Assignment 4 of MATP4820

Xinshi Wang 661975305

Problem 1

Consider the unconstrained quadratic minimization problem:

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x} \tag{QuadMin}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, and $\mathbf{b} \in \mathbb{R}^n$.

1. Let $\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$. Set the initial vector $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Do by hand three iterations of the BB method. Your first iteration should be steepest gradient descent with exact line search, and the second and third iterations should use Option-I BB stepsize.

$$x^{(1)} = x^{(0)} - \alpha_0 \nabla f(x^{(0)})$$

$$\alpha_0 = \frac{-(p^{(0)})^T \nabla f(x^{(0)})}{(p^{(0)})^T A^T p^{(0)}}$$

$$\nabla f(x^{(0)}) = Ax - b = \begin{bmatrix} 0 \\ -3 \end{bmatrix}$$

$$p^{(0)} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$$\alpha_0 = \frac{-\begin{bmatrix} 0 & 3 \end{bmatrix}}{\begin{bmatrix} 0 \\ -3 \end{bmatrix}\begin{bmatrix} 0 & 3 \end{bmatrix}} = \frac{1}{2}$$

$$x^{(1)} = x^{(0)} - \alpha_0 \nabla f(x^{(0)}) = \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix}$$

$$s^{(0)} = \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix}$$

$$\nabla f(x^{(1)}) = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix} - \begin{bmatrix} 0 \\ 3 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}$$

$$y^{(0)} = \nabla f(x^1) - \nabla f(x^{(0)}) = \begin{bmatrix} -\frac{3}{2} \\ 3 \end{bmatrix}$$

$$\alpha_1 = \frac{||s^{(0)}||^2}{< s^{(0)}, y^{(0)} >} = \frac{\frac{9}{4}}{\frac{9}{2}} = \frac{1}{2}$$

$$x^{(2)} = x^{(1)} - \alpha_1 \nabla f(x^{(1)}) = \begin{bmatrix} 0 \\ \frac{3}{2} \end{bmatrix} - \frac{1}{2}\begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{3}{2} \end{bmatrix}$$

$$S^{(1)} = X^{(2)} - X^{(1)} = \begin{bmatrix} \frac{3}{4} \\ 0 \end{bmatrix}$$

$$\nabla f(x^{(2)}) = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} \frac{3}{4} \\ \frac{3}{2} \end{bmatrix} - \begin{bmatrix} 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{3}{4} \end{bmatrix}$$

$$y^{(1)} = \nabla f(x^{(2)}) - \nabla f(x^{(1)})$$

$$= \begin{bmatrix} 0 \\ -\frac{3}{4} \end{bmatrix} - \begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ -\frac{3}{4} \end{bmatrix}$$

$$\alpha_2 = \frac{||s^{(1)}||^2}{< s^{(1)}, y^{(1)} >} = \frac{\frac{9}{4}}{\frac{9}{2}} = \frac{1}{2}$$

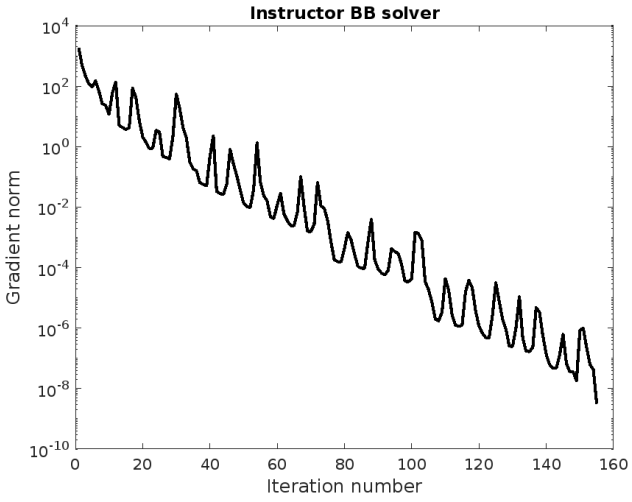
$$x^{(3)} = x^{(2)} - \alpha_2 \nabla f(x^{(2)}) = \begin{bmatrix} \frac{3}{4} \\ \frac{3}{2} \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 0 \\ -\frac{3}{4} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} \\ \frac{15}{8} \end{bmatrix}$$

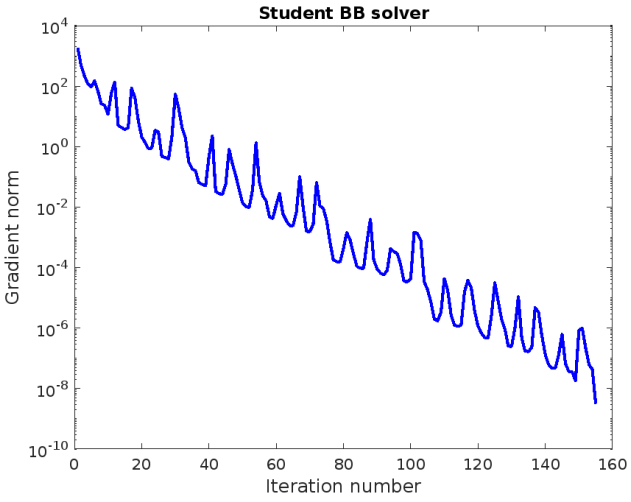
2. Write a solver for (QuadMin) by the BB method. Use the instructor's provided file `quadMin_BB.m` to write a Matlab function `quadMin_BB` with input $\mathbf{A}$, $\mathbf{b}$, initial vector $\mathbf{x}0$, and tolerance `tol`, and with stopping condition $\|\nabla f(\mathbf{x}^k)\| \leq$ tol. Also test your function by running the provided test file `test_BB.m` and compare to the instructor's function. Print your code and the results you get.

```
function [x, hist_res] = quadMin_BB(A,b,x0,tol)

% BB method for solving
% min_x 0.5*x'*A*x - b'*x

x = x0;

%% perform one steepest gradient descent with exact line search

% compute gradient of the objective
grad = A*x - b;

% evaluate the norm of gradient
res = norm(grad);

% save the value of res
hist_res = res;

alpha = grad'*grad/(-grad'*A'*-grad);

% update x by steepest gradient descent
x = x-alpha*grad;

%% main iteration
while res > tol

    % compute s and y
    s = x-x0;

    % save the old grad
    grad0 = grad;
```

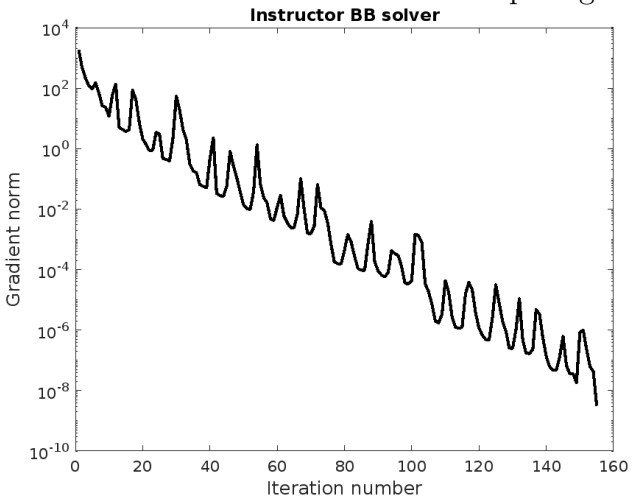
```
    % compute a new grad
    grad = A*x-b;

    y = grad-grad0;

    % compute alpha by option I or option II

    alpha = s'*s/(s'*y);

    % update x
    x0 = x;
    x = x-alpha*grad;

    % evaluate the norm of gradient
    res = norm(grad);

    % save the value of res
    hist_res = [hist_res; res];
end

end
```

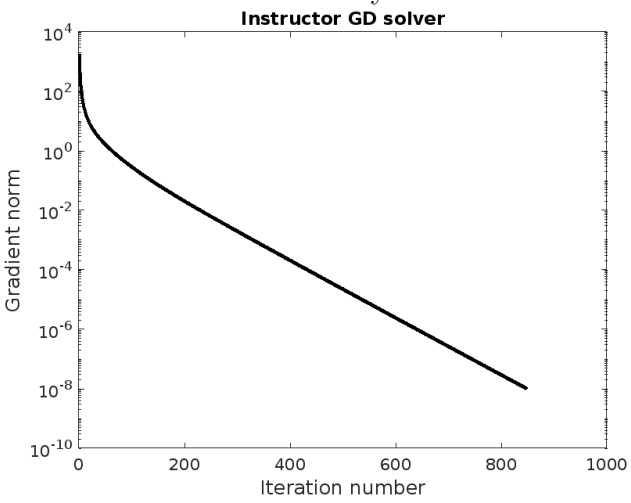
Student BB solver: Total running time is 0.0136
Final objective value is -20.6571
Instructor BB solver: Total running time is 0.0136
Final objective value is -20.6571
Instructor gradient descent solver: Total running time is 0.1290
Final objective value is -20.6571





3. Compare the steepest gradient descent method to the BB method by the provided test file `test_BB.m`. You can use the solver you developed in the 2nd homework or the instructor's solver of the steepest gradient descent. State what you observe.





From my observation, the gradient is going down constantly for the GD method and is bouncing for the BB method. Because the step size is determined by the inverse of the magnitude of the difference in gradients from the current and previous iterations. When this difference is small, the step size becomes large, which can lead to overshooting the minimum point and bouncing back and forth between points.

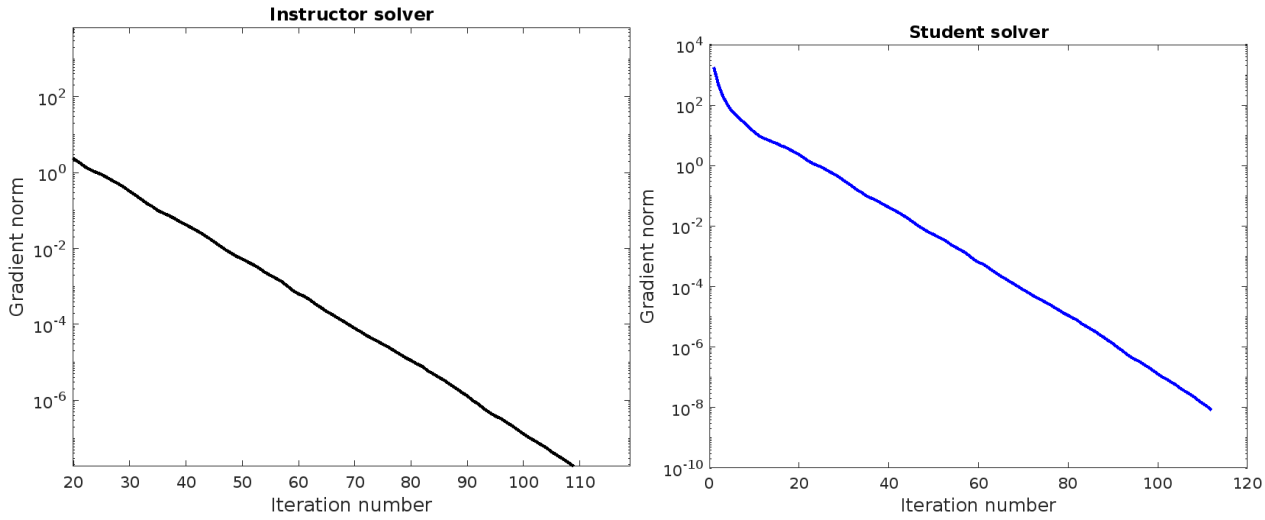
Problem 2

Consider the problem in (QuadMin) again. Write a solver for (QuadMin) by the DFP method. Use the instructor's provided file `quadMin_DFP.m` to write a Matlab function `quadMin_DFP` with input $\mathbf{A}$, $\mathbf{b}$, initial vector $\mathbf{x}0$, and tolerance `tol`, and with stopping condition $\|\nabla f(\mathbf{x}^k)\| \leq \text{tol}$. Also test your function by running the provided test file `test_DFP.m` and compare to the instructor's function. Print your code and the results you get, and print the figures you get.

```
function [x, hist_res] = quadMin_DFP(A,b,x0,tol)

% DFP method for solving
% min_x 0.5*x'*A*x - b'*x

x = x0;

%% perform one steepest gradient descent with exact line search

% compute gradient of the objective
grad = A*x-b;

% evaluate the norm of gradient
res = norm(grad);

% save the value of res
hist_res = res;

% choose the initial H matrix

H = eye(length(b));

%% main iteration
while res > tol

    % compute the search direction
    p = -H*grad;

    % use exact line search
```

```
    alpha = norm(p)^2 / ( p'*(A*p) );

    % update x
    x0 = x;
    x = x+alpha*p;

    grad0 = grad;
    % compute gradient of the objective
    grad = A*x-b;

    % compute s and y
    s = x-x0;

    y = grad-grad0;

    temp = H*y;
    % update H matrix
    H = H - (temp*temp')/(y'*temp)+(s*s')/(s'*y);

    % evaluate the norm of gradient
    res = norm(grad);

    % save the value of res
    hist_res = [hist_res; res];
end

end
```

Student solver: Total running time is 0.1518
Final objective value is -23.1038
Instructor solver: Total running time is 0.1544
Final objective value is -23.1038


Instructor solver
Gradient norm
Iteration number
Student solver
Gradient norm
Iteration number

Problem 3

Recall that in DFP and BFGS methods, we have a condition $(\mathbf{s}^{(k)})^\top \mathbf{y}^{(k)} > 0$, which is guaranteed if the Wolfe's conditions hold. Here, $\mathbf{s}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$. Now suppose the function f is strongly convex, namely, there is $\mu > 0$ such that $(\mathbf{x}_1 - \mathbf{x}_2)^\top (\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)) \geq \mu \|\mathbf{x}_1 - \mathbf{x}_2\|^2$. Prove that the condition $(\mathbf{s}^{(k)})^\top \mathbf{y}^{(k)} > 0$ always holds when $\mathbf{s}^{(k)} \neq \mathbf{0}$, even if the Wolfe's conditions may NOT hold.

Since we assume f is strongly convex,

$$(x_1 - x_2)^T(\nabla f(x_1) - \nabla f(x_2)) \geq \mu ||x_1 - x_2||^2$$

Let $x_1 = x^{(k+1)}, x_2 = x^k,$ then

$$(s^{(k)})^T(y^{(k)}) \geq \mu ||s^{(k)}||^2$$

Since $s^{(k)} \neq 0$

We have $\mu ||s^{(k)}|| \geq 0$

Therefore$(s^{(k)})^T y^{(k)} > 0$

Problem 4

Let $f(x_1, x_2) = 50(x_2 - x_1^2)^2 + (1 - x_1)^2$. Write the gradient $\nabla f(x_1, x_2)$ and the Hessian matrix $\nabla^2 f(x_1, x_2)$ at any point (x_1, x_2). Program the Newton's algorithm to minimize $f(x_1, x_2)$. Run the algorithm to 10 iterations and report the norm of gradient at each iteration. For the initial point, first use $(1.5, 1.5)$ and then $(-1.5, 1.5)$.

$$
\begin{aligned}
\frac{\partial f}{\partial x_1} &= 100(x_2 - x_1^2) \cdot -2x_1 + 2(1 - x_1) \cdot -1 \\
&= -200x_1x_2 + 200x_1^3 - 2 + 2x_1 \\
\frac{\partial f}{\partial x_2} &= 100(x_2 - x_1^2) \cdot 1 \\
&= 100x_2 - 100x_1^2 \\
\nabla f &= \begin{bmatrix} -200x_1x_2 + 200x_1^3 - 2 + 2x_1 \\ 100x_2 - 100x_1^2 \end{bmatrix} \\
\frac{\partial^2 f}{\partial x_1^2} &= -200x_2 + 600x_1^2 + 2 \\
\frac{\partial^2 f}{\partial x_1 x_2} &= \frac{\partial^2 f}{\partial x_2 x_1} = -200x_1 \\
\frac{\partial^2 f}{\partial x_2^2} &= 100 \\
\nabla^2 f &= \begin{bmatrix} -200x_2 + 600x_1^2 + 2 & -200x_1 \\ -200x_1 & 100 \end{bmatrix}
\end{aligned}
$$

```
x = [-1.5 1.5];
iter = 0;
%% main iteration
while iter < 10
    grad = [-200*x(1)*x(2)+200*(x(1)^3)-2+2*x(1) 100*x(2)-100*(x(1)^2)];
    hessian = [-200*x(2)+600*(x(1)^2)+2 -200*x(1);-200*x(1) 100];
    x = x-inv(hessian)*grad';
    fprintf('Iteration %d: Norm of grad is %f\n',iter+1, norm(grad));
    iter = iter+1;
end
fprintf('The final minimizer is x1 = %f and x2 = %f',x(1),x(2));
```

For point $(x_1, x_2) = (1.5, 1.5)$, we have:
Iteration 1: Norm of grad is 238.119718

Iteration 2: Norm of grad is 0.999779
Iteration 3: Norm of grad is 54.067700
Iteration 4: Norm of grad is 0.004085
Iteration 5: Norm of grad is 0.000932
Iteration 6: Norm of grad is 0.000000
Iteration 7: Norm of grad is 0.000000
Iteration 8: Norm of grad is 0.000000
Iteration 9: Norm of grad is 0.000000
Iteration 10: Norm of grad is 0.000000
The final minimizer is x1 = 1.000000 and x2 = 1.000000
For point $(x_1, x_2) = (-1.5, 1.5)$, we have:
Iteration 1: Norm of grad is 241.919408
Iteration 2: Norm of grad is 934.987443
Iteration 3: Norm of grad is 4.937416
Iteration 4: Norm of grad is 1316.511625
Iteration 5: Norm of grad is 0.032739
Iteration 6: Norm of grad is 0.059919
Iteration 7: Norm of grad is 0.000000
Iteration 8: Norm of grad is 0.000000
Iteration 9: Norm of grad is 0.000000
Iteration 10: Norm of grad is 0.000000
The final minimizer is x1 = 1.000000 and x2 = 1.000000