Math 566 **Homework 1** **Due Jan. 28, 2021**
Grady Wright

- Problems marked NLA (Numerical Linear Algebra) are from the book.

1. (**Matrix manipulations**) NLA: Question 1.1 (a)–(b)
2. (**Orthogonal matrices**) Two special types of matrices that we discussed in class and that we will use in several places in the course are orthogonal and unitary matrices:
   - A real square matrix Q is orthogonal if $Q^{-1} = Q^T$.
   - A complex square matrix is unitary if $Q^{-1} = Q^*$.

   Orthogonal and unitary matrices have the property that all of their eigenvalues, λ_j, satisfy $|\lambda_j| = 1$. This problem deals with orthogonal matrices.

   (a) Show that if a matrix Q is orthogonal and lower triangular, then it is diagonal.

   Hint: One way to proceed is to note that $QQ^T = I$, where I is the identity matrix. Perform forward substitution on this system, treating Q^T as an unknown, and show you can't get an upper triangular matrix.

   (b) What are its diagonal elements?
3. (**Rank one matrices**) Show the following
   - If A has rank one then $A = ab^T$ for some column vectors a and b.
   - If $A = ab^T$ for some column vectors a and b then A has rank one.

   Note: "Show" here means that you need to show the result mathematically.
4. (**Pythagorean theorem**) NLA: Question 2.2 (a)
5. (**Inverse of rank one-perturbation**) NLA: Question 2.6
6. (**Matrix norms**) Consider the following matrix:

$$A = \begin{bmatrix} 4 & -1 & 2 \\ 1 & 2 & 3 \\ -1 & 7 & -5 \end{bmatrix}$$

   (a) Compute the matrix 1−, 2−, ∞−, and Frobenius-norms of A.

   (b) For any n-by-n matrix A the following bounds on hold true

$$\begin{aligned} \frac{1}{\sqrt{n}}\|A\|_2 &\le \|A\|_1 \le \sqrt{n}\|A\|_2 \\ \frac{1}{\sqrt{n}}\|A\|_2 &\le \|A\|_\infty \le \sqrt{n}\|A\|_2 \\ \frac{1}{n}\|A\|_\infty &\le \|A\|_1 \le n\|A\|_\infty \\ \|A\|_1 &\le \|A\|_F \le \sqrt{n}\|A\|_2 \\ \rho(A) &\le \|A\|_p, \text{ for } p = 1, 2, \infty \end{aligned}$$

   Verify numerically that each of these bounds hold for the matrix A above by computing the norms explicitly. (Note that the last inequality holds true for any p).

7. (**Matrix multiplication**) Straightforward block partitioned 2×2 matrix multiplication $C = AB$ can be written

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

where

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

In the scheme discovered by Strassen (1969), the computations are rearranged into

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ P_2 &= (A_{21} + A_{22})B_{11} \\ P_3 &= A_{11}(B_{12} - B_{22}) \\ P_4 &= A_{22}(B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{12})B_{22} \\ P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= P_1 + P_4 - P_5 + P_7 \\ C_{12} &= P_3 + P_5 \\ C_{21} &= P_2 + P_4 \\ C_{22} &= P_1 - P_2 + P_3 + P_6 \end{aligned}$$

To make this into an efficient algorithm, each of the matrix-multiplications involved in computing $P_1, \ldots, P_7$ is again computed with Strassen's method, but now on matrices of half the size. This is applied recursively until it is cheaper to compute the matrix-matrix products directly, rather than with Strassen's method.

(a) Verify analytically (by hand or using Mathematica, MATLAB, or Maple) that the Strassen algorithm will give the same result as the traditional algorithm. Make sure to check that you never commute any matrices when doing the derivation.

(b) A MATLAB implementation of Strassen's algorithm is given below. Using this code, or translating it into another language, verify numerically that Strassen's method works for random input matrices of size $n = 2^m$, $m = 4, 5, 6$, and 7. Compare the answers this algorithm gives to the answer computed with regular matrix multiplication by computing the norm of the difference.

```
function c = strass(a,b)
nmin = 16;
[n,n] = size(a);
if n <= nmin;
   c = a*b;
else
   m = n/2; u = 1:m; v = m+1:n;
   p1 = strass(a(u,u)+a(v,v),b(u,u)+b(v,v));
   p2 = strass(a(v,u)+a(v,v),b(u,u));
   p3 = strass(a(u,u),b(u,v)-b(v,v));
   p4 = strass(a(v,v),b(v,u)-b(u,u));
   p5 = strass(a(u,u)+a(u,v),b(v,v));
   p6 = strass(a(v,u)-a(u,u),b(u,u)+b(u,v));
   p7 = strass(a(u,v)-a(v,v),b(v,u)+b(v,v));
   c = [p1+p4-p5+p7,p3+p5; p2+p4, p1-p2+p3+p6];
end
```

(c) One can show that the FLOPs required for Strassen's algorithm is $7 \cdot 7^m - 6 \cdot 4^m$ (i.e. $O(n^{\log_2 7}) \approx O(n^{2.81})$). Determine the value of $n = 2^m$ at which Strassen's algorithm has a lower FLOP count than the standard method of multiplying matrices. How practical does Strassen's algorithm seem?

(d) (Extra credit) Show that the total arithmetic operation count for Strassen's algorithm is $7 \cdot 7^m - 6 \cdot 4^m$.