Brad Nelson
Math 126
Homework #1
1/16/12

**1** If **u** and **v** are *m*-vectors, the matrix $A = I + \mathbf{u}\mathbf{v}^*$ is known as a *rank-one perturbation of the identity*. Show that if $A$ is non-singular, then its inverse has the form $A^{-1} = I + \alpha\mathbf{u}\mathbf{v}^*$ for some scalar $\alpha$, and give an expression for $\alpha$. For what **u** and **v** is $A$ singular? If it is singular, what is Nul $A$?

Suppose that $A$ is non-singular (its inverse exists). We will show that $A^{-1} = I + \alpha\mathbf{u}\mathbf{v}^*$ is the solution.

$$\begin{aligned}
I &= AA^{-1} \\
&= (I + \mathbf{u}\mathbf{v}^*)(I + \alpha\mathbf{u}\mathbf{v}^*) \\
&= I + \mathbf{u}\mathbf{v}^* + \alpha\mathbf{u}\mathbf{v}^* + \alpha(\mathbf{u}\mathbf{v}^*)(\mathbf{u}\mathbf{v}^*) \\
&= I + \mathbf{u}\mathbf{v}^* + \alpha\mathbf{u}\mathbf{v}^* + \alpha\mathbf{u}(\mathbf{v}^*\mathbf{u})\mathbf{v}^* \\
&= I + \mathbf{u}\mathbf{v}^* + \alpha\mathbf{u}\mathbf{v}^* + \alpha(\mathbf{v}^*\mathbf{u})\mathbf{u}\mathbf{v}^* \qquad \text{since } \mathbf{v}^*\mathbf{u} \text{ is a scalar} \\
&= I + (1 + \alpha + \mathbf{v}^*\mathbf{u}\alpha)\mathbf{u}\mathbf{v}^* \\
0 &= 1 + \alpha + \mathbf{v}^*\mathbf{u}\alpha \qquad \text{if } \mathbf{u}\mathbf{v}^* \neq \mathbf{0} \text{ (if it is, then } A = I = A^{-1}) \\
-1 &= \alpha(1 + \mathbf{v}^*\mathbf{u}) \\
\alpha &= \frac{-1}{1 + \mathbf{v}^*\mathbf{u}}
\end{aligned}$$

If $A$ is singular then,

$$\begin{aligned}
0 &= det|I + \mathbf{u}\mathbf{v}^*| \\
&= det|\mathbf{u}\mathbf{v}^* + I| \\
&= det|\mathbf{u}\mathbf{v}^* - (-1)I|
\end{aligned}$$

$\Rightarrow$ -1 is an eigenvalue of $\mathbf{u}\mathbf{v}^*$. In this case, $A\mathbf{u} = (I + \mathbf{u}\mathbf{v}^*)\mathbf{u} = I\mathbf{u} + \mathbf{u}\mathbf{v}^*\mathbf{u} = \mathbf{u} - \mathbf{u} = \mathbf{0}$, so $Nul(A) = span\{\mathbf{u}\}$. We also see, from the preceding equation, that $\mathbf{v}^*\mathbf{u} = -1$, which if plugged into the expression for $\alpha$ above yields an undefined number, which makes sense if $A$ is singular. ■

**2** The spectral radius $\rho(A)$ of a square matrix $A$ is the magnitude of its largest eigenvalue. Prove that $\rho(A) \leq ||A||_2$.

Let the spectral radius of $A$, $\rho(A) = |\lambda_1|$. Let $\mathbf{x_1}$ be the normalized eigenvector corresponding to the eigenvalue $\lambda_1$. The definition of the 2-norm of the matrix $A$ is

$$||A||_2 = \sup_{||\mathbf{x}||=1} ||A\mathbf{x}||$$

Since $\mathbf{x_1}$ is normalized, $\mathbf{x_1} \in \{\mathbf{x} : ||\mathbf{x}|| = 1\}$ and since $||A\mathbf{x_1}|| = ||\lambda_1\mathbf{x_1}|| = |\lambda_1|$, we see that $|\lambda_1| \in \{||A\mathbf{x}|| : ||\mathbf{x}|| = 1\}$. Thus, $\sup_{||\mathbf{x}||=1} ||A\mathbf{x}|| \geq |\lambda_1|$ which means $||A||_2 \geq \rho(A)$ ■

**3** Use the in-class worksheet on the following $m \times m$ bidiagonal matrix to answer the below.

$$A = \begin{pmatrix} 1 & 2 & & \\ & 1 & 2 & \\ & & 1 & \ddots \\ & & & \ddots \end{pmatrix}$$

(a) find a nontrivial lower bound on the condition number $\kappa(A)$
(b) predict the smallest $m$ such that roughly all significant digits will be lost in the solution $\mathbf{x}$ to a linear system $A\mathbf{x} = \mathbf{b}$ in double precision.
(c) demonstrate your last claim in a couple lines of code.

a) $\kappa(A) = ||A|| \cdot ||A^{-1}||$. From the in class worksheet, we have,

$$A = \begin{pmatrix} 1 & 2 & & \\ & 1 & 2 & \\ & & 1 & \ddots \\ & & & \ddots \end{pmatrix} \Rightarrow A^{-1} = \begin{pmatrix} 1 & -2 & 4 & \dots & (-2)^{m-1} \\ & 1 & -2 & \dots & (-2)^{m-2} \\ & & 1 & \dots & (-2)^{m-3} \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}$$

Using the definition of matrix 2-norm found in problem 2, we see that we can select the largest-norm column from a matrix to give a lower bound on the matrix norm. For example, selecting $\mathbf{x}$ to be the column vector with all zeros, except for a one in the last entry, we see that

$$A^{-1}\mathbf{x} = A^{-1}\begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} (-2)^{m-1} \\ \vdots \\ 1 \end{pmatrix}$$

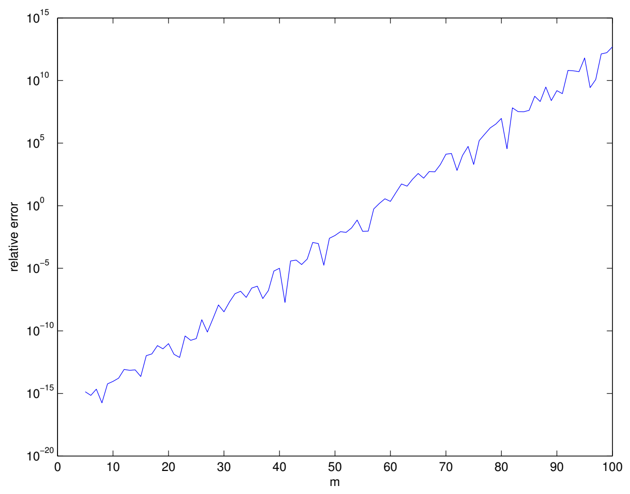
Since $||\mathbf{x}|| = 1$, using the logic in problem 2, $||A\mathbf{x}||$ offers a lower bound for $||A||$. Clearly, the above example selects the largest-norm column of $A^{-1}$, so we see that $||A^{-1}|| \geq \sqrt{(\sum_{i=0}^{m-1}(2^i)^2)} \geq 2^{m-1}$ by the Schwarz inequality. Using the same method, we see that $||A|| \geq 2$. Thus, $\kappa(A) \geq 2^m$.

b) Looking at the operation, $\mathbf{x} = A^{-1}\mathbf{b}$, since we know $A^{-1}$ can be stored exactly, we have the equation

$$\begin{aligned} x + \delta x &= A^{-1}(b + \delta b) \\ &= A^{-1}b + A^{-1}\delta b \\ \delta x &= A^{-1}\delta b \qquad \text{since } x = A^{-1}b \end{aligned}$$

Using the schwarz inequality, $||\delta x|| \leq ||A^{-1}|| \cdot ||\delta b||$. Since we have a lower bound on $||A^{-1}||$, for estimation purposes, let us just assume that $||\delta x|| \approx 2^{m-1}||\delta b||$. In double precision, 53 digits are stored, so $\delta b \approx 2^{-53}$. Thus, $||\delta x|| \approx 2^{m-54}$. Since roughly all the significant digits of $x$ are lost when $||\delta x|| \approx 1$, we can expect this to happen when $m \approx 54$.

c) To test the previous hypothesis, I wrote a MATLAB script which graphed the relative error in $x$ to the size of $m$ (see toeplitz.m).



As the graph shows, the relative error in $x$ is about 1 when $m$ is in the upper 50s. This is fairly close to the prediction of $m \approx 54$. ■

**4** How many nested loops are implied by each of the following MATLAB commands?

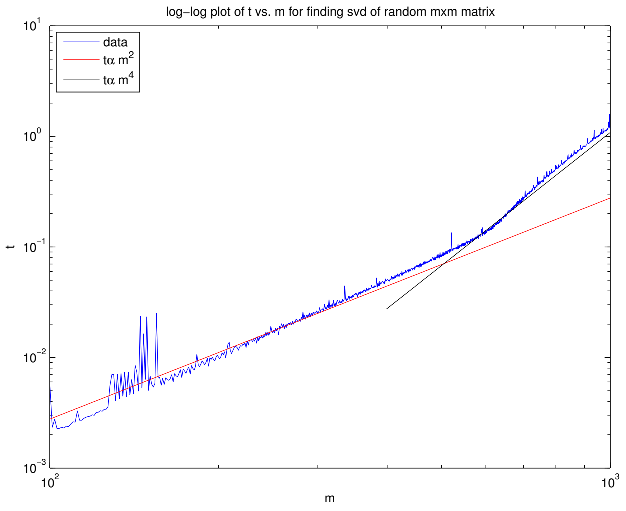
`A = rand(100, 100);`
2 loops are needed, one for each index of the array.
`x = 1 : 100;`
1 loop is needed for the array index.
`b = A * x';`
2 loops needed. The outer loop would run through the row index of the matrix $A$ and the vector $b$, the other would run through the column index of the matrix $A$ and the column index of $x$ (the row index of $x'$) to take the sum of the products.
`B = A * A;`
3 loops needed. The first two loops would run through the rows and columns of $B$ (they would be nested, but the order is arbitrary). The third loop would sum over the product of the components of the corresponding row and column of $A$. ■

**5** Give an exact formula, in terms of $\beta$ and $t$ for the smallest positive integer $n$ that does not belong to the floating-point system **F**, and compute $n$ for IEEE single- and double-precision. Give one line of code, and its output, which demonstrates that this is indeed the case for double-precision.

Since a computer can describe an integer $1 \leq m \leq \beta$, and describe powers of the exponent $(m/\beta^t)\beta^e$ where $e \in \mathbb{Z}$, the smallest integer that does not belong to the floating point system would be the first number for which $e > t$. This would be $n = \beta^t + 1$.
Single-precision: $\beta = 2, t = 24 \Rightarrow n = 2^{24} + 1 = 16\,777\,217$
Double-precision: $\beta = 2, t = 53 \Rightarrow n = 2^{53} + 1 = 9\,007\,199\,254\,740\,993$
Using MATLAB, we find that the line $(2^{53} + 1) - 2^{53}$ gives 0 as an answer, whereas $(2^{53} - 1) - 2^{53}$ gives -1. ■

**6** Measure how the time to compute the singular values of a random real dense $m \times m$ matrix scales with $m$, focusing on the range $10^2 \leq m \leq 10^3$. Produce a log-log graph of time vs $m$, and the simple power law to which it is asymptotic.

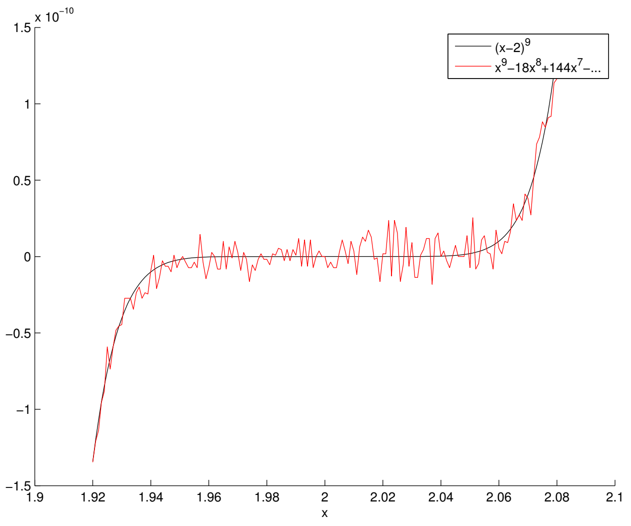
MATLAB script: svdtime.m



As seen on the graph, from $100 \leq m \leq 500$, the time taken is proportional to $m^2$. From $500 < m \leq 1000$, the time taken is proportional to $m^4$. ■

**7** Consider the polynomial $p(x) = (x-2)^9$.
(a) plot $p(x)$ for $x = 1.920, 1.921, 1.922, ..., 2.080$ evaluating $p$ via its coefficients 1, -18, 144,...
(b) overlay on your plot the same computed using $(x-2)^9$
(c) explain, including the size of the effect!

a,b) See poly.m for code.



c) The reason why the factored form of the polynomial is smooth compared to the form that uses coefficients has to do with machine imprecision. Floating point numbers are accurate to only $x(1+\epsilon_{mach})$, where $x \in \mathbb{R}$. Floating point operations are only accurate to within $(x \circledast y)(1+\epsilon_{mach})$ where $\circledast$ is a floating point operation corresponding to $* = +, -, \times, \div$ in $\mathbb{R}$. For double-precision arithmetic, $\epsilon_{mach} = 1.11 \times 10^{-16}$.
Factored form: since the quantity $(x-2)$ is very close to 0 in this domain, holding numbers in floating point and performing floating point operations will result in an absolute error $\ll 10^{-16}$, which is too small to be seen on the graph.
Coefficient form: now, since $x \approx 2$, performing operations on $x$ brings more imprecision into the answer. Just taking into consideration error in representation of $x$, the machine is accurate to $x(1+\epsilon_{mach})$. Performing an operation, such as evaluating the term $4032x^4$ in the coefficient representation of the polynomial

introduces error as follows: $x(1+\epsilon_{mach}) = x + x\epsilon_{mach}$. Since $x \approx 2$, let us introduce $\epsilon = x\epsilon_{mach} \approx 2\epsilon_{mach}$, so the number held in the computer is $x+\epsilon$. Now, $(x+\epsilon)^4 = x^4 + 4x^3\epsilon + O(\epsilon^2)$.... We can neglect lower order terms due to the very small size of $\epsilon$. Multiplying by the coefficient, the computer now holds $4032x^4 + 16128x^3\epsilon$, so the total error in the number is $16128x^3\epsilon$. Taking $x \approx 2$ again, and writing in terms of $\epsilon_{mach}$, we find the error to be $258048\epsilon_{mach} = 2.864 \times 10^{-11}$ for double precision. So we see that evaluating this term of the polynomial has given us an error of $10^{-11}$. Since the other terms will give errors on a similar, magnitude, summing them up will give an absolute error of $10^{-11}$, which is what is seen on the graph. ■