ECE 204 Project 1

Douglas Harder

January 15, 2026

1 Introduction

Stochastic matrices play a critical role in various areas of electrical and computer engineering, particularly in systems involving probabilistic transitions and network modeling. In signal processing, they are used to model and analyze Markov processes, enabling predictions about state transitions in communication channels affected by noise. In computer networks, stochastic matrices help model packet routing and congestion control by representing probabilities of data transmission between nodes. They are also foundational in machine learning algorithms for tasks such as random walks and reinforcement learning, where they capture the likelihood of transitions between states. Additionally, in control systems, stochastic matrices describe state evolution in uncertain environments, aiding in the design of robust controllers for systems affected by random disturbances. These matrices also underpin reliability analysis in fault-tolerant systems, modeling failure and recovery rates to ensure optimal performance.

In this project, we will review some of the tools and techniques used in linear algebra, we will then define column-stochastic matrices, and then you will be asked to explore certain characteristics of column-stochastic matrices.

A "Markov chain" describes a system that transitions between n possible states, where the probability of moving to the next state depends only on the current state and not on the sequence of previous states.

For example, consider a wireless sensor node that can be in one of two states: transmitting or idle. Based on previous data, the node's state changes based on the availability of data to send:

1. If the node is transmitting, there is a 0.4 (40%) chance it will remain transmitting in the next time step and a 0.6 (60%) chance it will switch to idle to save energy.
2. If the node is idle, there is a 0.9 (90%) chance it will remain idle and a 0.1 (10%) chance it will begin transmitting when new data arrives.

For example, if at time step $t = 0$, the node is idle, then there is 0.9 chance that it will remain idle, and a 0.1 chance that it will begin transmitting during time step $t = 1$.

Thus:

1. To determine the probability of the node is idle at time step $t = 2$, we must determine what is the likelihood that if the node was idle at during time step $t = 1$ that it will continue to be idle plus the likelihood that if the node was transmitting during time step $t = 1$ that it will transition to being idle during time step $t = 2$. This would be the calculation $0.9 \cdot 0.9 + 0.1 \cdot 0.6 = 0.87$.
2. To determine the likelihood that the node is transmitting at time step $t = 2$, we must determine what is the likelihood that if the node was transmitting during time step $t = 1$ that it will continue to transmit plus the likelihood that if the node was idle during time step $t = 1$ that it will begin transmitting during time step $t = 2$. This would be the calculation $0.1 \cdot 0.4 + 0.9 \cdot 0.1 = 0.13$. Therefore, at time step $t = 2$, there is an 87% chance it will be idle and a 13% chance that it will be transmitting.

With next step $t = 3$, we must calculate $0.87 \cdot 0.9 + 0.13 \cdot 0.6 = 0.861$ and $0.13 \cdot 0.4 + 0.87 \cdot 0.1 = 0.139$ to get the probabilities of transmitting during that next time step. We could repeat this with $t = 4, 5, 6, \ldots$, but this is cumbersome.

What is really interesting, however, is that the probabilities at each step can be represented as a two-dimensional vector. Because both entries are non-negative[1] and they sum up to one, we call these *stochastic vectors.* In this case, we started with $\mathbf{v}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, then $\mathbf{v}_1 = \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix}$, and then $\mathbf{v}_2 = \begin{pmatrix} 0.13 \\ 0.87 \end{pmatrix}$, and then $\mathbf{v}_3 = \begin{pmatrix} 0.139 \\ 0.861 \end{pmatrix}$. The first entry is the probability that the transmitter is transmitting, and the second is the probability that the sensor is idle.

For a general system where there are n states, a stochastic vector would be n-dimensional, where each entry is non-negative, and the sum of the entries is equal to one.

[1] x is positive if $x > 0$, while x is non-negative if $x \geq 0$. 0 is neither positive nor negative.

What is really nice, however, is that to get from one stochastic vector to the next, all we must do is multiply by a very particular matrix. For the above example, this 2×2 matrix would be $A = \begin{pmatrix} 0.4 & 0.1 \\ 0.6 & 0.9 \end{pmatrix}$. Notice that the matrix has the following probabilities:

1. $a_{1,1} = 0.4$, the probability of remaining in State 1 (transmitting) if the node is already transmitting,
2. $a_{2,1} = 0.6$, the probability of going idle (going to State 2) if the node is currently transmitting (State 1),
3. $a_{1,2} = 0.1$, the probability of beginning to transmit (going into State 1) if the node is currently idle, and
4. $a_{2,2} = 0.9$, the probability of the node remaining idle if the node is currently idle.

Given $\mathbf{v}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, we note $\mathbf{v}_1 = A\mathbf{v}_0$, $\mathbf{v}_2 = A\mathbf{v}_1$, and $\mathbf{v}_3 = A\mathbf{v}_2$, and in general, $\mathbf{v}_k = A\mathbf{v}_{k-1}$ or $\mathbf{v}_k = A^k\mathbf{v}_0$.

For a general system when n states, such a system can be described by an $n \times n$ matrix where column j stores the probabilities of transitioning from State j to State i. Thus, if there was a 30% chance that State 5 would transition to State 3, then $a_{3,5} = 0.3$. Such a matrix is said to be a "column-stochastic matrix". Such a matrix must have the following two properties:

1. All the entries must be greater than or equal to zero.
2. The sum of each column must equal one.

1.1 Why do we care?

In the example above, you might want to know: on average what percentage of the time is the node transmitting? This would let you know on average how much power will be needed, so if it is idle 90% of the time, you only need 50% of the power if it was only idle 80% of the time. One thing that is interesting, however, is that if you start with any initial stochastic vector $\mathbf{v}_0$, then as you keep iterating, $A^k\mathbf{v}_0$ always converges to the same vector. Whether or not the initial state is transmitting ($\mathbf{v}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$), idle ($\mathbf{v}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$), or there is a 50-50 chance that the node is transmitting ($\mathbf{v}_0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$), in the end, $A^k\mathbf{v}_0$ always converges to $\begin{pmatrix} 0.142857142857143 \\ 0.857142857142857 \end{pmatrix}$, meaning that, in the long run, this node is transmitting approximately $14\frac{1}{4}$% of the time and idle $85\frac{3}{4}$%.

In this project, we will investigate the relationship between this limiting vector and the properties of the matrix; those properties you learned about in your first-year course on linear algebra.

Remember that a matrix A has an eigenvalue λ if there is at least one non-zero vector $\mathbf{u}$ such that $A\mathbf{u} = \lambda\mathbf{u}$?

Thus, for the balance of this project, also remember that:

1. The state of a system is described by an n-dimensional stochastic vector, one that has the following two properties:
   (a) All the entries must be greater than or equal to zero.
   (b) The sum of the entries must equal one.
2. The transitions are described by an $n \times n$ column-stochastic matrix, one that has the following two properties:
   (a) All the entries must be greater than or equal to zero.
   (b) The sum of the entries of each column must equal one.

In this project, you will investigate some of the properties of column-stochastic matrices. You will use MATLAB or GNU Octave to generate column-stochastic matrices (to familiarize yourself with that tool), and then you will use interpolating polynomials and some of the issues when using floating-point numbers.

2 Project question

These will be submitted to Learn using Markdown (see the last section below). To generate a random column-stochastic vector, be sure to read up on the `rand(...)` and `sum(...)` functions and the element-wise division operator `./`, which are covered in the next section.

2.1 Question 1.

Recall that for a square $n \times n$ matrix A, a complex number λ is called an eigenvalue of A if there exists a nonzero vector $\mathbf{u} \in \mathbf{C}^n$ such that

$$A\mathbf{u} = \lambda\mathbf{u}.$$

In this case, $\mathbf{u}$ is called an eigenvector corresponding to λ.

Let A be a randomly generated stochastic $n \times n$ matrix, and let $\mathbf{1}_n$ denote the n-dimensional vector whose entries are all equal to one. All the information you require is in the following section describing the MATLAB programming language.

What do you observe?

1. Is $\mathbf{1}_n$ an eigenvector of A? That is, is $A\mathbf{1}_n$ a scalar multiple of $\mathbf{1}_n$? If yes, provide a proof; otherwise, provide a counter example.
2. Compute $A^\top\mathbf{1}_n$ numerically for several randomly generated $n \times n$ column-stochastic matrices A. Based on these observations, determine the relationship between the matrix $A^\top$ and the vector $\mathbf{1}_n$.
3. Interpret your result in terms of eigenvalues and eigenvectors. What eigenvalue does this imply must always be present for the transpose of a column-stochastic matrix?

Rubric

1. One mark for correctly deducing the answer and either providing a simple counter-example or having a complete proof that $\mathbf{1}_n$ is indeed an eigenvector of a column-stochastic matrix.
2. One mark for the correct observation.
3. One mark for correctly association the relationship between the column-stochastic matrix $A^\top$ and $\mathbf{1}_n$.

Markdown

The following can be used in Markdown: for $\mathbf{1}_n$, use `$\textbf{1}_n$`; and for $A^\top$, use `$A^\top$`.

2.2 Question 2.

Recall that if A with entries $a_{i,j}$ is an $n \times n$ column-stochastic matrix, then

$$\sum_{i=1}^{n} a_{i,j} = 1 \quad \text{for each } j = 1, \ldots, n.$$

The $(i,j)^{\text{th}}$ entry of $A^\top$ is $a_{j,i}$. Consequently, the sum of the entries in each row of $A^\top$ is equal to 1. Recall also that if you are calculating $\mathbf{v} = B\mathbf{u}$, then $v_i = \sum_{j=1}^{n} b_{i,j}u_j$. In this case, however, because you are multiplying $A^\top\mathbf{u}$, the i^{th} entry of the result will be calculated using $\sum_{j=1}^{n} a_{j,i}u_j$.

Prove the property you observed in Question 1. In particular, let A be an $n \times n$ column-stochastic matrix and let $\mathbf{1}_n$ be the vector of all ones. Show that

$$A^\top\mathbf{1}_n = \mathbf{1}_n,$$

and conclude that $\mathbf{1}_n$ is an eigenvector of $A^\top$. Hint: swap i and j in the last sum calculating the entries of the product $A^\top\mathbf{u}$.

Rubric

1. One mark for determining the summation that will be involved in calculating the k^{th} entry of the product $A^\top\mathbf{1}_n$.
2. One mark for showing that that the k^{th} entry of the product $A^\top\mathbf{1}_n$ must be 1 for $k = 1, \ldots, n$.

Markdown

The following can be used in Markdown: for $\sum_{i=1}^{n} a_{i,j} = 1$, use `$\sum_{i = 1}^n a_{i,j} = 1$`.

2.3 Question 3.

Create random square matrices A of various dimensions, and then calculate the eigenvalues of A, and the eigenvalues of $A^\top$. Is there a relationship between the eigenvalues of A and the eigenvalues of $A^\top$? Are the eigenvectors of A and $A^\top$ the same, or are they significantly different?

Rubric

1. One mark for determining whether there are zero or more eigenvalues that are common to to a random matrix A and its transpose $A^\top$.
2. One mark for determining whether there are zero or more eigenvectors that are common to a random matrix A and its transpose $A^\top$.

2.3.1 Matlab

Recall that if `A` is a variable assigned a matrix, then `A'` calculates the transpose of that matrix. While `eigs( A )` will print a vector of the eigenvalues of `A`, you need to use `[U, D] = eigs( A )` to get the orthogonal matrix of eigenvectors. See the discussion below for using the `eigs(...)` function.

2.4 Question 4.

All the entries of a column-stochastic matrix are real and non-negative. Thus answer the following questions based on your investigations:

1. Is it true that all the eigenvalues of a column-stochastic matrix are real? If it is true, provide a proof; otherwise, provide a counter example.
2. Is it true that all the eigenvalues of a column-stochastic matrix have a positive real component? If it is true, provide a proof; otherwise, provide a counter example.
3. Is there any other property about the absolute value of the eigenvalues of a column-stochastic matrix that you can deduce? You do not have to prove your result, but you should be able to make an observation about the absolute values.

For this question, you should create a number of column-stochastic matrices of various dimensions and find the eigenvalues.

Rubric

1. One mark for trying sufficiently many examples with large enough column-stochastic matrices (not just 2×2) to determine if the eigenvalues are always real, and if not, providing one counter-example.
2. One mark for trying sufficiently many examples with large enough column-stochastic matrices (not just 2×2) to determine if the eigenvalues always have a real component that is positive, and if not, providing one counter-example.
3. One mark for trying sufficiently many examples with large enough column-stochastic matrices (not just 2×2) to determine if there appears to be some upper bound on the absolute values of the eigenvalues.

2.5 Question 5.

In this question, you will prove one very specific case: you will show that a column-stochastic matrix cannot have an eigenvalue that is greater than 1. You will proceed as follows:

1. Assume that a column-stochastic matrix A has an eigenvalue $\lambda > 1$.
2. Use your result from Question 3 to find a corresponding eigenvalue for $A^\top$.
3. For the eigenvector $\mathbf{u}$ corresponding to the eigenvalue for $A^\top$, observe that this vector must have one entry u_k that is largest in absolute value, so in calculating $A^\top\mathbf{u}$, the corresponding entry will be λu_k.
4. Observe that calculating the entries of $A^\top\mathbf{u}$, each entry is a convex combination of the entries of $\mathbf{u}$. What did you learn about the upper and lower bounds of a convex combination of entries?

Note, with a little extra work, it is also possible to prove that a column-stochastic matrix cannot have a complex eigenvalue λ that has the property that $|\lambda| > 1$.

Rubric

1. One mark for correctly deducing an eigenvalue of $A^\top$ if A has an eigenvalue $\lambda > 1$.
2. One mark for correctly arguing that if the columns of A are all non-negative and sum to one, then the rows of $A^\top$ are all non-negative and sum to one, and so what does this mean about $A^\top\mathbf{u}$?
3. One mark for using the previous observation and the properties of a convex combination to determine if it is possible for the largest entry u_k of an eigenvector to equal λu_k with $\lambda > 1$.

Markdown

The following can be used in Markdown: for λ, use `$\lambda$`.

2.6 Question 6.

For this question, use `format long` to see all the digits in the sums.

Given a random $n \times n$ column-stochastic matrix A:

1. Create a random n-dimensional stochastic vector $\mathbf{u}$ and compute $A^k\mathbf{u}$ for increasing values of k, up to $k = 100$ (or larger if n is large). Does this sequence appear to converge? Give one example where $\|A^{k+1}\mathbf{u} - A^k\mathbf{u}\|_2$ is very small, meaning that the two are almost exactly equal by showing A, $\mathbf{u}$ and the two close vectors. You can use, for example, `norm( A^101*u - A^100*u )` for this calculation.
   Is $A^k\mathbf{u}$ still a stochastic vector for all k?

2. Let $\mathbf{v}$ be the stochastic vector with all entries equal to 0 except $v_1 = 1$. Compute $A^k\mathbf{v}$ for increasing values of k, again up to $k = 100$ (or larger if n is large). Does this sequence appear to converge? Give one example where $\|A^{k+1}\mathbf{u} - A^k\mathbf{u}\|_2$ is very small, meaning that the two are almost exactly equal. You can use, for example, `norm( A^101*v - A^100*v )` for this calculation.
   Is $A^k\mathbf{v}$ still a stochastic vector for all k?

3. For the same matrix A, compare the vectors to which $A^k\mathbf{u}$ and $A^k\mathbf{v}$ converge. What relationship do you observe between these two limits?

Rubric

1. One mark for using a value of $n \geq 4$.
2. One mark for determining if there is convergence.
3. One mark for the example where $A^{k+1}\mathbf{u}$ and $A^k\mathbf{u}$ are very close showing A, $\mathbf{u}$, $A^{k+1}\mathbf{u}$ and $A^k\mathbf{u}$ as output.
4. One mark for observing whether or not $A^k\mathbf{u}$ is still stochastic for a reasonable number of k, taking into account that numerical errors might mean that there may be some drift.
5. Three marks for repeating Items 2, 3 and 4 in this rubric but with an initial vector prescribed.
6. One mark for making the obvious observation about $A^k\mathbf{u}$ and $A^k\mathbf{v}$ when k is large.

2.7 Question 7.

What is the relationship between one of the eigenvectors of A and the vector to which $A^k\mathbf{u}$ converges? Use this to find the steady state of the system we described above, with a wireless sensor that is either idle or transmitting.

How might using the eigenvalues and eigenvectors of A be more convenient than explicitly computing $A^k\mathbf{u}$ for a given initial stochastic vector $\mathbf{u}$?

Rubric

1. Two marks for comparing the limit vector with the result of finding the eigenvalues and eigenvectors of the matrix A and making the correct deduction.
2. One mark for describing how this makes it easier to determine the limiting vector.

3 Using Matlab

In MATLAB or Octave (and any subsequent reference to "MATLAB" should be read as "MATLAB or Octave"), you can create a column vector of x-values using the following:

```
    v = [1.3 4.7 8.3 9.3]';    % or [1.3, 4.7, 8.3, 9.3]';
    u = [1.3; 4.7; 8.3; 9.3];
```

Leaving the apostrophe off creates a row vector. The `%` symbol is the comment symbol in Matlab, similarly to `//` in C++.

The semicolon at the end of the statement indicates that the output is to be suppressed: the identifier `v` will be assigned, but you will not see the output on the screen.

3.1 Complex and square matrix exponentiation

In MATLAB, the carat (ˆ) is used for exponentiation, for both complex numbers and square matrices:

```
format long
z = 0.5523413918 - 0.5832349158j;
zˆ15
    ans =  3.477236357066520e-02 + 1.378466862487123e-02i
z*z*z*z*z*z*z*z*z*z*z*z*z*z*z
    ans =  3.477236357066522e-02 + 1.378466862487124e-02i

A = [1.0541 0.8203; 0.5193 1.1387];
Aˆ15
    ans =
       2075.625289257045   2783.256860164245
       1761.971580498955   2362.670919355662
A*A*A*A*A*A*A*A*A*A*A*A*A*A*A
    ans =
       2075.625289257045   2783.256860164245
       1761.971580498953   2362.670919355660
```

3.2 Matrix constructors

To construct an $m \times n$ matrix of all zeros, use the `zeros` function:

```
A = zeros( 3, 5 )       % Construct a 3 x 5 matrix of zeros
    A =
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
```

While these display as integer zeros, they are stored as `double`. To construct an n-dimensional vector, set the other dimension to 1:

```
v = zeros( 3, 1 )       % Construct a 3-dimensional vector of zeros
    v =
        0
        0
        0
```

MATLABtreats n-dimensional vectors as $n \times 1$ matrices, but please don't think of vectors as matrices.

To construct an $m \times n$ matrix of all ones, use the `ones` function:

```
A = ones( 3, 5 )       % Construct a 3 x 5 matrix of ones:
    A =
        1     1     1     1     1
        1     1     1     1     1
        1     1     1     1     1
```

To construct an $m \times n$ matrix of random entries chosen from a uniform distribution from $(0, 1)$, ones, use the `rand` function:

```
A = rand( 3, 5 )
    A =
        0.442217   0.134473   0.886285   0.809402   0.013334
        0.488765   0.068059   0.117419   0.208350   0.463237
        0.533903   0.206501   0.260034   0.428336   0.262147
```

To construct an $n \times n$ identity matrix, you use the `eye` function:

```
A = eye( 3 )
    A =
        1     0     0
        0     1     0
        0     0     1
```

You can also explicitly define a matrix as follows in a similar way to vectors, but where entries in the same row are separated by either a space (or optionally a comma), and rows are separated by a semicolon:

```
A = [3.8 -4.7 5.2; 2.5 4.1 -1.7; 3.5 7.9 2.1]
     A =
        3.8000  -4.7000   5.2000
        2.5000   4.1000  -1.7000
        3.5000   7.9000   2.1000
```

3.3 Vector and matrix functions and operations

There are many functions that you can call to perform operations on a vector or a matrix. For example, `norm( v )` calculates the Euclidean norm of the vector `v`, and `rank( A )` calculates the rank of the matrix `A`, and `A'` calculates the transpose of that matrix:

```
A = [3.8 -4.7 5.2; 2.5 4.1 -1.7; 3.5 7.9 2.1]
    A =
       3.8000  -4.7000   5.2000
       2.5000   4.1000  -1.7000
       3.5000   7.9000   2.1000

A'          % Calculate the transpose of 'A'
    ans =

       3.8000   2.5000   3.5000
      -4.7000   4.1000   7.9000
       5.2000  -1.7000   2.1000

rank( A )
     ans = 3

B = [1 2 3; 4 5 6; 7 8 9];   % this matrix is not invertible
rank( B )
     ans = 2

v = [ 5 -6  8]';
norm( v )
     ans = 11.180
```

As this function works on the columns of a matrix, there are operators that also work on the entries of a vector or the columns or rows of a matrix. These are called element-wise operators, and they perform an operation on each entry of the matrix or vector. For example:

There are also operators that you can use: given two vectors `u` and `v` of the same dimension, you can calculate a linear combination of these vectors:

```
u = [ 1  3 -2]';
v = [ 5 -6  8]';
6.6*u - 4.7*v
    ans =
       -16.900
        48.000
       -50.800
```

Here we see an interesting feature of MATLAB: if you do not assign the output of a computation, it is by default assigned to the variable `ans`, which you can then use as an identifier in subsequent computations.

Given an $n \times n$ matrix A and an n-dimensional vector $\mathbf{u}$, you can multiply the vector by the matrix to calculate $A\mathbf{u}$:

```
A = [3.8 -4.7 5.2; 2.5 4.1 -1.7; 3.5 7.9 2.1];
u = [ 1  3 -2]';
```

```
w = A*u
     w =
       -20.700
        18.200
        23.000
```

Similarly, given an $n \times n$ matrix A and an n-dimensional vector $\mathbf{v}$, you can solve the system of linear equations defined by $A\mathbf{u} = \mathbf{v}$:

```
A = [3.8 -4.7 5.2; 2.5 4.1 -1.7; 3.5 7.9 2.1];
v = [ 5 -6  8]';
w = A \ v    % solve 'A*w = v'
     w =
       -1.8370
        0.9789
        3.1887
A*w          % show that 'w' is a solution by showing 'A*w' equals 'v':
     ans =
        5
       -6
        8
```

Given a square matrix A, you can find the eigenvalues or eigenvalues and eigenvectors of that matrix as follows:

```
A = rand( 3, 3 )
    A =
       0.968767   0.185109   0.026991
       0.351728   0.597209   0.407150
       0.530531   0.992824   0.562861

A = A + A'      % Make 'A' symmetric
    A =
       1.9375   0.5368   0.5575
       0.5368   1.1944   1.4000
       0.5575   1.4000   1.1257

e = eigs( A )
    e =
       3.0830
       1.4153
      -0.2406

[U, E] = eigs( A )
    U =
      -0.559728   0.828585   0.012275
      -0.590261  -0.409044   0.695898
      -0.581632  -0.382269  -0.718035

    E =
       3.0830        0        0
            0   1.4153        0
            0        0  -0.2406

U*E*U'
    ans =
       1.9375   0.5368   0.5575
       0.5368   1.1944   1.4000
       0.5575   1.4000   1.1257
```

The k^{th} columns of `U` is a unit eigenvector associated with the eigenvalue in the k^{th} entry along the diagonal of the matrix `E`.

You may recall that if A is a symmetric matrix, then all the eigenvalues of A are real and the eigenvectors can be made to form an orthogonal collection of eigenvectors. Also, given any matrix A, $A + A^\top$ is symmetric. The columns of U above are the eigenvectors associated with the corresponding eigenvectors along the diagonal entries of the matrix E, and because A is symmetric, if you normalize the eigenvectors of A (which they are done here), it follows that U must be orthogonal.

To multiply or divide the corresponding entries of two vectors of the same dimension, we must use the element-wise multiplication and division operators, and these are `.*` and `./`:

```
u = [5 -3 2]';
v = [2 10 4]';
u.*v
    ans =
       10
      -30
        8

u./v
    ans =
        2.5000
       -0.3000
        0.5000
```

If you divide or multiply a matrix by a scalar, each entry is multiplied or divided by that scalar:

```
A = [1 2 3; 4 5 6; 7 8 9];
2*A
    ans =
        2    4    6
        8   10   12
       14   16   18

A/5
    ans =
        0.2000   0.4000   0.6000
        0.8000   1.0000   1.2000
        1.4000   1.6000   1.8000
```

3.4 Column- and row-centric matrix functions and operations

The following functions behave as follows:

1. Given a row or column vector, or a $1 \times n$ or an $m \times 1$ matrix, they perform the given operations on the entries, and return a scalar.
2. Given an $m \times n$ matrix where $m \geq 2$ and $n \geq 2$, the operation is performed on the columns of the matrix.

```
% The sum of the entries:
sum( u )
     ans = 2
sum( A )
     ans =
            9.8000   7.3000   5.6000

% The product of the entries:
prod( u )
     ans = -6
prod( A )
     ans =
         33.250  -152.233   -18.564

% The minimum of the entries:
min( u )
     ans = -2
min( A )
     ans =
        2.5000  -4.7000  -1.7000

% The maximum of the entries:
max( u )
     ans = 3
max( A )
     ans =
        3.8000   7.9000   5.2000

% The average or mean of the entries:
mean( u )
     ans = 0.6667
mean( A )
     ans =
        3.2667   2.4333   1.8667

% The standard deviation of the entries:
std( u )
     ans = 2.5166
std( A )
     ans =
        0.6807   6.4632   3.4559

% The median of the entries:
median( u )
     ans = 1
median( A )
     ans =
        3.5000   4.1000   2.1000
```

If you want to multiply or divide each entry of each column of a matrix by a different number, you can use element-wise multiplication and division with a row vector:

```
A = [1 2 3; 4 5 6; 7 8 9];
A.*[-1 2 5]
     ans =
        -1    4   15
        -4   10   30
        -7   16   45

A./[2 -5 10]
     ans =
        0.5000  -0.4000   0.3000
        2.0000  -1.0000   0.6000
        3.5000  -1.6000   0.9000
```

If you want to multiply or divide each entry of each row of a matrix by a different number, you can use element-wise multiplication and division with a column vector:

```
A = [1 2 3; 4 5 6; 7 8 9];
A.*[-1 2 5]'
     ans =
        -1   -2   -3
         8   10   12
        35   40   45

A./[2 -5 10]'
     ans =
        0.5000   1.0000   1.5000
       -0.8000  -1.0000  -1.2000
        0.7000   0.8000   0.9000
```

3.5 Entry-wise functions

Given a function that normally takes a scalar as an argument, like `abs(...)` and `sin(...)`, and you pass that function a vector or matrix, that function is simply applied to each entry of the matrix:

```
A = rand( 3, 3 ) - 0.5     % subtract 0.5 from each entry of the matrix
    A =
       -0.117654  -0.451348  -0.134891
        0.392884  -0.200365   0.056349
        0.367103   0.118912  -0.067804

abs( A )                   % apply the absolute value function to each entry
    ans =
        0.117654   0.451348   0.134891
        0.392884   0.200365   0.056349
        0.367103   0.118912   0.067804
```

3.6 Display

You may recall that `double` uses 52 bits for the mantissa, so this translates to approximately 16 decimal digits of precision; however, Matlab only displays the output to five digits. If you want to see all 16 digits, use the display-format command `format`

```
A = [3.8 -4.7 5.2; 2.5 4.1 -1.7; 3.5 7.9 2.1];
v = [ 5 -6  8]';
w = A \ v    % solve 'A*w = v'
     w =
       -1.8370
        0.9789
        3.1887
format long
w = A \ v
     w =
       -1.837030011187315
        0.978890023833844
        3.188749452794397
format   % return to the default formating
w = A \ v
     w =
       -1.8370
        0.9789
        3.1887
```

4 Markdown by example

Markdown is a lightweight markup language that can be used to enter solutions into Crowdmark, and here are relevant features you can use:

~~~
Here is a paragraph with *italicized* text, **bold** text and `inline code`.

Leaving a blank line
indicates a new paragraph.
Consequently, all of this will appear as a single paragraph in your response.

If you want to show a block of matlab code, you can use:
```matlab
A = rand( 4, 4 );
A = A./sum(A);
[U, E] = eigs( A )
     U =
       -0.383727  -0.546609   0.040930   0.165486
       -0.504358  -0.166952  -0.047023  -0.282844
       -0.584729  -0.100801   0.708771  -0.606801
       -0.506428   0.814362  -0.702679   0.724159

     E =
        1.0000e+00           0           0           0
                 0 -2.1734e-01           0           0
                 0           0  1.0560e-01           0
                 0           0           0 -2.8649e-03
```
Here we have a table that summarizes approximations of $\sqrt{2}$ using the Babylonian
formula starting with an approximation $x_0 = 1$ (although, the use of the dollar sign will be
discussed below):

|Approximation | Absolute error | Relative error | Percent relative error | Significant digits |
|-|-|-|-|-|
|1 | 0.4142 | 0.2929 | 29.29% | 1 |
|1.5 | 0.08579 | 0.06066 | 6.066% | 1 |
|1.416666666 | 0.002453 | 0.001735 | 0.1735% | 3 |
|1.414215686 | 0.000002124 | 0.000001502 | 0.0001502% | 6 |

Observe that a table begins with a pipe and the |-|-|-|-| separates the header row from the table rows.
It is always important to put a blank line between structures such as tables above and lists below.

If you want a numbered list, just start numbering and indent by four characters if you want
a list inside a list:

1. Given an m x n matrix, set row = 1 and colunn = 1.
2. While row <= m and column <= n:
    1. If there are no leading non-zero entries
       in the current column, increment the column.
    2. Otherwise:
        - Find the row with the maximum leading non-zero entry in absolute value in that column and
          swap that row with the current row.
        - Next, add an appropriate multiples of the current row to
          each other row that has a leading non-zero
          entry in that column in order to zero out that leading non-zero entry.
        - Increment both the row and the column.
        - Return to Step 2.
~~~

This would be displayed as:

Here is a paragraph with *italicized* text, **bold** text and ‘inline code‘.

Leaving a blank line indicates a new paragraph. Consequently, all of this will appear as a single paragraph in your response.

If you want to show a block of matlab code, you can use:

```
A = rand( 4, 4 );
A = A./sum(A);
[E, D] = eigs( A )
     E =
       -0.383727  -0.546609   0.040930   0.165486
       -0.504358  -0.166952  -0.047023  -0.282844
       -0.584729  -0.100801   0.708771  -0.606801
       -0.506428   0.814362  -0.702679   0.724159

     D =
        1.0000e+00           0           0           0
                 0  -2.1734e-01           0           0
                 0           0   1.0560e-01           0
                 0           0           0  -2.8649e-03
```

Here we have a table that summarizes approximations of $\sqrt{2}$ using the Babylonian formula starting with an approximation $x_0 = 1$ (although, the use of the dollar sign will be discussed below):

Approximation	Absolute error	Relative error	Percent relative error	Significant digits
1	0.4142	0.2929	29.29%	1
1.5	0.08579	0.06066	6.066%	1
1.416666666	0.002453	0.001735	0.1735%	3
1.414215686	0.000002124	0.000001502	0.0001502%	6

Observe that a table begins with a pipe and the `|-|-|-|-|` separates the header row from the table rows. It is always important to put a blank line between structures such as tables above and lists below.

If you want a numbered list, just start numbering and indent by four characters if you want a list inside a list:

1. Given an m x n matrix, set row = 1 and column = 1.
2. While row <= m and column <= n:
    1. If there are no leading non-zero entries in the current column, increment the column.
    2. Otherwise:
        - Find the row with the maximum leading non-zero entry in absolute value in that column and swap that row with the current row.
        - Next, add an appropriate multiples of the current row to each other row that has a leading non-zero entry in that column in order to zero out that leading non-zero entry.
        - Increment both the row and the column.
        - Return to Step 2.

However, and finally, Markdown also supports $\LaTeX$:

```
Markdown is used to organize a document: it marks headings, paragraphs,
lists, tables, and blocks of code so that your answer is readable
and structured. \LaTeX mathematics is used to describe what mathematical
expressions mean, such as subscripts, fractions, summations,
and derivatives, so that formulas are displayed correctly and unambiguously.

In practical terms, Markdown tells the computer where things go in your
solution, while \LaTeX tells it how the mathematics itself is displayed.
A numerical analysis answer normally uses both: Markdown to organize
the explanation, and \LaTeX to write the equations.

If you want to display a mathematical equation inside a paragraph, just
surround the equation with dollar signs: $x + (y + z) = (x + y) + z$. \LaTeX
worries about how to format it. The caret operator indicates exponentiation,
so $(1 + x)^3 = 1 + 3x + 3x^2 + x^3$.

A target vector $\textbf{b}$ (bf = boldface) and a matrix $A$ may define
a system of linear equations $A\textbf{u} = \textbf{b}$. If you want a Greek
letter, you can use $\alpha$ or $\lambda$. Note that the backslash is used
to indicate that the following identifier should be interpreted by \LaTeX.
Thus, $\lambda$ is an eigenvector of the $n \times n$ square matrix $A$
if there exists a vector $\textbf{u} \in \textbf{R}^n$ such that
$A\textbf{u} = \lambda \textbf{u}$.

If $\textbf{u}$ is an $n$-dimensional vector, then its entries
are $u_1, u_2, \ldots, u_n$ (*three* low dots). If the subscript is more
substantial than one character, you must wrap the subscript in braces, so
the entries of an $m \times n$ matrix $A$ are $a_{i,j}$ where $i = 1, \ldots, m$
and $j = 1, \ldots, n$.

If you want to display a formula separate from the surrounding text, use two dollar
signs. For example, Newton's method starts with an initial approximation of the
root $x_0$ and subsequent approximations are calculated as follows:
$$ x_{n + 1} = x_n - \frac{ f(x_n) }{ f^\prime (x_n) }.$$
The \frac{ ... }{ ... } sets whatever is in the first set of braces as the
numerator of a fraction and whatever is in the second set of braces as the
denominator.

Here we define an inline $A\textbf{u} = \sum_{j = 0}^n a_{i,j} u_j$, and another that
is written as an equation:
$$ \textbf{u}\cdot \textbf{v} = \sum_{k = 0}^n u_k v_k.$$
```

This would be displayed as

> Finally, Markdown supports LaTeX:
>
> Markdown is used to organize a document: it marks headings, paragraphs, lists, tables, and blocks of code so that your answer is readable and structured. LaTeX mathematics is used to describe what mathematical expressions mean, such as subscripts, fractions, summations, and derivatives, so that formulas are displayed correctly and unambiguously.
>
> In practical terms, Markdown tells the computer where things go in your solution, while LaTeX tells it how the mathematics itself is displayed. A numerical analysis answer normally uses both: Markdown to organize the explanation, and LaTeX to write the equations.
>
> If you want to display a mathematical equation inside a paragraph, just surround the equation with dollar signs: $x + (y + z) = (x + y) + z$. LaTeX worries about how to format it. The caret operator indicates exponentiation, so $(1 + x)^3 = 1 + 3x + 3x^2 + x^3$.
>
> A target vector $\mathbf{b}$ (bf = boldface) and a matrix A may define a system of linear equations $A\mathbf{u} = \mathbf{b}$. If you want a Greek letter, you can use α or λ. Note that the backslash is used to indicate that the following identifier should be interpreted by LaTeX. Thus, λ is an eigenvector of the $n \times n$ square matrix A if there exists a vector $\mathbf{u} \in \mathbf{R}^n$ such that $A\mathbf{u} = \lambda\mathbf{u}$.
>
> If $\mathbf{u}$ is an n-dimensional vector, then its entries are $u_1, u_2, \ldots, u_n$ (*three* low dots). If the subscript is more substantial than one character, you must wrap the subscript in braces, so the entries of an $m \times n$ matrix A are $a_{i,j}$ where $i = 1, \ldots, m$ and $j = 1, \ldots, n$.
>
> If you want to display a formula separate from the surrounding text, use two dollar signs. For example, Newton's method starts with an initial approximation of the root x_0 and subsequent approximations are calculated as follows:
>
> $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$
>
> The `\frac{ ... }{ ... }` sets whatever is in the first set of braces as the numerator of a fraction and whatever is in the second set of braces as the denominator.
>
> Here we define an inline $A\mathbf{u} = \sum_{j=0}^{n} a_{i,j} u_j$ and another that is written as an equation:
>
> $$\mathbf{u} \cdot \mathbf{v} = \sum_{k=0}^{n} u_k v_k.$$

If you have issues with your Markdown or LaTeX, ask Microsoft's Co-pilot, OpenAI's ChatGPT, Google's Gemini, Elon Musk's Grok, or whichever other synthetic oracle you currently trust.