Solutions to Assignment # 1

Grading. Each part of the multi-part problems (**P1**, **P3**, **P4**) counted as 3 points, and the others (**P2**, **Exercise 1.3**) as 5 points. Total of 31 points.

P1. **(a)** To compute the eigenvalues you can construct the characteristic polynomial $p(\lambda) = \det(\lambda I - A)$ and find its roots. I get $p(\lambda) = \lambda^3 - 5\lambda^2 + 4\lambda$ so $\lambda = 0, 1, 4$ exactly.

To compute the rank you may do row reduction, and then count the number of nonzero rows. To compute the determinant you may expand in minors. Or you can use the above eigenvalues: $\operatorname{rank}(A) = 2$ because that is the number of nonzero eigenvalues, and $\det(A) = 0$ because that is the product of eigenvalues.

Note A is not invertible. To invert the lower left two-by-two block B you can use Gauss-Jordan elimination on the augmented matrix $[B|I]$ until it is converted into $[I|B^{-1}]$. To solve $Av = b$ for the given b you can use row operations to yield an equivalent system with infinitely many solutions:

$$\begin{aligned} 2x_1 + x_2 + x_3 &= -1 \\ x_2 + 2x_3 &= -5 \end{aligned}$$

If we let $x_3 = s$ then $x_1 = 2 + s/2$, $x_2 = -5 - 2s$, and $x_3 = s$ for any $s \in \mathbb{R}$.

(b) Now check your work in MATLAB:

```
>> A = [2 1 1; 4 0 -2; 2 2 3];
>> eig(A)
ans =
   4.00000
  -0.00000
   1.00000
>> rank(A)
ans =  2
>> det(A)
ans = 0
>> inv(A)
warning: matrix singular to machine precision
ans =
       Inf       Inf       Inf  ...
>> format rat,  B = A(2:3,1:2);  inv(B)
ans =
        1/4          0
       -1/4        1/2
>> b = [-1 8 -6]';  A \ b
ans =
      20/21
     -17/21
     -44/21
```

Note that the last result is for $s = -44/21$; this is one of many correct answers. (*But why does* MATLAB *report this one?*)

Observation. MATLAB did not do these calculations the way you did them by hand! And, while MATLAB's methods all generalize to give stable and accurate results for large matrices, the by-hand methods you were taught *do not*. Thus this course!

P2. This code produced Figures 1 and 2:

matstats.m

```
% MATSTATS  Compute and plot functions of random matrices.

N = 10;           % number of tries
S = 8;            % maximum size is m = 2^S
mm = 2.^(1:S);    % yes, this works!

% compute values
for k = 1:S
    m = mm(k);
    for n = 1:N
        A = 20.0 * rand(m,m) - 10.0;
        ranks(k, n) = rank(A);
        if ranks(k, n) ~= m
            fprintf('WARNING:  rank(A) is *not* m')
        end
        norms(k, n) = norm(A);
        absdets(k, n) = abs(det(A));
        if absdets(k, n) > 1.0e308
            fprintf('WARNING:  |det(A)| overflow for m=%d\n', m)
        end
    end
end

% make plots
figure(1),  loglog(mm, norms, 'ko', 'markersize', 12)
xlabel('m'),  ylabel('norm(A)'),  axis([1, 512, 1, 1000])
xticks(mm),  xticklabels({'2','4','8','16','32','64','128','256'})
print('a1p2norms.pdf')
figure(2),  loglog(mm, absdets, 'ko', 'markersize', 12)
xlabel('m'),  ylabel('|det(A)|'),  axis([1, 512, 1.0, 1.0e308])
xticks(mm),  xticklabels({'2','4','8','16','32','64','128','256'})
yticks(10.0.^(0:50:300))
print('a1p2dets.pdf')
```

I did not plot ranks because they are boring. Of the 80 matrices, in every case $\text{rank}(A) = m$. Apparently these random matrices are always full rank (invertible).

Figure 1 shows all values of the $\|A\| =$ `norm(A)`. These norms vary a little because of the random entries, and they grow slowly with m.

By contrast, Figure 2 shows that the determinants $|\det(A)| =$`abs(det(A))` grow absurdly fast with dimension. The data for $m = 256$ is missing because overflow occurred. (Our machines can't handle numbers above about 10^{308}, but there is nothing meaningful in the world that is that large. There are about 10^{80} fundamental particles in the universe!)

P3. **(a)** Let $b = Av$ so $b \in \mathbb{C}^{m\times 1}$. The entries of b are computed by this algorithm:

$$b_j = \sum_{k=1}^{n} a_{jk} v_k$$

for $j = 1, \ldots, m$. The number of multiplications to compute a single entry b_j is thus n, and the number of additions is $n - 1$. Therefore the total number of multiplications and additions to compute Av is mn and $m(n-1)$, respectively.

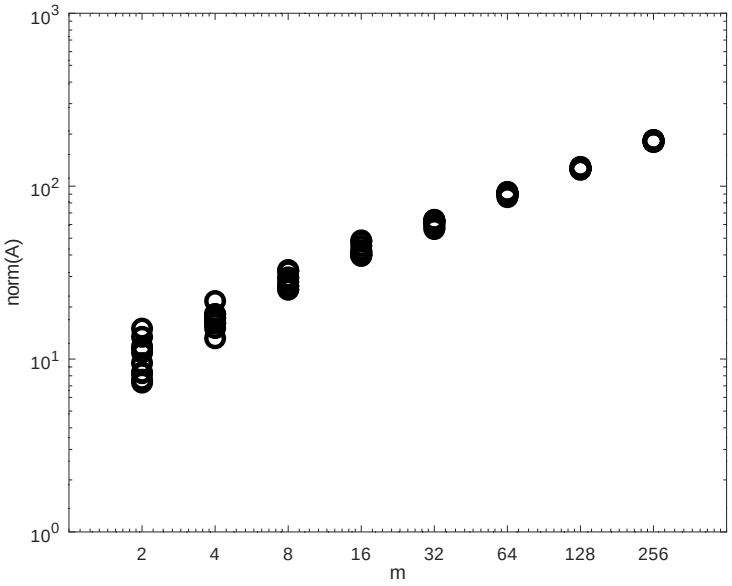

FIGURE 1. For **P2**: $\|A\|$ grows slowly with m.

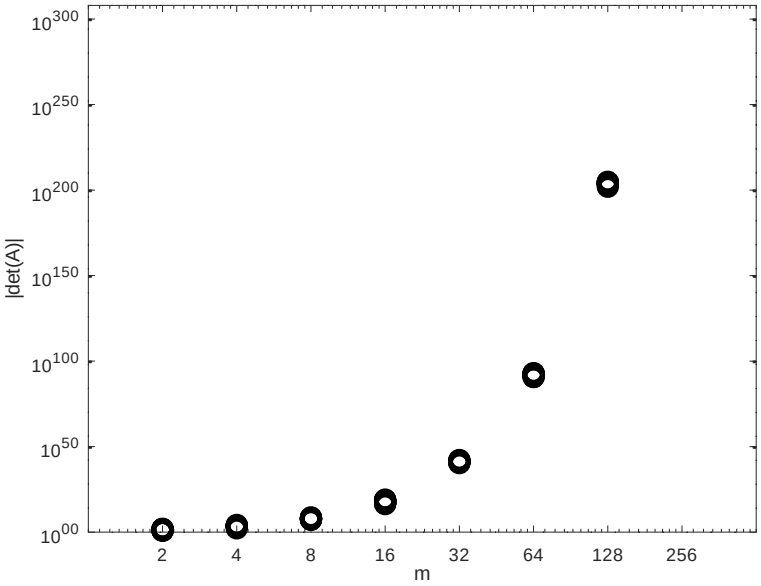

FIGURE 2. For **P2**: $|\det(A)|$ grows absurdly fast with m.

(b) My code looked like this:

matvec.m

```
function z = matvec(A,v)
% MATVEC  Computes  z = A * v.

if (size(v,2) ~= 1)
    error('v must be a column')
end
[m n] = size(A);
if (length(v) ~= n)
    error('length(v) must equal columns in A')
end

z = zeros(m,1);
for i = 1:m
    for j = 1:n
        z(i) = z(i) + A(i,j) * v(j);
    end
end
```

You may check that this works by hand on an example of your choice. To check that it works the same way as the built-in `A*v` on random matrices you could do this:

```
>> A = randn(4,3);  v = randn(3,1);
>> norm(matvec(A,v) - A*v)
ans = 0
```

Thus my `matvec.m` agrees with built-in multiplication.

(c) Next:

matmat.m

```
function C = matmat(A,B)
% MATMAT  Computes  C = A * B.

[m n] = size(A);
if (size(B,1) ~= n)
    error('number of rows of B must equal number of columns of A')
end
k = size(B,2);

C = zeros(m,k);
for i = 1:m
    for j = 1:k
        C(i,j) = A(i,1) * B(1,j);
        for s = 2:n
            C(i,j) = C(i,j) + A(i,s) * B(s,j);
        end
    end
end
```

This code does mkn multiplications and $mk(n-1)$ additions for a total of $mk(2n-1)$ operations. Finally, I compared against built-in multiplication on random matrices:

```
>> A = randn(3,4);  B = randn(4,3);
>> norm(matmat(A,B) - A*B)
ans = 0
```

Though it is *not* requested in the Assignment, it is interesting to compare speeds:

```
>> A = rand(51,87);  B = randn(87,64);
>> tic, C=A*B; toc, tic, C=matmat(A,B); toc
Elapsed time is 0.000834942 seconds.
Elapsed time is 3.68026 seconds.
>> A = rand(251,187);  B = randn(187,164);
>> tic, C=A*B; toc, tic, C=matmat(A,B); toc
Elapsed time is 0.00689316 seconds.
Elapsed time is 99.039 seconds.
```

The speed of my version is disastrous, 10^4 times slower! This `matmat.m` code will be much faster in MATLAB than in OCTAVE, and yet faster in Julia, but still not near the built-in. Why do you think the built-ins are so much faster?

P4. **(a)** The fundamental idea of this exercise is that

- operations that act on ROWs are implemented by LEFT multiplication
- operations on COLUMNs are RIGHT multiplications.

So here is what I did. I used MATLAB by starting with a 4×4 matrix with distinct integer entries. (Distinct entries makes it easier to follow through the steps.) For each operation I simply tacked on a row operation (on the left) or a column operation (on the right) as appropriate. I checked that each had the intended effect. Run the following code to see what I did.

manipulations.m

```
% MANIPULATIONS  Do some manipulations on a matrix by left and right
% matrix multiplications.

% matrix with convenient integer entries
B = magic(4)

% build and apply each factor
L1 = [0 0 1 0; 0 1 0 0; 1 0 0 0; 0 0 0 1];              L1*B
R2 = [1 0 0 0; 0 0 0 1; 0 0 1 0; 0 1 0 0];              L1*B*R2
R3 = eye(4); R3(3,3)=2;                                 L1*B*R2*R3
L4 = [1 0 1 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];           L4*L1*B*R2*R3
L5 = [1 -1 0 0; 0 1 0 0; 0 -1 1 0; 0 -1 0 1];     L5*L4*L1*B*R2*R3
R6 = [1 0 0 0; 0 1 0 0; 0 0 0 0; 0 0 1 1];        L5*L4*L1*B*R2*R3*R6
L7 = [0 1 0 0; 0 0 1 0; 0 0 0 1];              L7*L5*L4*L1*B*R2*R3*R6

% print resulting left and right factors
A = L7 * L5 * L4 * L1
C = R2 * R3 * R6
```

The result is the product

$$L_7 L_5 L_4 L_1 B R_2 R_3 R_6.$$

(b) If $A = L_7L_5L_4L_1$ and $C = R_2R_3R_6$ then the result of part **(a)** is the product ABC. Displaying these matrices allows us to check our work:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Exercise 1.3 in Lecture 1. *There are other ways to do it, but I think this was intended. Note that triangular matrices are invertible if and only if all of their diagonal entries are nonzero.*

If $R \in \mathbb{C}^{m\times m}$ is a nonsingular upper-triangular matrix then R^{-1} is upper-triangular.

Proof. Let $B = R^{-1}$, the unknown inverse. (*We know it exists, but we are trying to prove it is triangular.*) We write $I = BR$ and think of this product as the columns of R acting to give linear combinations of the (unknown) columns of B to produce the known columns e_j of the identity:

$$\left[\begin{array}{c|c|c|c} & & & \\ e_1 & e_2 & \dots & e_m \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ b_1 & b_2 & \dots & b_m \\ & & & \end{array}\right] \left[\begin{array}{c|c|c|c} r_{11} & r_{12} & \dots & r_{1m} \\ & r_{22} & \dots & r_{2m} \\ & & \ddots & \vdots \\ & & & r_{mm} \end{array}\right]$$

As equations for columns this says:

$$\begin{aligned} e_1 &= r_{11}b_1 \\ e_2 &= r_{12}b_1 + r_{22}b_2 \\ e_3 &= r_{13}b_1 + r_{23}b_2 + r_{33}b_3 \\ &\vdots \\ e_m &= r_{1m}b_1 + r_{2m}b_2 + r_{3m}b_3 + \cdots + r_{mm}b_m \end{aligned}$$

Since we know R is invertible it follows that these equations have unique solutions. That is, these equations determine the columns b_j.

The first two equations (i.e. equality in the first two columns) say

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} r_{11}b_{11} \\ r_{11}b_{21} \\ \vdots \\ r_{11}b_{m1} \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} r_{12}b_{11} + r_{22}b_{12} \\ r_{12}b_{21} + r_{22}b_{22} \\ \vdots \\ r_{12}b_{m1} + r_{22}b_{m2} \end{bmatrix}.$$

The first says $r_{11}b_{11} = 1$, so $b_{11} = 1/r_{11}$, and it says $b_{21} = b_{31} = \cdots = b_{m1} = 0$. Now that we know b_{i1} for $i = 1, 2, \dots, m$, we can use the second column equation to solve for b_{i2}. For $i = 1, 2$ we have $b_{12} = -(r_{12}b_{11})/r_{22}$ and $b_{22} = (1 - r_{12}b_{21})/r_{22} = (1 - 0)/r_{22} = 1/r_{22}$. However, for $i = 3, \dots, m$ we have $b_{i2} = -(r_{12}b_{i1})/r_{22} = 0/r_{22} = 0$.

Continuing in this way we get formulas for b_{ij}, and in particular we get $b_{ij} = 0$ if $i > j$. (That is, we can actually write down the entries of R^{-1}.) Thus $B = R^{-1}$ is upper triangular. $\square$