

Module 2 Practice Problems (Square Linear Systems)

1. (Condition numbers; **FNC** 1.2.3) Calculate the (relative) condition number of each function, and identify all values of x at which $\kappa_f(x) \rightarrow \infty$ (including limits as $x \rightarrow \pm\infty$).

(a) $f(x) = \tanh(x)$.

(b) $f(x) = \frac{e^x - 1}{x}$.

(c) $f(x) = \frac{1 - \cos(x)}{x}$.

2. (Catastrophic cancellation; **FNC** 1.3.4)

(a) Find the (relative) condition number for $f(x) = (1 - \cos x)/\sin x$.

- (b) Explain carefully how many digits will be lost to cancellation when computing f directly by the formula in (a) for $x = 10^{-6}$.

- (c) Show that the mathematically identical formul

$$f(x) = \frac{2 \sin^2(x/2)}{\sin(x)}$$

contains no poorly conditioned steps for $|x| < 1$.

- (d) Using MATLAB, compute and compare the formulas from (a) and (c) numerically at $x = 10^{-6}$.

3. (More catastrophic cancellation; **FNC** 1.3.5) Let $f(x) = (e^x - 1)/x$.

- (a) Find the condition number $\kappa_f(x)$. What is the maximum of $\kappa_f(x)$ over $[-1, 1]$?

- (b) Use the “obvious” algorithm

$$y = (\exp(x) - 1) / x;$$

to compute $f(x)$ at 1000 evenly spaced points in the interval $[-1, 1]$.

- (c) Use the first 18 terms of the Taylor series

$$f(x) = 1 + \frac{1}{2!}x + \frac{1}{3!}x^2 + \frac{1}{4!}x^3 + \dots$$

to create a second algorithm, and evaluate it at the same set of points.

- (d) Plot the relative difference between the two algorithms as a function of x . Which one do you believe is more accurate, and why?

4. (Interpolation; **FNC** 2.1.1) Suppose you want to interpolate the points $(-1, 0)$, $(0, 1)$, $(2, 0)$, $(3, 1)$, and $(4, 2)$ by a polynomial of as low a degree as possible.

- (a) What degree should you expect this polynomial to be? (The degree could be lower in special cases where some coefficients are exactly zero.)
- (b) Write out a linear system of equations for the coefficients of the interpolating polynomial.
- (c) Use MATLAB to solve the system in (b) numerically.
5. (Hermite interpolant; **FNC 2.1.4**) Say you want to find a cubic polynomial $p(x)$ such that $p(0) = 0$, $p'(0) = 1$, $p(1) = 2$, and $p'(1) = -1$. (This is known as a *Hermite interpolant*.) Write out a linear system of equations for the coefficients of $p(x)$.
6. (Triangular substitution and stability; **FNC 2.3.6**) Consider the following linear system $A\mathbf{x} = \mathbf{b}$:

$$\underbrace{\begin{bmatrix} 1 & -1 & 0 & \alpha - \beta & \beta \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{b}}.$$

- (a) (by hand) Show that $\mathbf{x} = (1, 1, 1, 1, 1)^T$ is the solution for any α and β .
- (b) (computer) Using MATLAB, solve the system with $\alpha = 0.1$ and $\beta = 10, 100, \dots, 10^{12}$, making a table of the values of β and $|x_1 - 1|$. Write down your observation.
7. (Gaussian transformation matrices; Su20 final exam) Let $\{\mathbf{e}_j \in \mathbb{R}^n \mid j \in \mathbb{N}[1, n]\}$ be the standard unit basis of \mathbb{R}^n , i.e., $\mathbf{e}_1 = (1, 0, 0, \dots, 0)^T$, $\mathbf{e}_2 = (0, 1, 0, \dots, 0)^T$, ..., $\mathbf{e}_n = (0, 0, 0, \dots, 1)^T$. In this problem, we denote by G_j the Gaussian transformation matrix of the form

$$G_j = I + \sum_{i=j+1}^n a_{i,j} \mathbf{e}_i \mathbf{e}_j^T.$$

In addition, let $P(i, j) \in \mathbb{R}^{n \times n}$ be the elementary permutation matrix obtained by interchanging the i -th and the j -th rows of the same-sized identity matrix.

- (a) Let $1 \leq j < k < \ell \leq n$. Show that $P(k, \ell)G_jP(k, \ell) = I + \sum_{i=j+1}^n b_{i,j} \mathbf{e}_i \mathbf{e}_j^T$, where

$$b_{i,j} = \begin{cases} a_{i,j}, & \text{if } i \neq k \text{ and } i \neq \ell, \\ a_{\ell,j}, & \text{if } i = k, \\ a_{k,j}, & \text{if } i = \ell. \end{cases}$$

- (b) Show that $G_j^{-1} = I - \sum_{i=j+1}^n a_{i,j} \mathbf{e}_i \mathbf{e}_j^T$.

- (c) Let $j < k$. Show that $G_j G_k = I + \sum_{i=j+1}^n a_{i,j} \mathbf{e}_i \mathbf{e}_j^T + \sum_{i=k+1}^n a_{i,k} \mathbf{e}_i \mathbf{e}_k^T$.

- (d) Use the previous parts to find PLU factorization, $PA = LU$, by hand.

$$A = \begin{bmatrix} 5 & -5 & -2 \\ 5 & -2 & 7 \\ 10 & -3 & 18 \end{bmatrix}.$$

8. (Permutation matrix; **LM** 10.1–8) Let

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}.$$

Do the following by hand.

- Multiply A by a permutation matrix P to interchange the 1st and the 4th rows. Write out P explicitly.
- Multiply A by a permutation matrix P to interchange the 1st and the 4th columns. Write out P explicitly.
- Multiply A by two permutation matrices P and Q to interchange the 1st and the 4th rows and columns. Write out P and Q explicitly.
- Find a permutation matrix (more complicated than those above) which moves columns as described below:
 - 2nd to 1st;
 - 3rd to 2nd;
 - 4th to 3rd;
 - 1st to 4th;
 - 5th to 5th (unmoved).

Show that this permutation matrix is not its own inverse. What is the smallest positive integer k such that $P^k = I$? Write this permutation matrix as a product of *elementary* permutation matrices.

9. (Vectorizing mylu.m) Below is an instructional version of LU factorization code presented in lecture.

```
function [L,U] = mylu(A)
% MYLU    LU factorization (demo only--not stable!).
% Input:
%   A     square matrix
% Output:
%   L,U   unit lower triangular and upper triangular such that LU=A
n = length(A);
L = eye(n); % ones on diagonal
% Gaussian elimination
for j = 1:n-1
    for i = j+1:n
        L(i,j) = A(i,j) / A(j,j); % row multiplier
        A(i,j:n) = A(i,j:n) - L(i,j)*A(j,j:n);
    end
end
U = triu(A);
end
```

Consider the innermost loop. Since the different iterations in i are all independent, it is possible to *vectorize* this group of operations, that is, rewrite it without a loop. In fact, the

necessary changes are to delete the keyword `for` in the inner loop, and delete the following end line. (You should also put a semicolon at the end of `i = j+1:n` to suppress extra output.)

- (a) Make the changes as directed and verify that the function works properly.
 - (b) Write out symbolically (*i.e.*, using ordinary elementwise vector and matrix notation) what the new version of the function does in the case $n = 5$ for the iteration with $j = 3$.
10. (Proper usage of `lu`; **FNC 2.6.1**) Suppose that $A \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$. On the left is correct MATLAB code to solve $A\mathbf{x} = \mathbf{b}$; on the right is similar but incorrect code. Explain using mathematical notation exactly what vector is found by the right-hand version.

```
[L,U] = lu(A);
x = U \ (L\b);
```

```
[L,U] = lu(A);
x = U \ L \ b;
```

11. (Matrix norms; Sp20 midterm) Let

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}.$$

- (a) Calculate $\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$, and $\|A\|_F$ all by hand.
- (b) Imagine that MATLAB does not offer `norm` function and you are writing one for others to use, which begins with

```
function MatrixNorm(A, j)
% MatrixNorm    computes matrix norms
% Usage:
%   mat_norm(A, 1) returns the 1-norm of A
%   mat_norm(A, 2) is the same as mat_norm(A)
%   mat_norm(A, 'inf') returns the infinity-norm of A
%   mat_norm(A, 'fro') returns the Frobenius norm of A
```

Complete the program. (*Hint:* To handle the second input argument properly which can be a number or a character, use `ischaracter` and/or `strcmp`.)

12. (FLOP counting; **FNC 2.5.5**) This problem is about evaluation of a polynomial

$$p(x) = c_1 + c_2x + c_3x^2 + \cdots + c_nx^{n-1}.$$

- (a) Here is a little code to do the evaluation.

```
y = c(1);
xpow = 1;
for i = 2:n
    xpow = xpow * x;
    y = y + c(i)*xpow;
end
```

Assuming that x is a scalar, how many flops does this code take, as a function of n ?

- (b) Here is another code to do the same task.

```
y = c(n);           % This algorithm is called Horner's rule.  
for j = n-1:-1:1  
    y = y*x + c(j);  
end
```

Assuming that x is a scalar, how many flops does this code take, as a function of n ? Compare the count to the one from (a).