Exercise 1: (a) Solve
\[
\min_{x \in \mathbb{R}^2} \frac{1}{2} x^T \begin{bmatrix} 1 & 0 \\ 0 & 900 \end{bmatrix} x
\]
using the steep descent with exact line search. You are welcome to copy the MATLAB code from my slides. Start your code with the initial point \(x_0 = [1000 \ 1]^T\). Stop until \(\|x_{n+1} - x_n\|_2 < 10^{-8}\). Report your solution and the number of iteration.

**Ans:** We consider solving an unconstrained quadratic programming problem. That is,
\[
\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T Q x + p^T x.
\]
Let \(g_n\) be the gradient of \(f(x)\) at \(x_n\) and
\[
h(\lambda) = f(x_n + \lambda(-g_n)) = \frac{1}{2} (x_n - \lambda g_n)^T Q (x_n - \lambda g_n) + p^T (x_n - \lambda g_n).
\]
Find \(\lambda^*\) such that \(\frac{dh(\lambda)}{d\lambda} = 0\). We have \(\lambda^* = \frac{g_n^T g_n}{g_n^T Q g_n}\).

function [x, f_value, iter] = grdlines(Q, p, x0, esp)
% min 0.5*x’Q*x+p’x
% Solving unconstrained minimization via
% steep descent with exact line search
% The stopping criterion:
% Either the ||gradient||_2^2 ,10^-12

% or \|x_{n+1} - x_n\|_2 < \epsilon
flag = 1; iter = 0; while flag > esp
    grad = Q*x0+p;
    temp1 = grad'*grad;
    if temp1 < 10^{-12}
        flag = esp
    else
        stepsize = temp1/(grad'*Q*grad);
        x1 = x0 - stepsize*grad;
        flag = norm(x1-x0);
        x0=x1;
    end;
    iter = iter+1;
end;

x = x0;

f_value = 0.5*x'*Q*x+p'*x;

(b) Implement the Newton’s method for minimizing a quadratic function
\( f(x) = \frac{1}{2} x^T Q x + p^T x \) in MATLAB code. Apply your code to solve the minimization problem in (a).

Ans: function [x, f_value, iter] = newtonqp(Q,p, x0, esp)

% min 0.5*x'*Q*x+p'*x
% Solving unconstrained QP via
% Newton’s method
%
% The stopping criterion:
% Either the \|\text{gradient}\|_2^2 ,10^{-12}
% % or \|x_{n+1} - x_n\|_2 < \epsilon
flag = 1; iter = 0; while flag > esp
    grad = Q*x0+p;
    temp1 = grad'*grad;
    if temp1 < 10^{-12}
        flag = esp
    else
        %d=inv(Q)*grad;
\[ d = x_0 + \text{inv}(Q) \cdot p; \]
\[ x_1 = x_0 - d; \]
\[ \text{flag} = \text{norm}(x_1 - x_0); \]
\[ x_0 = x_1; \]
\end{verbatim}

end;

\[ \text{iter} = \text{iter} + 1; \]
\end{verbatim}

end;

\[ x = x_0; \]

\[ f\text{-value} = 0.5 \cdot x' \cdot Q \cdot x + p' \cdot x; \]

Exercise 2: Find an approximate solution using MATLAB to the following system by minimizing \( \| Ax - b \|_p \) for \( p = 1, 2, \infty \). Write down both the approximate solution, and the value of the \( \| Ax - b \|_p \). Draw the solution points in \( R^2 \) and the four equations being solved.

\[
\begin{align*}
x_1 + 2x_2 &= 2 \\
2x_1 - x_2 &= -2 \\
x_1 + x_2 &= 3 \\
4x_1 - x_2 &= -4
\end{align*}
\]

Ans: (a) \( \| Ax - b \|_1 \):

\[
\text{function } [x, \text{ residual, one\_error}]=\text{oneapprox}(A,b) \\
% \text{Input A: mXn matrix} \\
% \text{b: m-vector} \\
% \text{Solve the problem by LP} \\
% \text{Output: the approximate solution of Ax=b} \\
% \text{one\_error = ||Ax-b||}_1 \\
% [m,n]=\text{size(A)}; \text{ obj\_p=[zeros(n,1); ones(m,1)];} \\
H=[A -\text{eye(m)}; -A -\text{eye(m)}]; h=[b;-b]; \\
[sol, one\_error]=\text{linprog(obj\_p,H,h)}; \\
x=\text{sol}([1:n]); \text{ residual=sol}([(n+1):(m+n)]);
\]

We have \( x^* = [-0.6667, 1.333]' \) and \( \| Ax^* - b \|_1 = 3 \).
\[(b) \|Ax - b\|_2: \text{This problem is equivalent to}\]

\[
\min_{x \in \mathbb{R}^2} \frac{1}{2} \|Ax - b\|_2^2 \iff \min_{x \in \mathbb{R}^2} \frac{1}{2} x' A' Ax - b' Ax.
\]

Hence, can use the code given in Exercise 1 (b). Please note that the objective function value returned by the code is not \(\|Ax - b\|_2\). We have \(x^* = [-0.4552, 1.6621]' \) and \(\|Ax^* - b\|_2 = 2.1367\). Of course, you can solve the normal equation, \(x^* = (A'A)^{-1}A'b\) directly.

\[(c) \|Ax - b\|_\infty:\]

```matlab
function [x, inf_error, residual ]=infapprox(A,b)

%Input A: mXn matrix
% b: m-vector
%
%Solve the problem by LP
%Output: the approximate solution of Ax=b
% inf_error =||Ax-b||_\infty

[m,n]=size(A); obj_p=[zeros(n,1); 1];
H=[A -ones(m,1); -A -ones(m,1)]; h=[b;-b];
[sol, one_error]=linprog(obj_p,H,h); x=sol(1:n);
inf_error=sol((n+1)); residual=A*x-b;
```

We have \(x^* = [-0.2, 1.8]'\) and \(\|Ax^* - b\|_\infty = 1.4\).