Assignment 3 of MATP4820

Xinshi Wang

Problem 1 (only for MATP 4800 students)

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 $\mathbf{b} = \begin{bmatrix} 3 \\ -3 \end{bmatrix}$. Set the initial vector $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Do by hand the conjugate gradient method to find the **exact solution**.

$$
\begin{aligned}
r^{(0)} &= Ax^{(0)} - b = \begin{bmatrix} -3 \\ 3 \end{bmatrix} \\
p^{(0)} &= -r^{(0)} = \begin{bmatrix} 3 \\ -3 \end{bmatrix} \\
||r^{(0)}|| &= \sqrt{18} > 0 \\
\alpha_0 &= \frac{\|r^{(0)}\|^2}{< p^{(0)}, Ap^{(0)} >} = \frac{18}{< \begin{bmatrix} 3 \\ -3 \end{bmatrix}, \begin{bmatrix} 9 \\ -9 \end{bmatrix} >} = \frac{18}{54} = \frac{1}{3} \\
x^{(1)} &= x^{(0)} + \alpha_0 p^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\
r^{(1)} &= r^{(0)} + \alpha_0 A p^{(0)} = \begin{bmatrix} 3 \\ -3 \end{bmatrix} \\
\beta_1 &= \frac{\|r^{(1)}\|^2}{\|r^{(0)}\|} = 1 \\
p^{(1)} &= -r^{(1)} + \beta_1 p^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\end{aligned}
$$

since the gradient is 0, we found a exact solution $x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

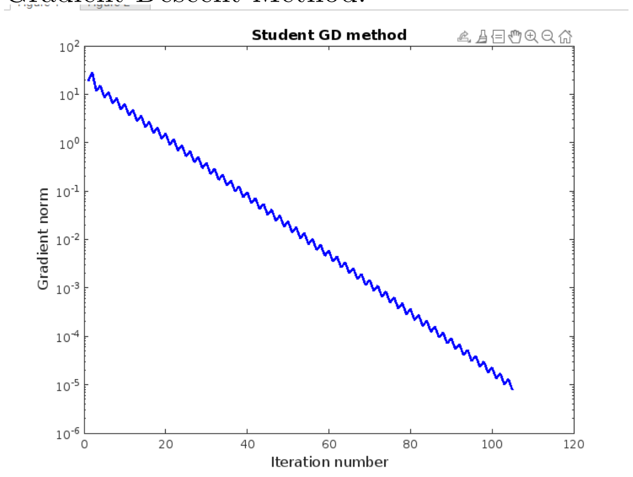
2. Use the instructor's provided file `quadMin_cg.m` to write a Matlab function `quadMin_cg` with input $\mathbf{A}$, $\mathbf{b}$, initial vector $\mathbf{x}0$, and tolerance `tol`, and with stopping condition $\|\nabla f(\mathbf{x}^k)\| \leq \text{tol}$. Also test your function by running the provided test file `test_cg.m` and compare to the instructor's function. Print your code and the results you get. Make discussions on what you observe for the difference between your steepest gradient method solver (from the 2nd assignment) and your conjugate gradient method solver.

```
function [x, hist_res] = quadMin_gd(A,b,x0,tol)
% conjugate gradient method for solving
% min_x 0.5*x'*A*x - b'*x
% get the size of the problem
n = length(b);
```

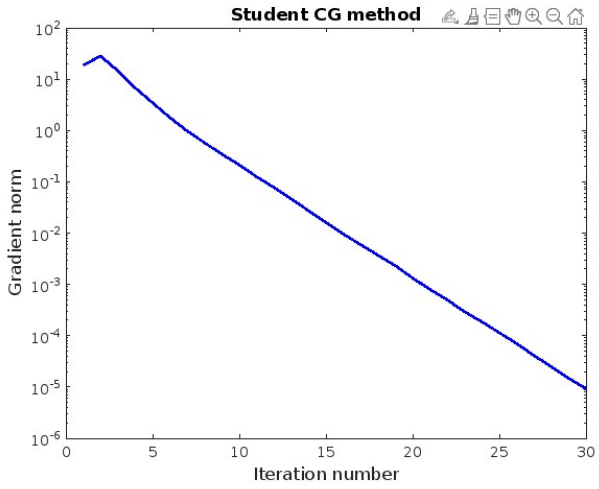
```
x = x0;
% compute vector r, i.e., gradient of the objective
r = A*x-b;
% set the first p vector as negative gradient
p = -r;
% evaluate the norm of gradient
res = norm(r);
% save the value of res
hist_res = res;
while res > tol
    % compute alpha
    alpha = res^2/(p'*(A*p));
    % update x
    x = x + alpha*p;
    % update r
    r = A*x-b;
    % compute beta
    beta = norm(r)^2/(res)^2;
    % obtain the new p vector
    p = -r+beta*p;
    % evaluate the norm of residual vector r
    res = norm(r);
    % save the value of res
    hist_res = [hist_res; res];
end
end
```

Results by student code
Total running time is 0.0069
Final objective value is -285.8224
Results by Instructor code
Total running time is 0.0024
Final objective value is -285.8224
While both of the methods would converge to the correct answer, the steepest gradient method solver (from the 2nd assignment) seems to converge faster compared with conjugate gradient method solver in general.

Gradient Descent Method:



Conjugate Gradient Method:



It is easier to find that CG method requires only 30 iterations while the steepest GD needs more than 100. Also, the CG method is more stable than the steepest GD method.

Problem 2

For ill-conditioned problems, the conjugate gradient (CG) method may converge very slowly, and a better method is the preconditioned conjugate gradient (PCG) method. Consider the quadratic minimization problem in (QuadMin) stated in Problem 1. Write a solver for (QuadMin) by the PCG method. We will use a non-singular triangular matrix $\mathbf{C}$ as the preconditioner.

1. Let $\mathbf{C} \in \mathbb{R}^{n\times n}$ be a non-singular upper triangular matrix. Write a solver `pcg_linsolv` for the linear system $\mathbf{C}^\top \mathbf{C}\mathbf{y} = \mathbf{r}$. Treat $\mathbf{C}$ and $\mathbf{r}$ as the input of the solver. Recall that to solve the linear system, we can first solve $\mathbf{C}^\top \mathbf{z} = \mathbf{r}$ by forward substitution and then solve $\mathbf{C}\mathbf{y} = \mathbf{z}$ by back substitution. [**Note**: you will need to use for-loop in your solver. You CANNOT use MATLAB's solver to directly solve the linear system.]

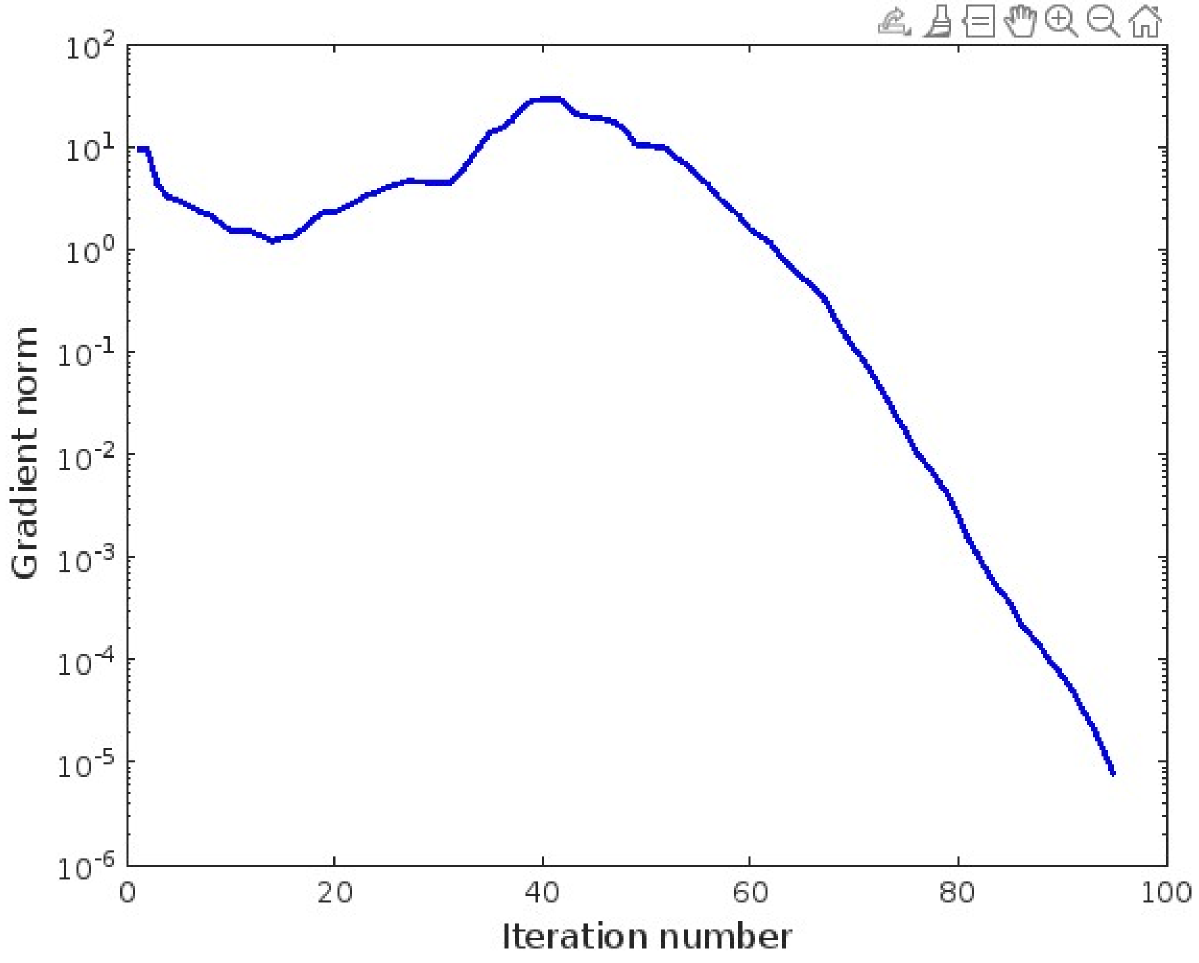
```
0-function [y] = pcg_linsolv(C,r)
L = C';
U = C;
n = size(L, 1);
z = zeros(n, 1);
for i = 1:n
    z(i) = r(i);
    for j = 1:i-1
        z(i) = z(i) - L(i,j)*z(j);
    end
    z(i) = z(i)/L(i,i);
end
y = zeros(n, 1);
for i = n:-1:1
    y(i) = z(i);
    for j = i+1:n
        y(i) = y(i) - U(i,j)*y(j);
    end
    y(i) = y(i)/U(i,i);
end
end
```

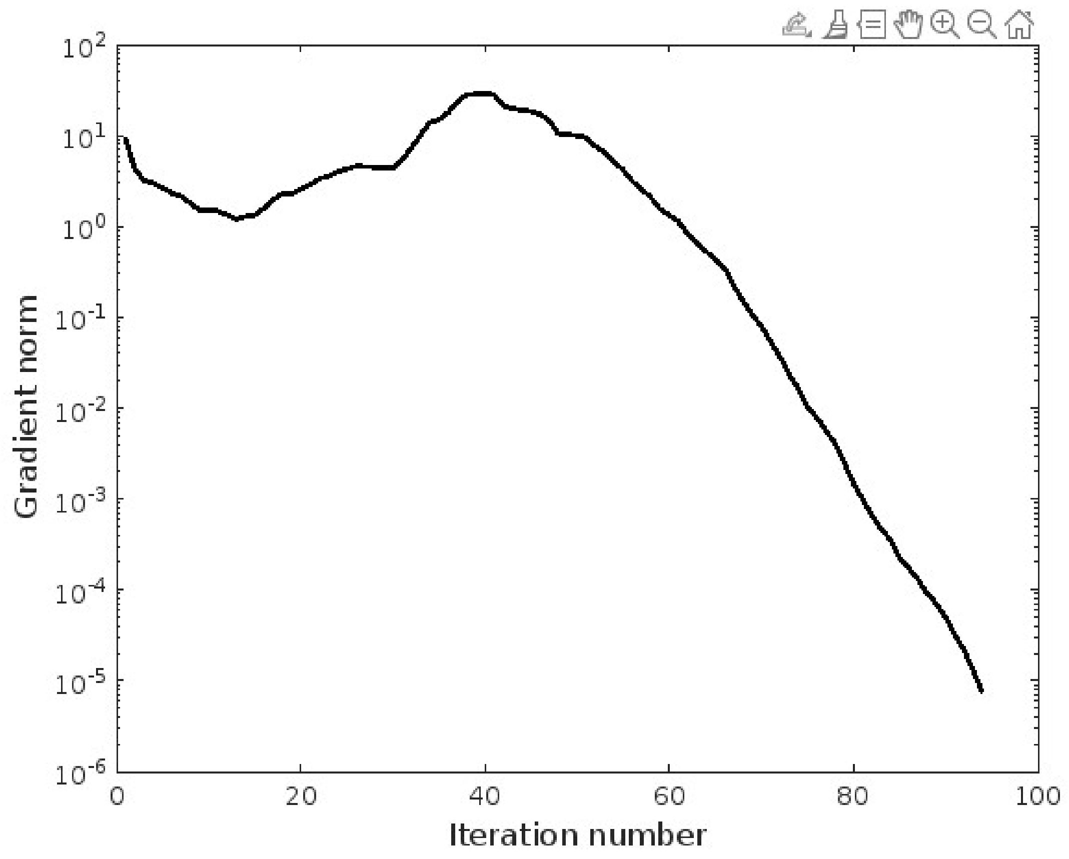
2. Write a Matlab function `quadMin_pcg` with input $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{b}$, where $\mathbf{C}$ is a non-singular upper triangular matrix, used as the conditioning matrix. Within your PCG

solver, use `pcg_linsolv` that you develop in the previous question as the linear solver for solving linear systems in the form of $\mathbf{C}^\top\mathbf{C}\mathbf{y} = \mathbf{r}$. Also test your function by running the provided test file `test_pcg.m` and compare to the instructor's function. Print your code and the results you get. [**Hint**: to debug your PCG solver, you can use MATLAB's linear system solver to directly solve $\mathbf{C}^\top\mathbf{C}\mathbf{y} = \mathbf{r}$. But your final version of your PCG solver MUST use `pcg_linsolv` that you develop.]

```
function [x, hist_res] = quadMin_pcg(A,C,b,x0,tol)
% conjugate gradient method for solving
% min_x 0.5*x'*A*x - b'*x
% get the size of the problem
n = length(b);
x = x0;
% compute vector r, i.e., gradient of the objective
r = A*x-b;
% set the first p vector
p = pcg_linsolv(C,-r);
% evaluate the norm of gradient
res = norm(r);
% save the value of res
hist_res = res;
while res > tol
    % compute alpha
    y = pcg_linsolv(C,r);
    alpha = r'*y/(p'*(A*p));
    % update x
    x = x+alpha*p;
    % update r
    r_new = r+alpha*A*p;
    y_new = pcg_linsolv(C,r_new);
    % compute beta
    beta = r_new'*y_new/(r'*y);
    % obtain the new p vector
    p = -y_new+beta*p;
    % evaluate the norm of residual vector r
    res = norm(r);
    y = y_new;
    r = r_new;
    % save the value of res
    hist_res = [hist_res; res];
end
end
```

Student solver: Total iteration is 96
Total running time is 0.8075
Final objective value is -17.8312
Instructor pcg solver: Total iteration is 95
Total running time is 0.4831
Final objective value is -17.8312
Instructor cg solver: Total iteration is 205
Final objective value is -17.8312





Gradient norm
Iteration number
10^2
10^1
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6
0
20
40
60
80
100

Problem 3

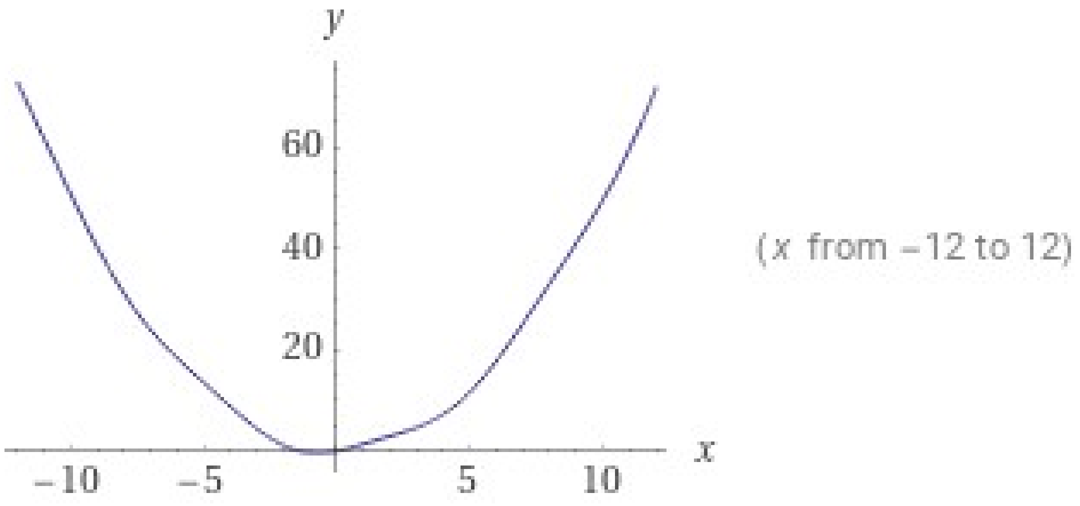
Let $f(x) = \frac{1}{2}x^2 + \sin x, x \in \mathbb{R}$.

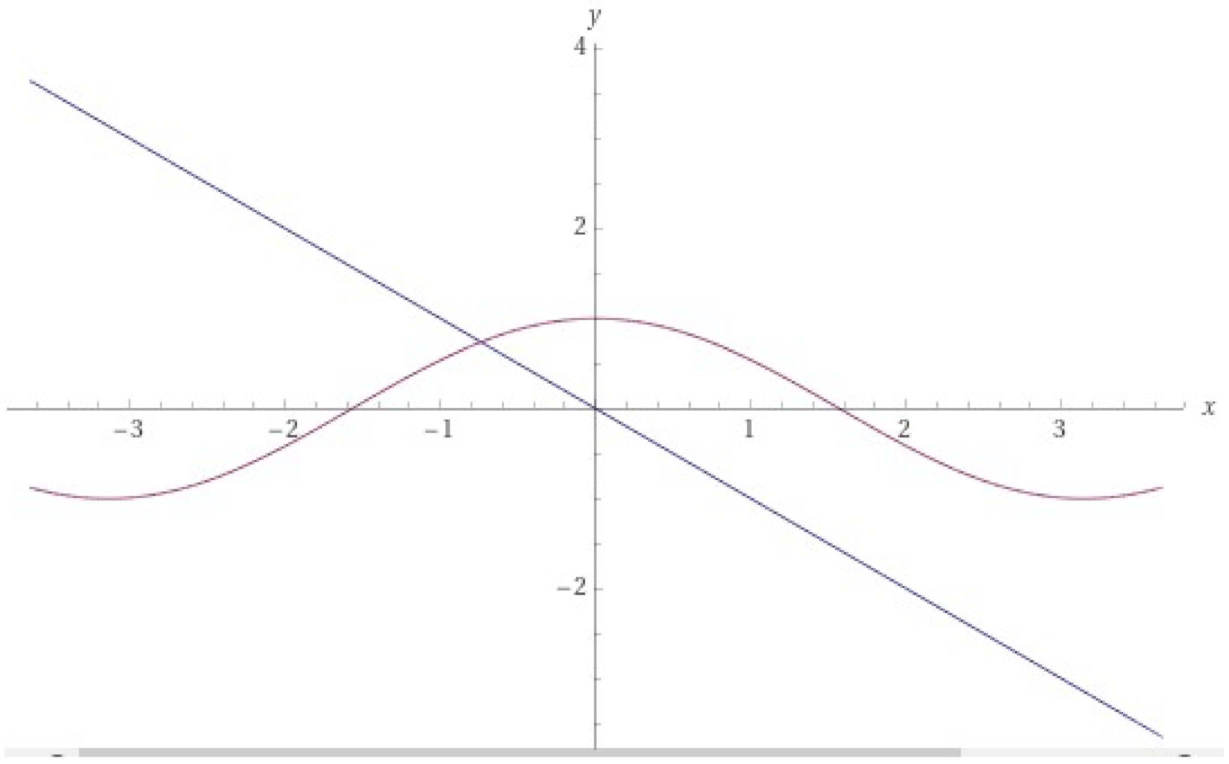
1. Prove that this function is convex and show by graph that this function has a minimizer.

$$f'(x) = \frac{1}{2} \cdot 2x + cos(x) = x + \cos(x)$$
$$f''(x) = 1 - \sin(x) \geq 0$$

Since the hessian matrix (second derivative) is positive semi-definite, the function is convex.

Since the intersection of the gradient is near the optimal value for the original function, we have a minimum.





2. Start from $x^{(0)} = -1.5$ and do 3 iterations of the Newton's method to find an approximate minimizer of $f(x)$ by hand (calculator is allowed).

Iteration 1:

$$p^{(0)} = -(\nabla^2 f(x^{(0)}))^{-1} \nabla f(x^{(0)}) \approx 0.7155$$

$$x^{(1)} = x^{(0)} + p^{(0)} = -1.5 + 0.7155 \approx -0.7845$$

Iteration 2:

$$p^{(1)} = -(\nabla^2 f(x^{(1)}))^{-1} \nabla f(x^{(1)}) \approx 0.0450$$

$$x^{(2)} = x^{(1)} + p^{(1)} = -0.7845 + 0.0450 \approx -0.7395$$

Iteration 3:

$$p^{(2)} = -(\nabla^2 f(x^{(2)}))^{-1} \nabla f(x^{(2)}) = 4.335e - 4$$

$$x^{(2)} = x^{(2)} + p^{(2)} = -0.7395 + 4.335e - 4 \approx -0.7391$$

Problem 4 (only for MATP 6100 students)

For the conjugate gradient method, prove that $\langle \mathbf{r}^{(k)}, \mathbf{r}^{(i)} \rangle = 0$ for any $i = 1, 2, \ldots, k-1$ and for $k \geq 2$. [Hint: prove this by induction: first verify $\langle \mathbf{r}^{(2)}, \mathbf{r}^{(1)} \rangle = 0$, i.e., the result holds for $k = 2$; then complete the induction step.]