Math 584 Homework 4
Due Wednesday, Nov. 9
By Marvyn Bailly

Problem 1 *Exercise 10.1*

Solution. We wish to find the eigenvalues, determinate, and singular values of the Householder reflector F. Recall that F is given by

$$F = I - 2\frac{vv^*}{v^*v},$$

where F reflects across the hyperplane H which is orthogonal to v. First let's consider the eigenvalues geometrically. We know that F reflects and thus applying F twice will bring a vector back to its original location. Thus we know that the action is length preserving and so the eigenvalues must have absolute value 1. Note that F is Hermitian as,

$$F^* = (I - 2\frac{vv^*}{v^*v})^* = I - 2\frac{vv^*}{v^*v} = F.$$

Thus the eigenvalues must be real and thus are ± 1. Now we also know that each eigenvalue corresponds to orthogonal eigenvectors. Thus if F reflects an eigenvector x over H, the corresponding eigenvalue must be -1. Since the other eigenvectors will be orthogonal to x, they must be orthogonal to this v and thus will not be reflected. Therefore the other eigenvectors will have corresponding eigenvalues equal to 1. We can show this algebraically by observing that

$$\begin{aligned} Fv &= (I - 2\frac{vv^*}{v^*v})v \\ &= v - 2\frac{vv^*v}{v^*v} \\ &= v - 2v \\ &= -v, \end{aligned}$$

and thus v is an eigenvector with a corresponding eigenvalue of -1. Now if let u be a vector that is orthogonal to v, we will get,

$$\begin{aligned} Fu &= (I - \frac{vv^*}{v^*v})u \\ &= u - 2(0) \\ &= u, \end{aligned}$$

and thus u has a corresponding eigenvalue of 1. If we let U be a basis of vectors that are orthogonal to v, we will have all the eigenvectors that are orthogonal and they will all have corresponding eigenvalues of 1. To find the determinate of the F, recall that the determinate is equal to the product of the eigenvalues and thus is -1. We also know that the singular values are the absolute values of the eigenvalues of and thus they are all equal to 1. $\square$

Problem 2 *Exercise 10.4 (a) and (b)*

Solution. Consider the 2×2 orthogonal matrices

$$F = \begin{pmatrix} -\cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \quad J = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix},$$

where $\det(F) = -1$ and is a Householder reflector and $\det(J) = 1$ and is a Givens rotation.

a. We first wish to geometrically describe the effects of left-hand multiplication of F and J on $\mathbb{R}^2$. Let's let $v \in R^2$ such that v is of the form

$$v = \begin{pmatrix} r\cos(\phi) \\ r\sin(\phi) \end{pmatrix}$$

where r is some magnitude and ϕ some rotation. Then observe that

$$\begin{aligned} Fv &= \begin{pmatrix} -\cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} r\cos(\phi) \\ r\sin(\phi) \end{pmatrix} \\ &= \begin{pmatrix} -r\cos(\theta)\cos(\phi) + r\sin(\theta)\sin(\phi) \\ \sin(\theta)\, r\cos(\phi) + \cos(\theta)\, r\sin(\phi) \end{pmatrix} \\ &= \begin{pmatrix} -\frac{r}{2}(\cos(\theta+\phi) + \cos(\theta-\phi)) + \frac{r}{2}(\cos(\theta-\phi) - \cos(\theta+\phi)) \\ \frac{r}{2}(\sin(\theta+\phi) + \sin(\theta-\phi)) + \frac{r}{2}(\sin(\theta+\phi) + \sin(\phi-\theta)) \end{pmatrix} \\ &= \begin{pmatrix} -r\cos(\theta+\phi) \\ r\sin(\theta+\phi) \end{pmatrix}. \end{aligned}$$

Thus we see that F rotates vectors in R^2 counterclockwise by θ and then reflects them over the y-axis. Similarly consider,

$$\begin{aligned} Fv &= \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} r\cos(\phi) \\ r\sin(\phi) \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta)\, r\cos(\phi) + \sin(\theta)\, r\sin(\phi) \\ -r\sin(\theta)\cos(\phi) + r\cos(\theta)\sin(\phi) \end{pmatrix} \\ &= \begin{pmatrix} \frac{r}{2}(\cos(\theta+\phi) + \cos(\theta-\phi)) + \frac{r}{2}(\cos(\theta-\phi) - \cos(\theta+\phi)) \\ -\frac{r}{2}(\sin(\theta+\phi) + \sin(\theta-\phi)) + \frac{r}{2}(\sin(\theta+\phi) + \sin(\phi-\theta)) \end{pmatrix} \\ &= \begin{pmatrix} r\cos(\theta-\phi) \\ r\sin(\theta-\phi) \end{pmatrix}. \end{aligned}$$

Thus we see that J rotates vectors in R^2 clockwise by θ.

b. We wish to reproduce the QR factorization algorithm using Given rotations. Consider the following MatLab code.

```
for i=1:n       %loop through each column of the matrix
    %using m-k Given rotations to zero out entries below the main diagonal
    for j=2:m-k+1
        x=A(k:m,k);                         %get kth column
        r = sqrt(x(1)*x(1)+x(j)*x(j));      %compute r
        c=x(1)/r;                           %compute cos term
        s=x(j)/r;                           %compute sin term
        G=eye(m-k+1,m-k+1);                 %form the Givens rotations
        G(1,1)=c;
        G(j,j)=c;
        G(1,j)=s;
        G(j,1)=-s;
        A(k:m,k:n)=G*A(k:m,k:n);    %Mult by G to zero out the jth entry
    end
end
R=A;
```

Note that G is unitary since $G_{ij}G_{ij}^* = I$ as G_{ij}^* means that we are rotating by $-\theta$. Thus we are obtaining Q through,

$$Q = \left(\prod G_{ij}\right)^*.$$

□

Problem 3 *Exercise 12.1*

Solution. Consider the 202×202 matrix A such that $\|A\|_2 = 100$ and $\|A\|_F = 101$. We want to find the sharpest possible bound on the 2-norm condition number $\kappa(A)$. Recall that the condition number of a matrix A is given by

$$\kappa(A) = \|A\|\|A^{-1}\|,$$

and in the case that $\|\cdot\| = \|\cdot\|_2$ we have

$$\kappa(A) = \frac{\sigma_1}{\sigma_m},$$

where m is the rank of A. Since $\|A\|_2 = \sigma_1 = 100$ we know one of these values. Next notice that

$$\|A\|_F = \sqrt{\sum_i \sigma_i^2} = \sqrt{(100)^2 + \sigma_2^2 + \cdots + \sigma_m^2} = 101.$$

Since we are trying to find the sharpest lower bound, let's assume $m = 202$. This means that,

$$\begin{aligned} \|A\|_F = 101 &= \sqrt{(100)^2 + \sigma_2^2 + \cdots + \sigma_m^2} \\ 202 &= \sigma_2^2 \cdots + \sigma_{202}^2, \end{aligned}$$

and thus the greatest value of σ_{202} is 1. Therefore we have that

$$\kappa(A) = \frac{\sigma_1}{\sigma_m} \geq \frac{100}{1} = 100.$$

□

Problem 4 *What is the gap between 4 and the next larger double precision number? What is the gap between 41 and the next larger double precision number? How many IEEE double precision numbers are there between an adjacent pair of nonzero IEEE single precision numbers?*

Solution. First recall that 4 in binary is given by 100. Thus 4 as a double precision number is given by 1.00×2^2. Therefore the gap between 4 and the next double precision number is 2^{-50} since there are 52 bits to count with.
Next consider 41 which can be written as 101001. Thus 41 as a double precision number is given by 1.01001×2^5. Using the same logic as before, the gap is 2^{-47}.
Now we want to find the number of double precision numbers between adjacent pairs of nonzero singular precision. Recall that a singular precision number has 23 bits for counting while a double has 52. thus the number of double precision between is given by $2^{52-23} - 1 = 2^{29} - 1$. $\square$

Problem 5 *Question 5*

Solution.

a. We wish to determine the absolute and relative condition numbers for the problem of evaluating $f(x) = e^x$. Recall that the absolute condition number is given by

$$\hat{\kappa} = \|J(x)\|,$$

where J is the Jacobian and the relative condition number is

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|/\|x\|}.$$

In our case, the Jacobian of the function is just the derivative and thus the absolute condition number is given by

$$\hat{\kappa} = |J| = |f'| = e^x.$$

Thus at $x = -20$, $f(x)$ is absolutely well condition. Next we can find the relative condition number to be

$$\kappa = \frac{|e^x|}{|e^x/x|} = |x|.$$

Thus at $x = -20$, $f(x)$ is relatively well conditioned.

b. We wish to study the stability of the provided MatLab code. I began by appending

```
format long e
disp("Alg got")
disp(newsum)
disp("Exp got " )
disp(exp(x))
disp("Absolute Error is")
disp(abs(exp(x) - newsum))
disp("Relative Error is")
disp(abs(exp(x) - newsum)/exp(x))
```

which will compare the result of the algorithm with the exact solution to provide the absolute and relative error. Testing the algorithm for positive values provides expected error. For example, $x = 20$ has absolute error of $5.960464477539063 \times 10^{-08}$ and the relative error is $1.228543294929598 \times 10^{-16}$. For $x = 10$, the absolute error is $7.275957614183426 \times 10^{-12}$ and the relative error is $3.303279646387444 \times 10^{-16}$. Thus we see that the algorithm is stable for $x > 0$. But when we plug in $x = -20$ we get that the absolute error is $4.851651954097902 \times 10^{08}$ and the relative error is $2.353852668370199 \times 10^{17}$ which is the opposite behavior as expected in the previous part (i.e. ill-conditioned). To find the issue, let's take a deeper look at the algorithm. By adding the line of code

```
terms = [terms term];
```

to the while loop, we can collect all the terms that are being added together to create the series approximation. Then using

```
max(terms)
```

we can find the largest size of the terms being added together. In the case of $x = -20$ this term is $4.309980412182178 \times 10^{07}$. The corresponding term that is being added to is $2.281524587589338 \times 10^{08}$ which means that if we need to add these two large numbers, the machine error will be amplified. On the other hand, when we are computing for $x > 0$, while we still have a large term of $4.309980412182178 \times 10^{07}$, the corresponding term that it is being added to is very small and thus the computer addition does not have as large of a magnification of the machine error.

c. To avoid the issue within the algorithm, we must avoid the addition of the negative terms. Using the fact that $e^{-x} = \frac{1}{e^x}$, we can compute the absolute value of all inputs x and if x is negative, return 1 over the approximation. Applying these modifications and running the algorithm for $x = 20$ now gives an absolute error of $4.135903062765138 \times 10^{-25}$ and a relative error of $2.006596217642398 \times 10^{-16}$ which is well conditioned as desired. The modified algorithm is here,

```
function hw4Q5(x)

oldsum = 0;
newsum = 1; % First term in series.
term = 1;
terms = [];
n = 0;
while newsum ~= oldsum % Iterate until next term is negligible.
    n = n + 1;
    term = term * abs(x)/n; % This is x^n / n!
    terms = [terms term];
    oldsum = newsum;
    newsum = newsum + term;
end


if(x < 0)
    newsum = 1/newsum;
end
```

```
format long e
disp("Alg got")
disp(newsum)
disp("Exp got " )
disp(exp(x))
disp("Absolute Error is")
disp(abs(exp(x) - newsum))
disp("Relative Error is")
disp(abs(exp(x) - newsum)/exp(x))
disp("Max middle")
disp(max(terms))
```

□

Problem 6 *Question 6*

Solution.

a. We are given

$$0.1_{10} \approx 0.00011001100110011001100_2.$$

To convert the binary number into a fraction consider that

$$2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21},$$

which we can compute by getting a common denominator of 2^{21} for each term. This gives that,

$$0.00011001100110011001100_2 = \left(\frac{209715}{2097152}\right)_{10},$$

and let's call this fraction x.

b. To find the absolute error consider that

$$\left|x - \frac{1}{10}\right| = \left|\frac{209715}{2097152} - \frac{1}{10}\right| = \frac{1}{10485760}.$$

c. To find the time error in seconds after 100 hours of operation, consider that

$$|360000 - 3600000x| = \left|360000 - 3600000\left(\frac{209715}{2097152}\right)\right| = \frac{5625}{16384} \approx 0.34\text{s}.$$

Thus we see that the time error is around 0.34 seconds.

d. Now consider a 1991 Scud missile that travels around 3750 miles which is 1676.4 meters per second. Thus the missile would travel

$$\approx 1676.4 \cdot \frac{5625}{16384} \approx 575.5\text{m}.$$

Thus the missile travels around 575.5 meters in the time error.

□