**MA3236 NONLINEAR PROGRAMMING**

Semester 1, 2018/2019

Assignment 1

Jakkarin Sae-Tiew A0157285W

1. Minimizing the Rosenbrock function using Backtracking Line Search

**Function:**

function [x, iter] = backtracking(impMethod,x0,rho,c,printyes)

… (see attached backtracking.m file)

end

is defined

***For initial point x2 [1.2; 1.2]:***

**Newton Method:**

>> [x, iter] = backtracking('newton',[1.2;1.2], 0.9, 0.6, 1);

iter x1 x2 f(x) step-len

---------------------------------------

0 1.200 1.200 5.800 0.729

1 1.197 1.368 0.462 0.729

2 1.187 1.391 0.066 0.900

3 1.150 1.319 0.023 0.729

4 1.083 1.167 0.010 1.000

5 1.043 1.086 0.002 0.900

6 1.014 1.026 0.000 0.810

7 1.004 1.009 0.000 0.810

8 1.001 1.002 0.000 0.729

9 1.000 1.001 0.000 0.729

10 1.000 1.000 0.000 0.729

---------------------------------------

**Contour and plot of each iterate for Newton Method**

**A close up of a map

Description generated with very high confidenceA close up of a map

Description generated with very high confidence**

**Steepest Descent Method:**

>> [x, iter] = backtracking('steepest descent',[1.2;1.2], 0.9, 0.6, 1);

iter x1 x2 f(x) step-len

---------------------------------------

0 1.200 1.200 5.800 0.001

1 1.135 1.227 0.387 0.001

2 1.117 1.235 0.033 0.001

3 1.113 1.236 0.014 0.001

4 1.112 1.237 0.013 0.001

5 1.112 1.237 0.013 0.002

6 1.112 1.237 0.013 0.002

7 1.112 1.237 0.013 0.001

8 1.112 1.237 0.013 0.002

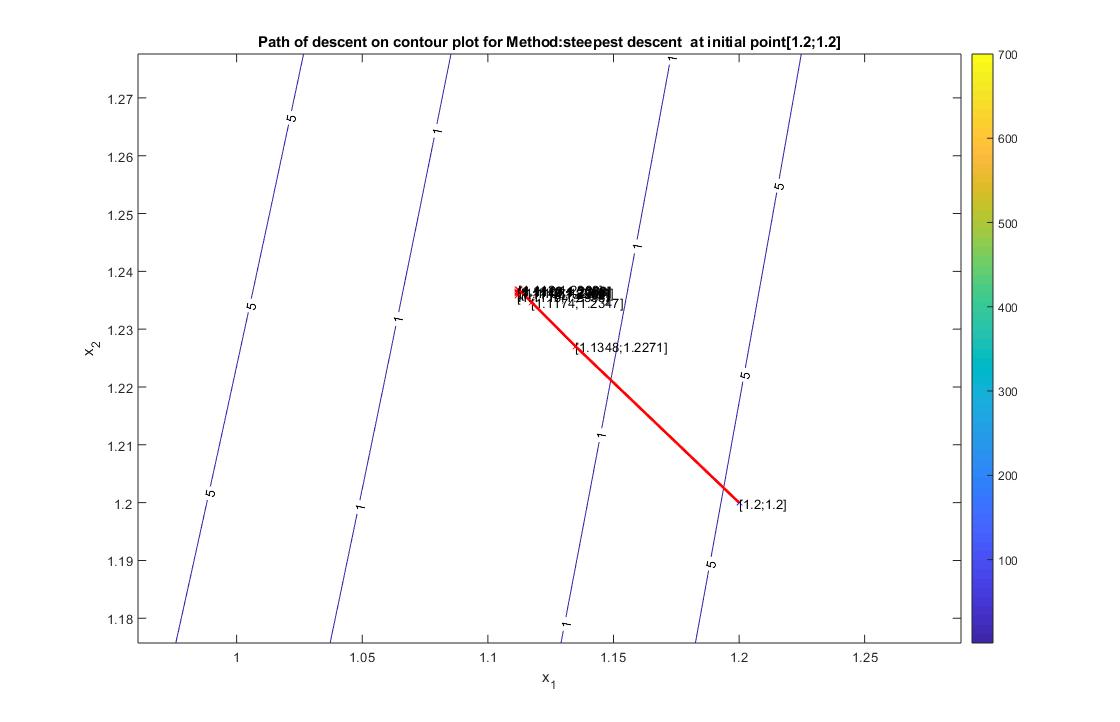
9 1.112 1.236 0.012 0.001

10 1.112 1.236 0.012 0.004

---------------------------------------

**Contour and plot of each iterate for Steepest Descent Method:**

**A close up of a map

Description generated with high confidence**

***For initial point [-1.2; 1]:***

**Newton Method:**

>> [x, iter] = backtracking('newton',[-1.2;1], 0.9, 0.6, 1);

iter x1 x2 f(x) step-len

---------------------------------------

0 -1.200 1.000 24.200 0.729

1 -1.182 1.278 6.191 0.810

2 -1.111 1.207 4.533 0.656

3 -0.900 0.755 3.903 1.000

4 -0.739 0.521 3.091 0.729

5 -0.533 0.234 2.596 1.000

6 -0.392 0.134 1.978 0.656

7 -0.207 0.002 1.626 1.000

8 -0.076 -0.011 1.188 0.656

9 0.083 -0.024 0.939 1.000

10 0.209 0.028 0.651 0.810

---------------------------------------

**Contour and plot of each iterate for Newton Method:**

**A close up of a map

Description generated with very high confidence**

**Steepest Descent Method:**

>> [x, iter] = backtracking('steepest descent',[-1.2;1], 0.9, 0.6, 1);

iter x1 x2 f(x) step-len

---------------------------------------

0 -1.200 1.000 24.200 0.001

1 -1.078 1.050 5.605 0.001

2 -1.041 1.065 4.205 0.001

3 -1.033 1.068 4.133 0.001

4 -1.030 1.068 4.125 0.282

5 -0.814 0.616 3.513 0.001

6 -0.794 0.626 3.221 0.001

7 -0.787 0.627 3.200 0.478

8 -0.140 -0.048 1.756 0.004

9 -0.117 0.003 1.261 0.006

10 -0.102 0.015 1.216 0.034

---------------------------------------

**Contour and plot of each iterate for Steepest Descent Method:**

**A close up of a map

Description generated with high confidence**

1. Run ten iterations of the steepest descent and conjugate gradient algorithms to find approximate minimizers of the quadratic function:

**Steepest Descent Method**

**Functions:**

function [x,p,iter] = steepestDesc(fun)

…

end

function [fx,grad] = quadFn(x, A, b)

fx = 0.5\*x'\*A\*x-b'\*x;

grad = A\*x-b;

end

are defined

**Results:**

>> [x,p,iter] = steepestDesc('quadFn');

iter ||x - x\*||

------------------

0 20.551

1 20.491

2 20.445

3 20.396

4 20.352

5 20.304

6 20.261

7 20.214

8 20.172

9 20.126

**10 20.085**

Therefore,

**Conjugate Gradient Method**

**Functions:**

function [x,p,iter] = conjugateGrad(fun)

… (see attached conjugateGrad.m file)

end

function [fx,grad] = quadFn(x, A, b)

fx = 0.5\*x'\*A\*x-b'\*x;

grad = A\*x-b;

end

are defined

**Results:**

>> [x,p,iter] = conjugateGrad('quadFn');

iter ||x - x\*||

------------------

0 20.551

1 20.491

2 20.331

3 20.068

4 19.727

5 19.383

6 18.622

7 17.375

8 15.797

9 14.264

**10 13.411**

Therefore,