7.dat` and the resulting 2×1 vector $\phi(t)$ in `A8.dat`.

We have just completed one step of the iterative method **gradient descent**. Starting with an initial guess of `[x; y] = [1; 10]`, we found that the gradient of f at this point was not zero, so we needed to calculate a new guess. To find the new guess, we found the direction of steepest descent $-\nabla f(x, y)$. Then we found the minimum value of f in this direction using the `fminbnd` command. The resulting point $\phi(t)$ is our new guess.

Things to think about: The (optional) video on gradient descent (Video 14) shows how to find the minimum of ϕ analytically, but for a different f. Can you do that here instead of using `fminbnd`? Try using Newton's method to find the minimum. Does it run faster? If so, is the boost in speed worth it?

(f) Finally, we will repeat the process from steps (c) - (e) over and over again to find the minimum of f. You can use the following template as a guide:

```
% Make initial guess
% Calculate infinity norm of the gradient

while %(infinity norm of gradient is > tolerance)
     % Make the function φ(t)
     % Find the minimum of f(φ(t)) between t=0 and t=0.1 using fminbnd
     % Make new guess =  φ(t)
     % Calculate infinity norm of the gradient
end
```

You should use a tolerance of 10^{-4} and an initial guess of `[1; 10]`. Save your final guess (which should be a 2×1 vector) in `A9.dat`. Save the total number of steps in `A10.dat`. (The initial guess does not count as a step, but every other new guess does.)

Things to think about: Time your gradient descent algorithm. How does it compare to `fminsearch`? Rosenbrock's banana function is particularly difficult for the gradient descent algorithm. Try some other functions $f(x, y)$ and see how many steps it takes to find the minimum. In particular, try $f(x, y) = x^2 + y^2$. This time minimize $f(\phi(t))$ between $t = 0$ and $t = 1$.

The slow part of the process is where we find the minimum of $f(\phi(t))$ (using `fminbnd`). What happens if you choose a constant t value instead of recalculating this minimum at every step? Be sure to try values that are too large and values that are too small. Try this with Rosenbrock's banana function as well as $f(x, y) = x^2 + y^2$.