PS 1 Solutions

① (a) (i) $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \sum_{i=1}^{n} x_i y_i \Rightarrow$ n $\times$'s and $n-1$ $+$'s

Total of $2n-1$ FLOPs : $\mathcal{O}(n)$ [or $\mathcal{O}(2n)$].

(ii) $A\vec{x}$ = n dot products for a total of $n(2n-1)$ FLOPs : $\mathcal{O}(n^2)$ [or $\mathcal{O}(2n^2)$]

(iii) $A \cdot B$ = n^2 dot products for a total of $n^2(2n-1)$ FLOPs : $\mathcal{O}(n^3)$ [or $\mathcal{O}(2n^3)$].

(iv) Need to use Gauss-Jordan elimination on $[A \mid I_n]$:

$$\left[\begin{array}{cccc|cccc} \star & \star & \star & \star & 1 & 0 & 0 & 0 \\ \star & \star & \star & \star & 0 & 1 & 0 & 0 \\ \star & \star & \star & \star & 0 & 0 & 1 & 0 \\ \star & \star & \star & \star & 0 & 0 & 0 & 1 \end{array}\right]$$

← For simplicity, count all operations (even 0's) on right side of bar.

Column 1:

Scale pivot to 1: $(n-1)+n$ $\times$'s

$(n-1)$ Eliminations:

at $(n-1)+n$ $+$'s $(\star - c \cdot \star)$

and $(n-1)+n$ $\times$'s

Total: $(n-1) \cdot 2[(n-1)+n] + [(n-1)+n]$

Column 2:

Scale pivot to 1: $(n-2)+n$ $\times$'s

$(n-2)$ Eliminations:

at $(n-2)+n$ $+$'s

and $(n-2)+n$ $\times$'s

Total: $(n-2) \cdot 2[(n-2)+n] + [(n-2)+n]$

⋮

Column j:

Scale pivot to 1: $(n-j)+n$ $\times$'s

$(n-j)$ Eliminations:

at $(n-j)+n$ $+$'s

and $(n-j)+n$ $\times$'s

Total: $(n-j) \cdot 2[(n-j)+n] + [(n-j)+n]$

Total: $\sum_{j=1}^{n} 2(n-j)(2n-j) + \sum_{j=1}^{n} (2n-j)$

$$= 2\sum_{j=1}^{n} (2n^2 - 3jn + j^2) + \sum_{j=1}^{n} (2n-j)$$

$$= \sum_{j=1}^{n} \left(4n^2 - (6n+1)j + 2j^2\right)$$

$$= 4n^2 \sum_{j=1}^{n} 1 - (6n+1)\sum_{j=1}^{n} j + 2\sum_{j=1}^{n} j^2$$

$$= 4n^3 - (6n+1) \cdot \frac{n(n+1)}{2} + 2 \cdot \frac{n(n+1)(2n+1)}{6}$$

$$= 4n^3 - \frac{n}{2}(6n^2+7n+1) + \frac{n}{3}(2n^2+3n+1)$$

$$= 4n^3 - 3n^3 - \frac{7}{2}n^2 - \frac{1}{2}n + \frac{2}{3}n^3 + n^2 + \frac{1}{3}n$$

$$= \left(4-3+\frac{2}{3}\right)n^3 + \left(-\frac{7}{2}+1\right)n^2 + \left(-\frac{1}{2}+\frac{1}{3}\right)n$$

$$= \frac{5}{3}n^3 - \frac{5}{2}n^2 - \frac{1}{6}n \quad : \quad \mathcal{O}(n^3) \ \left[\text{or } \mathcal{O}\left(\tfrac{5}{3}n^3\right)\right].$$

(v) For an n-determinant, we expand:

$$\begin{vmatrix} a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & \ddots & \vdots \\ \star & \star & \cdots & \star \end{vmatrix} = \sum_{j=1}^{n} (-1)^{j+1} a_j \, |\star|_{(n-1)} .$$

That is, we produce n determinants of $(n-1)\times(n-1)$ matrices. This forces n multiplications. Likewise, an $(n-1)$-determinant forces $(n-1)$ multiplications. Therefore, there must be $n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 = n!$ multiplications. There is a comparable number of additions/subtractions. Hence, cofactor expansion is $\mathcal{O}(n!)$.

However, we can use Gaussian elimination to put the matrix in REF. The determinant is then the product of the pivots. Therefore, we can compute the determinant more quickly in $\mathcal{O}(n^3)$ FLOPs.

(b) As a more general case of ①(a)(iv), consider

$$\begin{bmatrix} \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star \end{bmatrix}_{m \times n}$$

, a "wide matrix" with $n > m$.

Assuming A is full rank

Column 1: $(m-1) \cdot 2(n-1) + (n-1)$

Column 2: $(m-2) \cdot 2(n-2) + (n-2)$

⋮

Column j: $(m-j) \cdot 2(n-j) + (n-j)$

Total: $\sum_{j=1}^{m} 2(m-j)(n-j) + \sum_{j=1}^{m} (n-j)$

$$= 2\sum_{j=1}^{m} \left(mn - (m+n)j + j^2\right) + \sum_{j=1}^{m} (n-j)$$

$$= 2\sum_{j=1}^{m} j^2 - [2(m+n)+1]\sum_{j=1}^{m} j + (2mn+n)\sum_{j=1}^{m} 1$$

$$= 2 \cdot \frac{m(m+1)(2m+1)}{6} - \frac{[2m+(n+1)]}{2} \cdot m(m+1) + mn(2m+1)$$

$$= \frac{2m}{3}(2m^2+3m+1) - \frac{m}{2}\left(2m^2 + (n+3)m + (n+1)\right) + 2m^2n + mn$$

$$= \frac{4}{3}m^3 + 2m^2 + \frac{2}{3}m - m^3 - \frac{(n+3)}{2}m^2 - \frac{(n+1)}{2}m + 2m^2n + mn$$

$$= \frac{1}{3}m^3 + 2m^2n + \frac{1-n}{2}m^2 + mn + \left(\frac{2}{3} + \frac{(n+1)}{2}\right)m$$

$$= \mathcal{O}\left(\frac{1}{3}m^3 + 2m^2n\right)$$

Consider now a "tall matrix" with $m > n$:

$$\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}_{m \times n}$$

. Gaussian elimination produces

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} .$$

Specifically, we do Gauss-Jordan elimination to get

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$

, the $n \times n$ identity augmented with a dense $(m-n)\times n$ matrix $\left[\dfrac{I_n}{\star}\right]$.

We don't need to eliminate the entries in $\star$ because we know they are zeroed out. Hence, this will take $\mathcal{O}(n^3)$ flops.

<u>To summarize:</u> If $m > n$, this is $\mathcal{O}(n^3)$.

If $m < n$, this is $\mathcal{O}\left(\frac{1}{3}m^3 + 2m^2n\right)$.

PROBLEM 2 (a)

```
function [ L, U ] = myLU( A )
% Author - Kevin Lamb
% Date - 1/26/19
%
% myLU() takes in an n x n matrix A and, if A is regular,
%       constructs the LU factorization of A, where
%       U is an upper-triangular matrix, L is a lower-
%       unitriangular matrix, and A = LU.

%% Input checks

[m n] = size(A);

% Initialize L and U matrices
L = zeros(n,n);
U = zeros(n,n);

% Check for square input
if m ~= n
    disp('Matrix must be a square!');
    return
end

%% Perform Gaussian elimination

for j = 1:n   % Loop over columns

    % Check if A is regular
    if A(j,j) == 0
        disp('Matrix is not regular!');
        disp('Ignore output.');
        return
    end     % End if

    for i = (j+1):n   % Loop over rows below j.

        % Get the entry for L
        mult = A(i,j)/A(j,j);

        % Row i -> Row i - mult * Row j
        for k = (j+1):n
            A(i,k) = A(i,k) - mult*A(j,k);
            L(i,j) = mult;
        end   % End k-for

    end;    % End i-for

    % Put a 1 on the diagonal for L
    L(j,j) = 1;

end;    % End j-for

%% Don't forget about U!

for i = 1:n
    for j = i:n
        U(i,j) = A(i,j);
    end
```

```
end

-------------------------------------------------------------------------
-----
TESTING "BAD INPUTS"

EDU>> A = [1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

EDU>> [L,U] = myLU(A)
Matrix must be a square!

L =

     0     0     0
     0     0     0
     0     0     0


U =

     0     0     0
     0     0     0
     0     0     0

EDU>> A = [1 1;1 1]

A =

     1     1
     1     1

EDU>> [L,U] = myLU(A)
Matrix is not regular!
Ignore output.

L =

     1     0
     1     0


U =

     0     0
     0     0

-------------------------------------------------------------------------
--
DEMOSTRATING WORKING ALGORITHM

EDU>> A = [1 2;2 1]

A =

     1     2
```

```
     2     1

EDU>> [L,U] = myLU(A)

L =

     1     0
     2     1


U =

     1     2
     0    -3

EDU>> L*U == A

ans =

     1     1
     1     1

EDU>> A = [1 2 3;4 5 6;7 9 8]

A =

     1     2     3
     4     5     6
     7     9     8

EDU>> [L,U] = myLU(A)

L =

    1.0000         0         0
    4.0000    1.0000         0
    7.0000    1.6667    1.0000


U =

     1     2     3
     0    -3    -6
     0     0    -3

EDU>> L*U == A

ans =

     1     1     1
     1     1     1
     1     1     1
```

PROBLEM 2 (b)

For n = 5:

```
EDU>> M

M =

     5     4     3     2     1
     4     5     4     3     2
     3     4     5     4     3
     2     3     4     5     4
     1     2     3     4     5

EDU>> [L,U] = myLU(M); U(5,5)

ans =

    1.7143
```

So that 1.7143 approximates the smallest pivot of M_5.

For n = 10:

```
EDU>> M

M =

    10     9     8     7     6     5     4     3     2     1
     9    10     9     8     7     6     5     4     3     2
     8     9    10     9     8     7     6     5     4     3
     7     8     9    10     9     8     7     6     5     4
     6     7     8     9    10     9     8     7     6     5
     5     6     7     8     9    10     9     8     7     6
     4     5     6     7     8     9    10     9     8     7
     3     4     5     6     7     8     9    10     9     8
     2     3     4     5     6     7     8     9    10     9
     1     2     3     4     5     6     7     8     9    10

EDU>> [L,U] = myLU(M); U(10,10)

ans =

    1.8333
```

So that 1.833 approximates the smallest pivot of M_10.

For n = 100, we simply show:

```
EDU>> [L,U] = myLU(M); U(100,100)

ans =

    1.9804
```

And for n = 1000, we have:

```
EDU>> [L,U] = myLU(M); U(1000,1000)

ans =

    1.9980
```

where we also remark that the program begins to noticeably hang for more time.

It would seem that the smallest pivot is approaching a limit of 2.

PROBLEM 2 (c)

```
EDU>> H = hilb(2)

H =

    1.0000    0.5000
    0.5000    0.3333

EDU>> [L,U] = myLU(H); U

U =

    1.0000    0.5000
         0    0.0833

EDU>> H = hilb(3)

H =

    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000

EDU>> [L,U] = myLU(H); U

U =

    1.0000    0.5000    0.3333
         0    0.0833    0.0833
         0         0    0.0056

EDU>> H = hilb(4)

H =

    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429

EDU>> [L,U] = myLU(H); U

U =

    1.0000    0.5000    0.3333    0.2500
         0    0.0833    0.0833    0.0750
         0         0    0.0056    0.0083
         0         0         0    0.0004

EDU>> H = hilb(5)
```

```
H =

    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111

EDU>> [L,U] = myLU(H); U

U =

    1.0000    0.5000    0.3333    0.2500    0.2000
         0    0.0833    0.0833    0.0750    0.0667
         0         0    0.0056    0.0083    0.0095
         0         0         0    0.0004    0.0007
         0         0         0         0    0.0000

EDU>> U(5,5)

ans =

   2.2676e-05
```

The last pivot of this matrix is a strong indicator of the singularity of the Hilbert matrices. The product of the pivots of U yields the determinant
of H, so we should track it as we increase n. Observe that indeed the four matrices
are indeed nonisingular, but it appears that this will be hard to determine even
for the next few values of n. Indeed, observe that

```
EDU>> H = hilb(100);
EDU>> [L,U] = myLU(H); U(100,100)

ans =

   1.7136e-17

EDU>> H = hilb(500);
EDU>> [L,U] = myLU(H); U(500,500)

ans =

  -1.2819e-17

EDU>> H = hilb(1000);
EDU>> [L,U] = myLU(H); U(1000,1000)

ans =

   6.1862e-18
```

This pivot is getting much smaller. More importantly, the other pivots in U are getting ever smaller as well. We should not trust these values going forward because they might be critically affecting the factorization.
For example:

```
EDU>> H = hilb(20);
EDU>> [L,U] = myLU(H); L

L =

  Columns 1 through 7

    1.0000         0         0         0         0         0         0
    0.5000    1.0000         0         0         0         0         0
    0.3333    1.0000    1.0000         0         0         0         0
    0.2500    0.9000    1.5000    1.0000         0         0         0
    0.2000    0.8000    1.7143    2.0000    1.0000         0         0
    0.1667    0.7143    1.7857    2.7778    2.5000    1.0000         0
    0.1429    0.6429    1.7857    3.3333    4.0909    3.0000    1.0000
    0.1250    0.5833    1.7500    3.7121    5.5682    5.6538    3.5000
    0.1111    0.5333    1.6970    3.9596    6.8531    8.6154    7.4667
    0.1000    0.4909    1.6364    4.1119    7.9301   11.6308   12.6000
    0.0909    0.4545    1.5734    4.1958    8.8112   14.5385   18.5294
    0.0833    0.4231    1.5110    4.2308    9.5192   17.2466   24.9118
    0.0769    0.3956    1.4505    4.2308   10.0792   19.7104   31.4675
    0.0714    0.3714    1.3929    4.2059   10.5147   21.9149   37.9858
    0.0667    0.3500    1.3382    4.1634   10.8467   23.8628   44.3167
    0.0625    0.3309    1.2868    4.1086   11.0933   25.5673   50.3599
    0.0588    0.3137    1.2384    4.0454   11.2693   27.0464   56.0528
    0.0556    0.2982    1.1930    3.9766   11.3876   28.3204   61.3608
    0.0526    0.2842    1.1504    3.9043   11.4583   29.4096   66.2697
    0.0500    0.2714    1.1104    3.8300   11.4901   30.3339   70.7791

  Columns 8 through 14

         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
    1.0000         0         0         0         0         0         0
    4.0000    1.0000         0         0         0         0         0
    9.5294    4.5000    1.0000         0         0         0         0
   17.6471   11.8421    4.9998    1.0000         0         0         0
   28.0960   23.8815   14.4040    5.4961    1.0000         0         0
   40.4582   40.9398   31.4266   17.1989    5.9204    1.0000         0
   54.2654   62.8983   57.7289   40.3601   19.8412    6.1256    1.0000
   69.0650   89.3339   94.2900   79.0918   49.5831   20.8855    2.4844
   84.4545  119.6436  141.4344  136.8714  102.8804   53.2158    1.7978
  100.0942  153.1437  198.9400  216.2677  187.5455  112.7978   -6.3265
  115.7089  189.1391  266.1736  318.8569  310.7399  210.2243  -30.3041
  131.0828  226.9669  342.2223  445.2685  478.4763  356.1750  -80.9837
  146.0522  266.0228  426.0069  595.3048  695.3453  560.6547 -171.0185

  Columns 15 through 20

         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
```

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1.0000	0	0	0	0	0
5.3490	1.0000	0	0	0	0
16.9203	2.5159	1.0000	0	0	0
41.1625	2.8560	5.0716	1.0000	0	0
84.6933	-3.0898	15.9262	1.4350	1.0000	0
154.8687	-24.2039	39.1507	-1.0538	4.6245	1.0000

This factorization may not be trustworthy after n gets too large.
In order to properly answer this question for arbitrary values of n,
we may need to appeal to some theory to study this question further.