APPM 5610 - Homework 5

Eappen Nelluvelil

February 23, 2022

1. Suppose that a $n \times n$ matrix $\mathbf{A}$ is symmetric and positive definite. Consider the following iteration:

$$\mathbf{A}_0 = \mathbf{A}$$

here, $\mathbf{G}_k\mathbf{G}_k^T$ is the Cholesky factorization of a symmetric positive definite matrix. Show that this iteration is well-defined, i.e., $\mathbf{G}_k\mathbf{G}_k^T$ is symmetric and positive definite. Show that if

$$\mathbf{A} = \begin{bmatrix} a & b \\ b & c \end{bmatrix},$$

with $a \geq c$, has eigenvalues $\lambda_1 \geq \lambda_2 > 0$, then the $\mathbf{A}_k$'s converge to diag (λ_1, λ_2).

(a) We first show that this iteration is well-defined, i.e., $\mathbf{G}_k\mathbf{G}_k^T$ is symmetric and positive definite. By construction, $\mathbf{G}_k\mathbf{G}_k^T$ is symmetric, i.e., $\left(\mathbf{G}_k\mathbf{G}_k^T\right)^T = \left(\mathbf{G}_k^T\right)^T\mathbf{G}_k^T = \mathbf{G}_k\mathbf{G}_k^T$, i.e., the Cholesky decomposition is symmetric.
To show that $\mathbf{G}_k\mathbf{G}_k^T$ is positive-definite, let $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \neq \mathbf{0}$ be given. Then, we see that

$$\mathbf{x}^T\mathbf{G}_k\mathbf{G}_k^T\mathbf{x} = \mathbf{y}^T\mathbf{y},$$

where $\mathbf{y} = \mathbf{G}_k^T\mathbf{x}$. Since $\mathbf{G}_k$ is full-rank, $\mathbf{y} \neq 0$, so we have that $\mathbf{y}^T\mathbf{y} > 0$, as desired.

(b) Let $\mathbf{A}$ be given as above. We see that $\mathbf{A}_k = \mathbf{G}_k\mathbf{G}_k^T$, where

$$\mathbf{A}_k = \begin{bmatrix} a_k & b_k \\ b_k & c_k \end{bmatrix},$$
$$\mathbf{L}_k = \begin{bmatrix} \sqrt{a_k} & 0 \\ \frac{b_k}{\sqrt{a_k}} & \frac{\sqrt{c_k a_k - b_k^2}}{\sqrt{a_k}} \end{bmatrix},$$

where $k = 0, 1, 2, \ldots$ and $\mathbf{A}_0 = \mathbf{A}$. We also note that in the iteration, $\mathbf{A}_k = \mathbf{G}_k\mathbf{G}_k^T$ and $\mathbf{A}_{k+1} = \mathbf{G}_k^T\mathbf{G}_k$. Rearranging the expression for $\mathbf{A}_{k+1}$, we see that

$$\begin{aligned} \mathbf{A}_{k+1} = \mathbf{G}_k^T\mathbf{G}_k &\implies \mathbf{G}_k\mathbf{A}_{k+1} = \mathbf{G}_k\mathbf{G}_k^T\mathbf{G}_k \\ &\implies \mathbf{G}_k\mathbf{A}_{k+1} = \mathbf{A}_k\mathbf{G}_k \\ &\implies \mathbf{A}_{k+1} = \mathbf{G}_k^{-1}\mathbf{A}_k\mathbf{G}_k, \end{aligned}$$

i.e., that $\mathbf{A}_k$ and $\mathbf{A}_{k+1}$ are similar matrices. Thus, the iteration produces a sequence of similar matrices, which means that the eigenvalues of $\mathbf{A}_{k+1}$ and $\mathbf{A}$ are the same for each k. Furthermore, the ordering of the eigenvalues is also preserved.
It remains to show that the off-diagonal entries of the $\mathbf{A}_k$'s are monotonically decreasing with every iteration.

We see that

$$\begin{aligned}\mathbf{A}_{k+1} &= \mathbf{G}_k^T \mathbf{G}_k \\ &= \begin{bmatrix} a_k + \frac{b_k^2}{a_k} & \frac{b_k\sqrt{c_k a_k - b_k^2}}{a_k} \\ \frac{b_k\sqrt{c_k a_k - b_k^2}}{a_k} & \frac{c_k a_k - b_k^2}{a_k} \end{bmatrix}\end{aligned}$$

for $k = 0, 1, 2, \ldots$. It suffices to show that $\frac{b_k\sqrt{c_k a_k - b_k^2}}{a_k} < b_k$, which is equivalent to showing that

$$\frac{b_k^2\left(c_k a_k - b_k^2\right)}{a_k^2} < b_k^2$$

We see that

$$\begin{aligned}\frac{b_k^2\left(c_k a_k - b_k^2\right)}{a_k^2} - b_k^2 &= \left(\frac{\left(c_k a_k - b_k^2\right)}{a_k^2} - 1\right) b_k^2 \\ &= \left(\frac{\left(c_k a_k - b_k^2 - a_k^2\right)}{a_k^2}\right) b_k^2 \\ &= \left(\frac{\left(c_k - a_k\right) a_k - b_k^2}{a_k^2}\right) b_k^2 \\ &< 0\end{aligned}$$

because $a_k \geq c_k > 0$ and $b_k^2 > 0$. Thus, the off-diagonal entries of the $\mathbf{A}_k$'s are monotonically decreasing with each iteration, and the off-diagonal entries are bounded below by 0. Thus, the $\mathbf{A}_k$'s tend to the diagonal matrix diag (λ_1, λ_2) as $k \to \infty$, as desired.

2. Compute a QR step with the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & \epsilon \\ \epsilon & 1 \end{bmatrix}$$

(a) without a shift.
We can write the above matrix as $\mathbf{A} = \mathbf{QR}$, where

$$\begin{aligned}\mathbf{Q} &= \frac{1}{\sqrt{4+\epsilon^2}} \begin{bmatrix} 2 & \epsilon \\ \epsilon & -2 \end{bmatrix}, \\ \mathbf{R} &= \frac{1}{\sqrt{4+\epsilon^2}} \begin{bmatrix} 4+\epsilon^2 & 3\epsilon \\ 0 & \epsilon^2 - 2 \end{bmatrix}\end{aligned}$$

One QR step produces the matrix $\mathbf{A}_1 = \mathbf{RQ}$, which is given by

$$\begin{aligned}\mathbf{A}_1 &= \frac{1}{4+\epsilon^2} \begin{bmatrix} 8+5\epsilon^2 & -2\epsilon+\epsilon^2 \\ \epsilon^3 - 2\epsilon & -2\epsilon^2 + 4 \end{bmatrix} \\ &\sim \frac{1}{4} \begin{bmatrix} 8 & -2\epsilon \\ -2\epsilon & 4 \end{bmatrix},\end{aligned}$$

where we said that ϵ^2 and ϵ^3 are asymptotically like 0.

(b) with the shift $\mu = 1$
We can write the shifted matrix $\mathbf{A} - \mathbf{I}$ as $\mathbf{A} - \mathbf{I} = \mathbf{QR}$, where

$$\begin{aligned}\mathbf{Q} &= \frac{1}{\sqrt{1+\epsilon^2}} \begin{bmatrix} 1 & \epsilon \\ \epsilon & -1 \end{bmatrix} \\ \mathbf{R} &= \frac{1}{\sqrt{1+\epsilon^2}} \begin{bmatrix} 1+\epsilon^2 & \epsilon \\ 0 & \epsilon^2 \end{bmatrix}.\end{aligned}$$

One QR step with the shift produces the matrix $\mathbf{A}_1 = \mathbf{RQ} + \mathbf{I}$, which is given by

$$\begin{aligned}\mathbf{A}_1 &= \frac{1}{1+\epsilon^2}\begin{bmatrix} 1+2\epsilon^2 & \epsilon^3 \\ \epsilon^3 & -\epsilon^2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &\sim \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix},\end{aligned}$$

where we said that ϵ^2 and ϵ^3 are asymptotically like 0.

Which approach appears to be better?

The shifted approach appears to be better because with the shift, one step of QR iteration recovers a matrix is much closer to a diagonal matrix than without the shift.

3. Implement the QR iteration for a real symmetric tridiagonal matrix and demonstrate its performance on a 100×100 example.

   We test the performance of the QR iteration on the 100×100 tridiagonal matrix $\mathbf{A}$, where $\mathbf{A}$ has ones on the sub- and super-diagonals, twos on the main diagonal, and zeros everywhere else.

   We compute the eigenvalues of $\mathbf{A}$ using the QR iteration and MATLAB's `eigs` function, and the absolute value of the difference of the two sets of eigenvalues is computed in the attached code.

   We see that the QR iteration, implemented using the algorithm given in class, does not recover the large eigenvalues of $\mathbf{A}$ well, with the difference between the two sets of computed eigenvalues being on the order of 10^{-3}.

```
clc;
clear;
```

Question 1

```
a = 2;
b = 1;
c = 1;

C = [a b; b c];

iters = 1;
C_k = chol_alg(C, iters);

disp(C_k);
```

Problem 2

```
eps = 1e-10;
B = [2 eps; eps 1];

% Compute one step of the QR algorithm with and without a shift
iters = 1;
B_k_1 = qr_alg(B, 0, iters);
B_k_2 = qr_alg(B, 1, iters);

% disp(B_k_1);
% disp(B_k_2);
```

Question 3

```
n = 99;
sub_diag  = ones(n, 1);
main_diag = 2*ones(n+1, 1);
A = diag(main_diag) + diag(sub_diag, -1) + diag(sub_diag, 1);

iters = 1000;
A_k = qr_alg(A, 0, iters);


% Compare the error between the eigenvalues returned by the QR iteration
% and MATLAB's eigs function
E_computed = diag(A_k);
E_actual   = eigs(A, n+1);

disp(abs(E_computed - E_actual));

function [A_prev] = chol_alg(A, iters)
    A_prev = A;
    for k = 1:iters
        A_prev = chol(A_prev); A_prev = A_prev*A_prev';
    end
end
```

```
function [A_prev] = qr_alg(A, mu, iters)
    A_prev = A;
    for k = 1:iters
        [Q_prev, R_prev] = qr(A_prev - mu*eye(size(A)));
        A_prev = R_prev*Q_prev + mu*eye(size(A));
    end
end
```

```
   2.500000000000000   0.500000000000000
   0.500000000000000   0.500000000000000

   0.002028075561455
   0.003111603508338
   0.002288953958013
   0.000027949547963
   0.001900416781986
   0.002105722675200
   0.001487381051598
   0.000903995085260
   0.000511282540006
   0.000274624490019
   0.000141311812393
   0.000070052322813
   0.000033599495316
   0.000015645452863
   0.000007091898202
   0.000003135983932
   0.000001354961029
   0.000000572730654
   0.000000237044719
   0.000000096124615
   0.000000038206343
   0.000000014887584
   0.000000005687550
   0.000000002130116
   0.000000000781934
   0.000000000281265
   0.000000000099097
   0.000000000034144
   0.000000000011531
   0.000000000003817
   0.000000000001219
   0.000000000000388
   0.000000000000123
   0.000000000000026
   0.000000000000024
   0.000000000000027
   0.000000000000024
   0.000000000000004
   0.000000000000017
   0.000000000000002
   0.000000000000011
   0.000000000000016
```

0.000000000000013
0.000000000000004
0.000000000000008
0.000000000000001
0.000000000000014
0.000000000000005
0.000000000000004
0.000000000000018
0.000000000000003
0.000000000000003
0.000000000000004
0.000000000000002
0.000000000000007
0.000000000000003
0.000000000000003
0.000000000000014
0.000000000000002
0.000000000000002
0.000000000000003
0.000000000000004
0.000000000000003
0.000000000000002
0.000000000000000
0.000000000000004
0.000000000000007
0.000000000000003
0.000000000000001
0.000000000000001
0.000000000000003
0.000000000000001
0.000000000000004
0.000000000000004
0.000000000000004
0.000000000000001
0.000000000000002
0.000000000000002
0.000000000000001
0.000000000000001
0.000000000000001
0.000000000000000
0.000000000000001
0.000000000000001
0.000000000000001
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000
0.000000000000000

```
0.000000000000000
0.000000000000000
0.000000000000001
0.000000000000001
```

Published with MATLAB® R2021b