Caleb Logemann
MATH 565 Continuous Optimization
Homework 3

1. Page 100: Problem 4.9.
   Derive the solution of the two-dimensional subspace minimization problem in the case where B is positive definite.

   Proof. The two-dimensional subspace minimization problem can be stated as

$$\min_p\{m(p)\} = \min_p\left\{f + g^T p + \frac{1}{2}p^T Bp\right\} \qquad s.t. \|p\| \leq \Delta, p \in \operatorname{span}\left(g, B^{-1}g\right)$$

If $\left\|B^{-1}g\right\| \leq \Delta$, then clearly $p = B^{-1}g$ is the optimal solution to the minimization subproblem. Therefore let $\left\|B^{-1}g\right\| > \Delta$. In this case, since the model function is convex as B is positive definite, the minimal solution will be on the boundary. In other words $\|p\| = \Delta$. Also since $p \in \operatorname{span}(g, B^{-1}g)$, we can rewrite p as $p = c_1 g + c_2 B^{-1}g$. In this case the model function can be expressed as a function of c_1 and c_2,

$$m(c_1, c_2) = f + g^T\left(c_1 g + c_2 B^{-1}g\right) + \frac{1}{2}\left(c_1 g + c_2 B^{-1}g\right)^T B\left(c_1 g + c_2 B^{-1}g\right)$$

or

$$m(c_1, c_2) = f + \|g\|^2 c_1 + \|g\|^2_{B^{-1}} c_2 + \frac{1}{2}\|g\|^2_B c_1^2 + \|g\|^2 c_1 c_2 + \frac{1}{2}\|g\|^2_{B^{-1}} c_2^2$$

Also we can rewrite the condition $\|p\|^2 = \Delta^2$ as an expression $g(c_1, c_2) = 0$.

$$\begin{aligned}
g(c_1, c_2) &= \left\|c_1 g + c_2 B^{-1}g\right\|^2 - \Delta^2 \\
&= \left(c_1 g + c_2 B^{-1}g\right)^T\left(c_1 g + c_2 B^{-1}g\right) - \Delta^2 \\
&= \|g\|^2 c_1^2 + 2\|g\|^2_{B^{-1}} c_1 c_2 + \left\|B^{-1}g\right\|^2 c_2^2 - \Delta^2
\end{aligned}$$

In order to minimize $m(c_1, c_2)$ subject to the condition $g(c_1, c_2) = 0$, we can first solve for c_2 in terms of c_1 using $g(c_1, c_2) = 0$ as follows.

$$0 = \|g\|^2 c_1^2 - \Delta^2 + 2\|g\|^2_{B^{-1}} c_1 c_2 + \left\|B^{-1}g\right\|^2 c_2^2$$

Since this is a quadratic equation in c_2

$$\begin{aligned}
c_2 &= \frac{-2\|g\|^2_{B^{-1}} c_1 \pm \sqrt{4\|g\|^4_{B^{-1}} c_1^2 - 4\left(\|g\|^2 c_1^2 - \Delta^2\right)\left(\|B^{-1}g\|^2\right)}}{2\|B^{-1}g\|^2} \\
c_2(c_1) &= \frac{-\|g\|^2_{B^{-1}} c_1 \pm \sqrt{\|g\|^4_{B^{-1}} c_1^2 - \left(\|g\|^2 c_1^2 - \Delta^2\right)\left(\|B^{-1}g\|^2\right)}}{\|B^{-1}g\|^2}
\end{aligned}$$

Now m can be written as a one-dimensional equation in c_1.

$$m(c_1) = f + \|g\|^2 c_1 + \|g\|^2_{B^{-1}} c_2(c_1) + \frac{1}{2}\|g\|^2_B c_1^2 + \|g\|^2 c_1 c_2(c_1) + \frac{1}{2}\|g\|^2_{B^{-1}} c_2(c_1)^2$$

In order to minimize this equation we must solve the equation $\frac{\partial m}{\partial c_1} = 0$.

$$\frac{\partial m}{\partial c_1} = \|g\|^2 + \|g\|^2_{B^{-1}} c_2'(c_1) + \|g\|^2_B c_1 + \|g\|^2\left(c_2(c_1) + c_1 c_2'(c_1)\right) + \|g\|^2_{B^{-1}} c_2(c_1) c_2'(c_1)$$

Finding the roots of this equation is extremely intensive, so I will just state that c_1 needs to be a root of this equation. Then we have already computed c_2 in terms of c_1, and the minimizer is

$$p = c_1 g + c_2 B^{-1} g$$

□

2. Page 100: Problem 4.10.
Show that if B is any symmetric matrix, then there exists $\lambda \geq 0$ such that $B+\lambda I$ is positive definite.

Proof. Let B be a symmetric matrix. Since B is symmetric all of the eigenvalues of B are real. If all of the eigenvalues of B are positive, then B is already positive definite. In this case, $B = B + 0I$ is positive definite. Therefore consider when B has some negative eigenvalues. Let μ_1 be the most negative eigenvalue, that is $\mu_1 \leq \mu_i$ for any eigenvalue, μ_i, of B. I will let $\lambda = -\mu_1 + 1$. I now claim that $B + \lambda I$ is positive definite. To see this note that the eigenvectors of B can form an orthonormal basis of $\mathbb{R}^N$, when $B \in \mathbb{R}^{N\times N}$. Let $\{v_i\}$ denote this basis and consider that any $x \in \mathbb{R}^N$ can be expressed as $x = \sum_{i=1}^{N} (a_i v_i)$. Now consider $x^T(B + \lambda I)x$.

$$\begin{aligned}
x^T(B+\lambda I)x &= \sum_{i=1}^{N} \left(a_i v_i^T\right)(B+\lambda I)\sum_{j=1}^{N}(a_j v_j) \\
&= \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\left(a_i a_j v_i^T (B+\lambda I) v_j\right)\right) \\
&= \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\left(a_i a_j \left(v_i^T B v_j + \lambda v_i^T v_j\right)\right)\right) \\
&= \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\left(a_i a_j \left(\mu_j v_i^T v_j + \lambda v_i^T v_j\right)\right)\right)
\end{aligned}$$

Since $v_i^T v_j = 0$ for all $i \neq j$

$$\begin{aligned}
&= \sum_{i=1}^{N}\left(a_i^2\left(\mu_j v_i^T v_i + \lambda v_i^T v_i\right)\right) \\
&= \sum_{i=1}^{N}\left(a_i^2\left(\mu_i \|v_i\|^2 + \lambda \|v_i\|^2\right)\right) \\
&= \sum_{i=1}^{N}\left(a_i^2(\mu_i + \lambda)\right)
\end{aligned}$$

Note that $\mu_i + \lambda > 0$ for any eigenvalue μ_i as $\lambda + \mu_1 = 1$ and $\mu_1 \leq \mu_i$ for all eigenvalues, therefore

$$x^T(B+\lambda I)x > 0$$

This shows that $B + \lambda I$ is positive definite. □

3. Implement the linear conjugate gradient method in MATLAB or PYTHON.

The following function implements the linear conjugate gradient method in PYTHON.

```python
import numpy as np
def linearConjugateGradient(A, b, x, MaxIter, TOL):
    r = np.dot(A, x) - b
    delta = np.dot(r, r)
    p = -r
    k = 0
    mstop = 1
    while k < MaxIter and mstop:
        k+=1
        w = np.dot(A, p)
        alpha = delta/np.dot(p, w)
        x = x + alpha*p
        r = r + alpha*w
        deltaOld = delta
        delta = np.dot(r, r)
        if np.sqrt(delta) < TOL:
            mstop = 0
        else:
            beta = delta/deltaOld
            p = -r + beta*p
    return (x, k)
```

4. Page 133: Problem 5.1. (Use your method from the previous problem).
Implement Algorithm 5.2 and use it to solve linear systems in which A is the Hilbert matrix, whose elements $A_{i,j} = 1/(i+j-1)$. Set the right-hand-side to $b = (1, 1, \ldots, 1)^T$ and the initial point to $x_0 = 0$. Try dimensions $n = 5, 8, 12, 20$ and report the number of iterations required to reduce the residual below 10^{-6}.

The following script uses the linear conjugate gradient method to solve the given linear system.

```python
import numpy as np
from scipy.linalg import hilbert
execfile('linearConjugateGradient.py')

TOL = 1e-6
MaxIter = 100
for n in [5, 8, 12, 20]:
    A = hilbert(n)
    b = np.ones(n)
    x = np.zeros(n)
    (x, k) = linearConjugateGradient(A, b, x, MaxIter, TOL)
    print(k)
```

The required number of iterations to achieve error less than 10^{-6} is shown in the table below for different size matrices.

n	Number of Iterations
5	6
8	19
12	40
20	75

5. Page 133: Problem 5.2.
Show that if the nonzero vectors $p_0, p_1, \ldots, p_l$ satisfy (5.5), where A is symmetric and positive definite, then these vectors are linearly independent. (This result implies that A has a most n conjugate directions.)

Proof. Let A be symmetric and positive definite and let $p_0, p_1, \dots, p_l$ be A-conjugate, that is

$$p_i^T A p_j = 0 \qquad \forall i \neq j.$$

Consider a set of constants $c_0, c_1, \dots, c_l$, such that

$$\sum_{i=0}^{l} (c_i p_i) = 0$$

The set of vectors $\{p_i\}$ are linearly independent if $c_i = 0$ for all i. Consider the following

$$0 = \sum_{i=0}^{l} \left(c_i p_i^T\right) A \sum_{j=0}^{l} (c_j p_j)$$

$$0 = \sum_{i=0}^{l} \left(\sum_{j=0}^{l} \left(c_i c_j p_i^T A p_j\right)\right)$$

Since these vectors are A-conjugate

$$0 = \sum_{i=0}^{l} \left(c_i^2 p_i^T A p_i\right).$$

Note that since A is positive definite $p_i^T A p_i > 0$ for all i, and since $c_i^2 \geq 0$ this implies that

$$0 = c_i$$

This shows that $p_0, p_1, \dots, p_l$ are linearly independent. Therefore any set of A-conjugate vectors must also be linearly independent. Since a set of linearly independent vectors can be at most of size n, this implies that a set of A-conjugate vectors can be at most of size n. □

6. Let $n = N^2$. Downlonad the MATLAB file CreateA.m from the course website. The correct syntax for calling this code is

$$A = CreateA(N);$$

This creates a matrix of size $N^2 \times N^2$.

Apply your conjugate gradient method to this problem for various N. Make a table that records the number of iterations required to achieve a reasonable tolerance for $N = 10, 20, 40, 80, 160, 320$. You should use the same tolerance in each case. How does the number of iterations scale with N? What does this tell you about the condition number of A as N varies?

```
TOL = 1e-6;
MaxIter = 700;
for n=[10, 20, 40, 80, 160, 320]
    x0 = zeros(n*n, 1);
    b = ones(n*n, 1);
    A = CreateA(n);
    [x, k] = linearConjugateGradient(A, b, x0, TOL, MaxIter);
    disp(k)
    ek = TOL;
    e0 = sqrt((x - x0)'*A*(x - x0));
    e = (ek/(2*e0))^(1.0/k);
    c = ((e + 1)/(1 - e))^2;
    disp(c)
    disp(condest(A))
end
```

For this problem, I used $b = [1, 1, \ldots, 1]^T$ and $x_0 = 0$. The error tolerance was 10^{-6}. The following table shows N, the number of iterations required to converge, how the number of iterations scales with N, a lower bound for the condition number κ of A, and the actual condition number κ. The lower bound for the condition number κ is found using the following error condition,

$$\|x_k - x^*\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_0 - x^*\|_A.$$

N	number of iterations	scale	κ lower bound	κ
10	15	1.5	4	70
20	35	1.75	14	258
40	73	1.825	53	989
80	149	1.8625	191	3865
160	302	1.8875	690	15276
320	607	1.8969	2480	60728

As can be seen the number of iterations is approaching $\approx 1.9N$. Also it can be seen that the condition number of A, grows as N grows.