HOMEWORK 3 SOLUTIONS

MATH 170A

Problem 0.1. Watkins 2.1.23

Solution. Let’s try $A = B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. Then $\|A\|_{\text{max}} = \|B\|_{\text{max}} = 1$, but $\|AB\|_{\text{max}} = 2$. Thus submultiplicativity fails. $\square$

Problem 0.2. Watkins 2.1.27

Solution. (a) This is a easy calculation, using homogeneity:

$$\frac{\|A(cx)\|}{\|cx\|} = \frac{|c|\|Ax\|}{|c||x|} = \frac{\|Ax\|}{\|x\|}.$$

(b) Here is one proof: We know that there is a vector $x$ that maximizes $\frac{\|Ax\|}{\|x\|}$, i.e., so that that fraction is equal to $\|A\|$. However, the previous result says we can scale the vectors, so we can rescale $x$ so that $\|x\| = 1$. That means that there is a vector $\|A\| = \|Ax\| = \max_{\|x\|=1} \|Ax\|$. (The denominator is 1, so it goes away.) $\square$

Problem 0.3. Watkins 2.1.28

Solution. (a) $\|I\|_F = \sqrt{\sum_{i=1}^n i} = \sqrt{n}$, and $\|I\|_2 = \max \frac{\|Ax\|}{\|x\|} = \max \frac{\|x\|}{\|x\|} = 1$.

(b) This is not a trivial proof, but it comes down to a calculation.

$$\|A\|_2 = \max_{\|x\|=1} \|Ax\|$$

$$= \max_{\|x\|=1} \sqrt{\sum_j (\sum_i a_{ji}x_i)^2}$$

$$\leq \max_{\|x\|=1} \sum_j \sum_i |a_{ji}x_i|$$

$$\leq \max_{\|x\|=1} \sqrt{\sum |a_{ji}|^2 \sum |x_i|^2}$$

$$= \|A\|_F \cdot 1 = \|A\|_F$$

To get the third line, I used the standard triangle inequality. For the fourth line, I used the Cauchy-Schwartz inequality. $\square$
Problem 0.4. Watkins 2.1.32

Solution. To prove \( \|x\|_\infty \leq \|x\|_2 \), assume that \( x_1 \) is the biggest entry in \( x \). (Any choice would work.) Then
\[
\|x\|_\infty = |x_1| = \sqrt{|x_1|^2} \leq \sqrt{\sum |x_i|^2} = \|x\|_2,
\]
since adding more terms will just make it bigger.

To prove \( \|x\|_2 \leq \|x\|_1 \), simply use the triangle inequality:
\[
\|x\|_2 = \sqrt{\sum |x_i|^2} \leq \sum |x_i| = \|x\|_1.
\]

To prove \( \|x\|_1 \leq \sqrt{n}\|x\|_2 \), we prove the square of this inequality. The middle step is a standard inequality, similar to \((a + b)^2 \leq 2(a^2 + b^2)\).
\[
\|x\|_1^2 = (\sum |x_i|)^2 \leq n \sum |x_i|^2 = n\|x\|_2^2
\]

To prove \( \|x\|_2 \leq \sqrt{n}\|x\|_\infty \), we’ll again prove the square of this inequality, and that \( x_1 \) is the biggest entry.
\[
\|x\|_2^2 = \sum |x_i|^2 \leq \sum |x_1|^2 = n|x_1|^2 = n\|x\|_\infty
\]

Problem 0.5. Watkins 2.1.33

Solution. These are all just combining the previous inequalities. To increase the size of a fraction, you need to increase the numerator and/or decrease the denominator. I’ll do one of them.
\[
\|A\|_1 = \max \frac{\|Ax\|_1}{\|x\|_1} \leq \max \frac{\sqrt{n}\|Ax\|_2}{\|x\|_2} = \sqrt{n}\|A\|_2.
\]

Problem 0.6. Watkins 2.2.6

Solution. (a) This first one just uses that \( A = (A^{-1})^{-1} \).
\[
\kappa(A) = \|A\|\|A^{-1}\| = \|(A^{-1})^{-1}\|\|A^{-1}\| = \kappa(A^{-1})
\]

(b) This uses total homogeneity.
\[
\kappa(cA) = \|cA\|\|(cA)^{-1}\| = |c|\|A\|\|c^{-1}\|\|A^{-1}\| = \kappa(A)
\]

Problem 0.7. Watkins 2.2.15

Solution. (a) Since \( A_\alpha = \alpha I \),
\[
\|A_\alpha\| = \max \frac{\|\alpha Ix\|}{\|x\|} = \max |\alpha| \frac{\|x\|}{\|x\|} = |\alpha|.
\]
The others are similar.
(b) As we showed earlier, $\kappa(\alpha A) = \kappa(A)$. However, $\det(\alpha A) = \alpha^n \det(A)$. Thus, as $\alpha \to 0$, the determinant goes to zero, but the condition number stays the same.\hfill \Box

**Problem 0.8.** If you are solving $Ax = b$ and your $A$ and $b$ are both known to within 0.01%, and $\kappa(A) = 1000$, how accurate do you know your solution $\hat{x}$ is?

**Solution.** Since $0.0001 = \|\delta A\|/\|A\| \leq 1/\kappa(A) = 0.001$, we can use the estimate (2.3.10) from the book. Using that, we get that the error is
\[
\frac{\|\delta x\|}{\|x\|} \leq \frac{1000(0.0001 + 0.0001)}{1 - 1000 \cdot 0.0001} = 0.2222,
\]
so the error is less than 22%.\hfill \Box

**Problem 0.9.** Watkins 2.4.3 (except rework 1.2.20b instead of 1.2.17. The high accuracy you find is only for the solution to the given $Ax = b$. If your spring coefficients or forces are measured incorrectly, you could still have a larger error.)

**Solution.** To remind you, the matrix is $A = \text{toeplitz}([-2, 1, 0, \ldots])$, where the $\cdots$ is 17 more zeros, and $b$ is the zero vector but with $b_5 = -1$ and $b_{16} = 1$. (You could have all numbers being the opposite sign as well.) If you set that up,
\[
\vec{x} \approx (.52, 1.0, 1.6, 2.1, 2.6, 2.1, 1.7, 1.2, .71, .24, -.24, -.71, -1.2, -1.7, -2.1, -2.6, -2.1, -1.6, -1, -1, -52).
\]
Using the actually calculated vector, and using it to calculate the residual, I got
\[
\hat{r} \approx (0, 0, 0, 4.4e - 16, 0, 0, 0, -1.1e - 16, -5.6e - 17, 5.6e - 17, -1.1e - 16, 0, 0, 0, 0, 0, 0, 0, -1.1e - 16, 2.2e - 16).
\]
The two norm of the residual is $5.882 \cdot 10^{-16}$. The condition number for $A$ is $\kappa_2(A) = 178.06$. Thus the relative error is at most
\[
\frac{\|\delta x\|}{\|x\|} \leq 178.06 \frac{5.882 \cdot 10^{-16}}{1.4142} \approx 6.8 \cdot 10^{-14},
\]
so the error is quite small.\hfill \Box

**Problem 0.10.** Watkins 2.5.7

**Solution.** This is very similar to the calculation in the book:
\[
\hat{x} - \hat{y} = x(1 + \epsilon_1) - y(1 + \epsilon_2) = x - y + x\epsilon_1 - y\epsilon_2 = (x - y) \left( 1 + \frac{x}{x - y} \epsilon_1 - \frac{y}{x - y} \epsilon_2 \right).
\]
The coefficient of $(x - y)$ is very small unless $x - y$ is much smaller than $x$ or $y$. In that case, it could be large, meaning a large error. (Notice, though, that if $\epsilon_1 = \epsilon_2$, this equation reduces to $(x - y)(1 + \epsilon_1)$, so still a small error.)\hfill \Box
**Problem 0.11.** Watkins 2.5.12

*Solution.* (a) Since any floating point will be normalized so the first digit is 1 or bigger, $1.0 \cdots 0$ is the smallest significand possible. The absolute difference between two consecutive numbers is always $0.00 \cdots 01 \times \beta^t$, and so the maximum absolute error is half of that, $0.00 \cdots 005 \times \beta^t$. (Note I did not normalize it.) Thus, the way to maximize the relative error is to minimize the denominator, i.e. the number we are rounding to. (Really, we should divide by the number we are rounding from, but this problem seems to insist on dividing by the number we are rounding from.) However, since the $\beta^t$ will be the same for the original number and our absolute error, it is irrelevant, since it will cancel out. Thus, we can only minimize the significand, which we do by choosing $1.00 \cdots 00 \times \beta^t$.

(b) Dividing the absolute error by our number, we have

$$\frac{0.00 \cdots 005 \times \beta^t}{1.00 \cdots 00 \times \beta^t} = 0.00 \cdots 005 = 5 \times \beta^{-s} = \frac{1}{2} \beta^{1-s}.$$ 

□

**Problem 0.12.** Watkins 2.6.6

*Solution.* I wrote a short script to run this for me.

```matlab
for i=1:4
    n=4*i
    z=ones(n,1);
    H=hilb(n);
    b=H*z;
    xhat=H\b
    cond(H,2)
    err=norm(xhat-z,2)
    res=norm(b-H*xhat,2)
end
```

I won’t list the vectors, but the condition numbers were $1.6 \times 10^4$, $1.5 \times 10^{10}$, $1.7 \times 10^{16}$ and $4.2 \times 10^{17}$ for $n = 4, 8, 12, 16$ respectively. The errors were $3 \times 10^{-13}$, $9 \times 10^{-7}$, $8.7 \times 10^{-4}$ and 0.11. The residual norms were $1.6 \times 10^{-16}$ up through $1.1 \times 10^{-15}$, i.e., all very small. In other words, the residuals were small, but the error was large because the condition numbers were so large. □

**Problem 0.13.** Watkins 2.7.12

*Solution.* One example is $G = \begin{bmatrix} 1 & 2 \\ 10^{-16} & 2 \end{bmatrix}$ and $\delta G = \begin{bmatrix} 5 \times 10^{-16} & 10^{-16} \\ -10^{-16} & 3 \times 10^{-17} \end{bmatrix}$. In this case, $\|\delta G\|_\infty/\|G\|_\infty = 2 \times 10^{-16}$ while $|\delta g_{21}|/|g_{21}| = 1$. □

**Problem 0.14.** Watkins 3.1.5
Solution. (a) The system is 
\[
\begin{bmatrix}
1 & 1 \\
1 & 1.5 \\
1 & 2 \\
1 & 2.5 \\
1 & 3 \\
\end{bmatrix}
\begin{bmatrix}
b \\
m \\
\end{bmatrix}
= 
\begin{bmatrix}
1.1 \\
1.2 \\
1.3 \\
1.3 \\
1.4 \\
\end{bmatrix},
\]
where we are trying to find the line \( y = mt + b \) that best fits the data. 
(b) I found \((b, m) = (0.98, 0.14)\), so \( y \approx 0.14t + 0.98 \).

d) \( \|r\|_2 = 0.054772 \).\]

Problem 0.15. Watkins 3.1.8

Solution. My system is 
\[
\begin{bmatrix}
1 & 2.7 & 0.37 \\
1 & 4.5 & 0.22 \\
1 & 7.4 & 0.14 \\
1 & 12.2 & 0.082 \\
1 & 20.1 & 0.50 \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c \\
\end{bmatrix}
= 
\begin{bmatrix}
1.1 \\
1.2 \\
1.3 \\
1.3 \\
1.4 \\
\end{bmatrix},
\]
where I am trying to fit my data to the model \( a + be^t + ce^{-t} \). (I didn’t actually round in MATLAB.) Solving this, I get \((a, b, c) = (1.325, 0.00483, -0.641)\). Though the question didn’t ask for it, here’s a graph.
Problem 0.16. Watkins 3.5.1

Solution. (a) If \( x, y \in S^\perp \), and \( z \in S \), then
\[
\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle = 0 + 0
\]
and so \( x + y \) is orthogonal to anything in \( S \).
(b) If \( x \in S^\perp \), \( c \) is a number, and \( z \in S \), then
\[
\langle cx, z \rangle = c \langle x, z \rangle = c \cdot 0 = 0,
\]
so \( cx \) is orthogonal to anything in \( S \).  

Problem 0.17. Watkins 3.5.2

Solution. By definition of a basis, any vector \( x \in \mathbb{R}^n \) can be written as
\[
x = c_1q_1 + \cdots + c_nq_n.
\]
Since they are orthonormal, \( \langle x, q_i \rangle = c_i \). If \( s \in S \), then \( s = s_1q_1 + \cdots + s_kq_k \), and so, since we can break up inner products over addition, \( \langle x, c \rangle \) is zero (for every \( s \)) if and only if each of \( c_1, \cdots c_k \) are zero. (Otherwise that \( c_i \) term would show up in the inner product.)

In other words, \( x \in S^\perp \) if and only if \( c_1, \cdots c_k \) are zero, and so
\[
x = c_{k+1}q_{k+1} + \cdots + c_nq_n.
\]
In other words, if and only if \( x \) is in the span of \( q_{k+1}, \cdots q_n \).  

Problem 0.18. Watkins 3.5.7

Solution. (a) To show that it is a subspace, we need to show that if \( x, y \in N(A) \), and \( c \) is a number, then \( x + y \) and \( cx \) are in \( N(A) \). These are easy calculations:
\[
A(x + y) = Ax + Ay = 0 + 0 = 0
\]
\[
A(cx) = cAx = c \cdot 0 = 0
\]
(b) If \( x, y \in R(A) \), then there are vectors \( \hat{x} \) and \( \hat{y} \) such that \( A\hat{x} = x \) and \( A\hat{y} = y \). Then, \( x + y \) is in the range of \( A \) since \( A(\hat{x} + \hat{y}) = x + y \) and \( cx \) is in the range of \( A \) since \( A(c\hat{x}) = cx \). □

**Problem 0.19.** Watkins 3.5.11

*Solution.* We want to show that if \( x \in N(A^T) \), then \( x \in R(A^\perp) \). If \( x \in N(A^T) \), then \( A^T x = 0 \). To show that \( x \in R(A^\perp) \), we want to show that \( \langle x, y \rangle = 0 \) for any \( y \in R(A) \). If \( y \in R(A) \), then there is some \( \hat{y} \) so that \( A\hat{y} = y \). We calculate

\[
\langle x, y \rangle = \langle x, A\hat{y} \rangle = \langle A^T x, \hat{y} \rangle = \langle 0, \hat{y} \rangle = 0,
\]

and so \( x \in R(A^\perp) \). □

**Problem 0.20.** Watkins 3.5.18a

*Solution.* To prove “(thing A) if and only if (thing B),” you have to prove “If (thing A) then (thing B)” and “If (thing B) then (thing A).” We will do that.

If \( A\hat{x} = y \), then we can subtract the two equations to get \( Ax - A\hat{x} = y - y \), which can be simplified to \( A(x - \hat{x}) = 0 \), which exactly says \( x - \hat{x} \in N(A) \).

If \( x - \hat{x} \in N(A) \), then \( A(x - \hat{x}) = 0 \), which can be rearranged as \( A\hat{x} = Ax = y \). □

**Problem 0.21.** Watkins 3.5.23

*Solution.*

\[
\begin{bmatrix}
1 & 1 \\
2 & \end{bmatrix}
\begin{bmatrix}
x \\
x
\end{bmatrix}
=
\begin{bmatrix}
1 & 1 \\
2 & \end{bmatrix}
\begin{bmatrix}
9 \\
5
\end{bmatrix},
\]

and so \( x = 7 \). □

**Problem 0.22.** Write a function in MATLAB that takes as input a \( 2 \times 2 \) matrix \( A \) and return as output the approximate matrix 2-norm. Compute the approximate matrix 2-norm, by multiplying \( A \) by 100 unit vectors, \( x_k \) for \( k = 1, 2, ..., 100 \), pointed in directions \( (2 \times \pi/100) \times k \) radians. The approximate matrix 2-norm is the maximum of the 100 resultant vector norms, \( \|Ax_k\|_2 \). (Note: this is a conceptual assignment and isn’t how mathematicians usually calculate the matrix 2-norm. We’ll learn a better way later.) To test your code, if you enter the matrix

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix},
\]

you should get an output of 5.4644, which is close to the “true” value 5.4650.

*Solution.* Here is my solution.

```matlab
function answer = twonorm(A)

answer=0;
for k=1:100
```


\[
\text{angle} = 2\pi k / 100;
\]

\[
\text{product} = A \left[ \cos(\text{angle}); \sin(\text{angle}) \right];
\]

\[
\text{if } \text{norm}(\text{product}) > \text{answer}
\]

\[
\text{answer} = \text{norm}(\text{product});
\]

\[
\text{end}
\]

\[
\text{end}
\]

**Problem 0.23.** What does $fl(xy)$ mean? Why do we care about that?

*Solution.* $fl(xy)$ means that you do the multiplication $xy$, then store it as a floating point. Of course, since floating points have finite precision, this usually involves rounding, and thus $fl(xy)$ represents an approximate calculation on the potentially exact data $x$ and $y$.

We care about it because that’s what computers do. We need to understand this so that we can figure out when catastrophic rounding errors could happen on a computer system.

**Problem 0.24.** Explain the difference(s) between backwards stability and well-conditioned. Also, if you know both, what can you conclude?

*Solution.* Backwards stable means that doing the algorithm with exact data but approximate calculations (such as with floating points) is equivalent to doing the algorithm with approximate data but exact calculations.

Well-conditioned means that doing an exact calculation with approximate data will have well-bounded error.

Together, they mean that the algorithm, using the well-conditioned matrix, will produce an accurate answer, even on a computer with approximate calculations.

**Problem 0.25.** Explain why it is generally impossible to numerically (i.e., on a computer, using floating point calculations) tell the difference between ill-conditioned and singular matrices.

*Solution.* If a matrix is singular, when doing Gaussian elimination (for example), eventually one of your pivots should be zero. However, due to rounding errors, the pivot will likely not be exactly zero, and so elimination can continue.

In an ill-conditioned matrix, the pivots are not supposed to be zero, but they do get very, very small. However, because of rounding errors, a small pivot could end up being zero by accident!

Thus the difference between ill-conditioned matrices and singular matrices may change during the approximate floating point calculation, and so it is impossible to tell them apart.

**Problem 0.26.** Explain why using a tiny pivot can be disastrous.
Solution. If you unnecessarily use a tiny pivot, the calculated multipliers will be very large. Using a large multiplier when you subtract off the multiples of the given row means that the rows of your remaining submatrix will be nearly multiples of each other, since the original data there will have been swamped by the large multiple of the given row. Since they are nearly multiples of each other, in the next step of elimination, there will most likely be catastrophic cancelation, leading to large errors.

□

Problem 0.27. Explain why we use Gaussian elimination with partial pivoting even though it is not backwards stable.

Solution. The main reason is that while elimination can be unstable (if, say $\|U\| \approx 2^{n-1}$), it is very rare. Most matrices behave well, even if there are a few that behave badly. Plus, if you suspect that your matrix might be one of the few that misbehaves, you can always check, a posteriori, whether your calculation was small, but checking whether the residual and/or $\|U\|\|L\|/\|A\|$ were small.

We could, of course, use Gaussian elimination with complete pivoting, but that is computationally more expensive, so we tend to avoid that. □