MANE4280 Assignment 2

Jianfeng Yan

October 3, 2017

Question 1

a. func.m

```
function [ f ] = func( x )
  if size(x, 1) ~= 3
  fprintf('size of x should be 3');
  exit;
  end
  A = [41, 12, 12; 12, 36, 32; 12, 32, 36];
  f =  besselj(0, dot(x, A*x));
end
```

b. calc_gradient.m

```
function [ dfdx ] = calc_gradient( func, x )
  dim = size(x, 1);
  dfdx = zeros(dim, 1);
  x0 = x;
  eps = 1.e-12;
  for i = 1 : dim
    x0(i) = x0(i) + eps*1i;
    f0 = func(x0);
    x0(i) = x0(i) - eps*1i;
    dfdx(i) = imag(f0) / eps;
  end
end
```

c

To calculate the gradient, run the following code:

```
x0 = [pi; 1; -exp(1)];
dfdx = calc_gradient(@func, x0)
```

Question 2

The following function satisfies all the requirements:

$$f = (x + y)^2$$

To see this:

- Since $f \geq 0$, and $f(0,0) = 0$, $[0,0]$ is the global minimizer, and hence, a local minimizer.
- $H = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$, which is not diagonal.
- One of its eigenvalue of H is 0, so the Hessian is positive semi-definite, instead of positive definite. Therefore, the second-order sufficient condition is not satisfied.

Question 3

a

The following code solves the constrained optimization problem.

```
param1 = 10;
param2 = 1;
obj = @(x) obj_BB2(x, param1, param2);
x0 = [0;0];
optns = optimoptions(@fmincon, ...
                     'SpecifyObjectiveGradient',true, ...
                     'SpecifyConstraintGradient',true, ...
                     'Display', 'iter', 'Algorithm', 'sqp');

[x, fval] = fmincon(obj, x0, [], [], [], [], [], [], @cnstr_BB2, optns);
```

The minimizer is $x^* = \begin{bmatrix} -1.314362669710740 & -0.209132625285504 \end{bmatrix}^T$.

b

If the design variable does not satisfy the constraints, we can penalize the objective function, that is, make the function value sufficiently large. The penalty can be proportional to $\|c(x)\|$. Of course there are many choices to construct the penalty. We can choose those that are differential.