MATH693a_HW1

September 21, 2018

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        from scipy import linalg as LA
        import time
        %matplotlib notebook
        %matplotlib inline
```

MATH693a
Dr. Peter Blomgren
Homework 1
Matteo Polimeno
Due Date: 09/19/2018
The Rosenbrock Function is given by

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1}$$

We can easily define this function in Python.

```
In [3]: def rb(x):
            rb = 100*(x[1]-x[0]**2)**2 + (1-x[0])**2
            return rb
```

Its gradient can easily be found by computing the first derivative with respect to x_1 and x_2. For $f'(x_1)$ we have

$$\begin{aligned} f'(x_1) &= 100(4x_1^3 - 4x_2x1) + 2x_1 - 2 && (2)\\ &= 100(4x_1^3 - 4x_2x_1) + 2(x_1 - 1) && (3)\\ &= 400x_1(x_1^2 - x_2) + 2(x_1 - 1); && (4) \end{aligned}$$

and for $f'(x_2)$

$$\begin{aligned} f'(x_2) &= 100(2x_2 - 2x_1^2) && (5)\\ &= 200(x_2 - x_1^2). && (6) \end{aligned}$$

Therefore, the gradient of the Rosenbrock Function is given by

$$\nabla f(x) \;=\; < 400x_1(x_1^2 - x_2) + 2(x_1 - 1),\, 200(x_2 - x_1^2) > . \tag{7}$$

In order to find the minimum, we need to set the gradient equal to zero

$$\nabla f(x) = 0 \tag{8}$$

for which it holds

$$(x_1, x_2) = (1, 1) \tag{9}$$

As for the Hessian, we have

$$f_{x_1x_1} = 1200x_1^2 - 400x_2^2 + 2\,; \tag{10}$$
$$f_{x_1x_2} = -400x_1 = f_{x_2x_1}\,; \tag{11}$$
$$f_{x_2x_2} = 200 \tag{12}$$
$$\tag{13}$$

Thus the Hessian is

$$\begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

We can easily define the gradient and the Hessian in Python and then use them to apply the steepest descent and the Newton method for the backline search algorithm

```
In [4]: def rb_grad(x):
            df1 = 400*x[0]*(x[0]**2-x[1])+2*(x[0]-1)
            df2 = 200*(x[1]-x[0]**2)
            grad = np.array([df1,df2])
            return grad
```

```
In [5]: def hess(x):
            h11 = 1200*x[0]**2-400*x[1]+2
            h12 = -400*x[0]
            h21 = -400*x[0]
            h22 = 200
            hess = np.array([[h11,h12],[h21,h22]])
            return hess
```

```
In [6]: def invhess(x):
            invhess = LA.solve(hess(x),rb_grad(x))
            return invhess
```

```
In [7]: def rb_back_line_search(x,method):
            alpha_bar = 1.
            iiter = 1
            max_iter = 100000
```

```
        tol = 1e-8
        c1 = 1e-4
        rho = .5
        alphavals = []
        pvals = []
        start = time.time()
        while LA.norm(rb_grad(x)) > tol and iiter < max_iter:
            iiter = iiter + 1
            alpha = alpha_bar
            if method=='Newton':
                p = -invhess(x) #search direction for Newton method
            else:
                p = -rb_grad(x)/(LA.norm(rb_grad(x))) #search direction for steepest descent
            pt = np.transpose(p)
            while rb(x+alpha*p) > rb(x)+c1*alpha*(pt).dot(rb_grad(x)):
                alpha = rho*alpha
            xnew = x + alpha*p
            x = xnew
            alpha_k = alpha
            alphavals.append(alpha_k)
            alpha10 = alphavals[0:10]
        end = time.time()
        print("Elapsed time for " + method + " method is " + np.str(end-start))
        print("Minimum found at " + np.str(x))
        print("Value of Rosenbrock Function at minimum is " + np.str(rb(x)))
        print("Search direction is p = " + np.str(p))
        print("First 10 alpha values found to be " + np.str(alpha10))
        print("Number of iterations necessary for convergence is " + np.str(iiter))
```

```
In [8]: rb_back_line_search([1.2,1.2],'steepest descent')
```

```
Elapsed time for steepest descent method is 9.69500017166
Minimum found at [1.         1.00000001]
Value of Rosenbrock Function at minimum is 7.244209187686237e-18
Search direction is p = [ 0.89012922 -0.45570822]
First 10 alpha values found to be [0.125, 0.03125, 0.001953125, 0.00048828125, 0.00048828125, 0.
Number of iterations necessary for convergence is 17067
```

```
In [13]: rb_back_line_search([-1.2,1.],'steepest descent')
```

```
Elapsed time for steepest descent method is 9.42100000381
Minimum found at [1.         0.99999999]
Value of Rosenbrock Function at minimum is 7.033102791657379e-18
Search direction is p = [ 0.89927774 -0.43737804]
First 10 alpha values found to be [0.25, 0.125, 0.0625, 0.0078125, 0.0078125, 0.0078125, 0.00781
Number of iterations necessary for convergence is 17895
```

```
In [11]: rb_back_line_search([1.2,1.2],'Newton')
```

```
Elapsed time for Newton method is 0.00599980354309
Minimum found at [1. 1.]
Value of Rosenbrock Function at minimum is 1.088287359901554e-25
Search direction is p = [-1.77859130e-07 -3.53202652e-07]
First 10 alpha values found to be [1.0, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Number of iterations necessary for convergence is 9
```

```
In [12]: rb_back_line_search([-1.2,1.],'Newton')
```

```
Elapsed time for Newton method is 0.00499987602234
Minimum found at [1. 1.]
Value of Rosenbrock Function at minimum is 3.743975643139474e-21
Search direction is p = [1.11032194e-06 2.49053263e-06]
First 10 alpha values found to be [1.0, 0.125, 1.0, 1.0, 1.0, 0.25, 1.0, 1.0, 1.0, 0.5]
Number of iterations necessary for convergence is 22
```