6.4 Systems of Linear Equations

Engineers love linear equations! They are everywhere in the systems that engineers design. Many (maybe most) systems are linear. Systems are sometimes described by differential equations, but these equation sets can usually be changed to linear equations by application of the Laplace transform. The main point though is that in real word design and analysis applications, we have *systems of equations* with multiple unknown variables, not just one equation.

In the *Linear System Applications* section, we will consider some examples of where systems of linear equations come from. For now, we will consider how to solve them.

6.4.1 An Example

The following system of equations have three unknown variables.

$$\begin{cases} 2x - 3y = 3 \\ 4x - 5y + z = 7 \\ 2x - y - 3z = 5 \end{cases}$$

The first step is to represent the equations in terms of matrices.

$$\begin{bmatrix} 2 & -3 & 0 \\ 4 & -5 & 1 \\ 2 & -1 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \\ 5 \end{bmatrix}$$

The common notation for describing this equation is $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, where the vector $\boldsymbol{x}$ represents the unknowns. In linear algebra notation, we describe the solution to the problem in terms of the *Calculating a Matrix Inverse*. Note that because matrix multiplication is not commutative, we have to be careful about the order of the terms.

$$\boldsymbol{A}^{-1}\,\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{x} = \boldsymbol{A}^{-1}\,\boldsymbol{b}$$

MATLAB, can solve the problem this way with the `inv` function, but MATLAB has a more efficient way to solve it.

6.4.2 Jumping Ahead to MATLAB

MATLAB's left-divide operator `\` solves systems of linear equations just as is accomplished by multiplying by the inverse of a matrix, but it is computationally more efficient.

```
>> A = [2 -3 0;4 -5 1;2 -1 -3];
>> b = [3 7 5]';
>> x = A\b
x =
```

```
3.0000
1.0000
0.0000
```

Thus, the values of the three unknown variables are: $x = 3$, $y = 1$, and $z = 0$.

6.4.3 Elimination

Gaussian elimination is a procedure for changing a matrix to a form where the solution is simple to calculate. A sequence of linear operations are applied to the rows of the matrix to produce an upper triangular matrix. Such a matrix has all zeros below the main diagonal.

Three operations are allowed in elimination.

1. Swap two rows – this step done first can reduce the number of additional elimination steps.
2. Add a multiple of one row to another row, replacing that row.
3. Multiply a row by a nonzero constant.

When a row has all zeros to the left of the main diagonal, that non-zero element on the diagonal is called the *pivot*. The pivot is used to determine a multiple of the row to add to another row to produce a needed zero.

We begin with our classic matrix equation $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$. Any operations done on the $\boldsymbol{A}$ matrix must also be applied to the $\boldsymbol{b}$ vector. After elimination, we have $\boldsymbol{U}\,\boldsymbol{x} = \boldsymbol{c}$, where $\boldsymbol{U}$ is an upper triangular matrix of the form:

$$\boldsymbol{U} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{bmatrix}$$

Note: One could continue the elimination steps until the $\boldsymbol{U}$ is an identity matrix, but it is faster to stop at an upper triangular matrix and use back substitution as illustrated in the example below.

To help carry forward the operations to the right side of the equation, we form an augmented matrix. Using the numbers from our previous example, we have the following. The pivots in each step are underlined.

$$\left[\begin{array}{ccc|c} \underline{2} & -3 & 0 & 3 \\ 4 & -5 & 1 & 7 \\ 2 & -1 & -3 & 5 \end{array}\right]$$

Add *-1 of row 1* to *row 3*.

$$\left[\begin{array}{ccc|c} \underline{2} & -3 & 0 & 3 \\ 4 & -5 & 1 & 7 \\ 0 & 2 & -3 & 2 \end{array}\right]$$

Add *-2 of row 1* to *row 2*. The pivot then moves to *row 2*.

$$\left[\begin{array}{ccc|c} 2 & -3 & 0 & 3 \\ 0 & \underline{1} & 1 & 1 \\ 0 & 2 & -3 & 2 \end{array}\right]$$

Add *-2 of row 2* to *row 3* to finish the row operations.

$$\left[\begin{array}{ccc|c} 2 & -3 & 0 & 3 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & -5 & 0 \end{array}\right]$$

Our matrix equation is now in the upper triangular form.

$$\begin{bmatrix} 2 & -3 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & -5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

The final step is now called back substitution where we start at the last row and work up to determine the values of the variables.

$$\begin{cases} -5z = 0 \\ y + 0 = 1 \\ 2x - 3 + 0 = 3 \end{cases}$$

$$x = 3, y = 1, z = 0$$

Practice Problem

Here is another system of equations with an integer solution. Use elimination to solve for variables x_1, x_2, x_3. Then use MATLAB to verify your answer.

$$\begin{cases} -3x_1 + 2x_2 - x_3 = -1 \\ 6x_1 - 6x_2 + 7x_3 = -7 \\ 3x_1 - 4x_2 + 4x_3 = -6 \end{cases}$$

6.4.4 Elimination to Find the Matrix Inverse

The Gauss–Jordan method uses elimination to calculate the inverse of a matrix. We start with an augmented matrix of the form $[\boldsymbol{A}|\boldsymbol{I}]$ and do row operations until we have $[\boldsymbol{I}|\boldsymbol{A}^{-1}]$.

$$\left[\begin{array}{ccc|ccc} \underline{2} & -3 & 0 & 1 & 0 & 0 \\ 4 & -5 & 1 & 0 & 1 & 0 \\ 2 & -1 & -3 & 0 & 0 & 1 \end{array}\right]$$

Add *-1 of row 1* to *row 3*.

$$\left[\begin{array}{ccc|ccc} \underline{2} & -3 & 0 & 1 & 0 & 0 \\ 4 & -5 & 1 & 0 & 1 & 0 \\ 0 & 2 & -3 & -1 & 0 & 1 \end{array}\right]$$

Add *-2 of row 1* to *row 2*. The pivot then moves to *row 2*.

$$\left[\begin{array}{ccc|ccc} 2 & -3 & 0 & 1 & 0 & 0 \\ 0 & \underline{1} & 1 & -2 & 1 & 0 \\ 0 & 2 & -3 & -1 & 0 & 1 \end{array}\right]$$

Add *-2 of row 2* to *row 3*.

$$\left[\begin{array}{ccc|ccc} 2 & -3 & 0 & 1 & 0 & 0 \\ 0 & \underline{1} & 1 & -2 & 1 & 0 \\ 0 & 0 & -5 & 3 & -2 & 1 \end{array}\right]$$

Add *3 of row 2* to *row 1*. The pivot then moves to *row 3*.

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 3 & -5 & 3 & 0 \\ 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 0 & \underline{-5} & 3 & -2 & 1 \end{array}\right]$$

Add *3/5 of row 3* to *row 1*; and add *1/5 of row 3* to *row 2*. Then a pivot is no longer needed.

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 0 & -16/5 & 9/5 & 3/5 \\ 0 & 1 & 0 & -7/5 & 3/5 & 1/5 \\ 0 & 0 & -5 & 3 & -2 & 1 \end{array}\right]$$

Divide *row 1 by 2*; and divide *row 3 by -5*.

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -16/10 & 9/10 & 3/10 \\ 0 & 1 & 0 & -7/5 & 3/5 & 1/5 \\ 0 & 0 & 1 & -3/5 & 2/5 & -1/5 \end{array}\right]$$

$$A^{-1} = \begin{bmatrix} -1.6 & 0.9 & 0.3 \\ -1.4 & 0.6 & 0.2 \\ -0.6 & 0.4 & -0.2 \end{bmatrix}$$

Note: This result matches what MATLAB found in the *Calculating a Matrix Inverse* section. All of the tedious row operations certainly makes one appreciate that MATLAB can perform the calculations for us.

6.4.5 The Row and Column View

A system of linear equations may be viewed from either the perspective of its rows or its columns. Both of these views can be plotted to give us a geometric understanding of systems of linear equations. The row view is what is most commonly known from algebra. It shows each row of the system as a line where the solution is the point where the lines intersect. The column view shows each column as a vector and presents the solution as a linear combination of the column vectors. The column view yields a very useful perspective that will be especially useful when we consider over-determined systems.

Let's illustrate the two views with a simple example.

$$\begin{cases} 2x + y = 4 \\ -x + y = 1 \end{cases}$$

$$\begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

Using elimination to find the solution:

$$\left[\begin{array}{cc|c} 2 & 1 & 4 \\ -1 & 1 & 1 \end{array}\right]$$

Add $1/2$ of row 1 to row 2:

$$\left[\begin{array}{cc|c} 2 & 1 & 4 \\ 0 & 3/2 & 3 \end{array}\right]$$

$$\begin{cases} \frac{3}{2}y = 3 \Rightarrow y{=}2 \\ 2x + 2 = 4 \Rightarrow x{=}1 \end{cases}$$

$$\begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

Row View Plot

$$\begin{cases} y = -2x + 4 \\ y = x + 1 \end{cases}$$

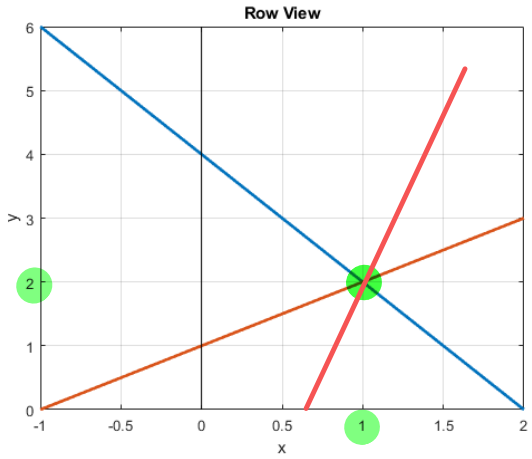

Fig. 6.7: In the row view of a system of equations, the solution is where line equations intersect.

If we had only one equation, or two equations that plot parallel lines, then we have the under-determined case where there is no intersection and no solution.

Additional equations could be added (over-determined). If the additional equations plot a line that intersects at the same point, then we still have an exact solution. If it does not intersect the other lines at the same point, then we can only approximate a solution, which we will cover in the *Over-determined Systems and Vector Projections* section.

Column View Plot

Let's view the system of equations as a linear combination of column vectors.

$$1 \times \begin{bmatrix} 2 \\ -1 \end{bmatrix} + 2 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

If we only had one equation (under-determined), then the set of possible end points of the vector is a line, not a point.

For a full rank matrix, there should always be a non-zero linear combination of the vectors that can reach any point on the plane except the origin, (0, 0). If vector $\boldsymbol{b}$ is non-zero, then it is always in the column space of $\boldsymbol{A}$ when $\boldsymbol{A}$ is full rank. The all-zero multipliers provides the only way to get to back to the origin for a full rank system. But a singular system with parallel vectors can have a non-zero linear combination leading back to the origin – this is called the *null solution*. (See *Null Space*)

If another row is added to the system making it over-determined, then the plot moves from $\mathbb{R}^2$ to $\mathbb{R}^3$. There can still be an exact solution as long the new row is in the same column space that was established by the first two rows. For example, we could add the equation $3x + 2y = 7$ to the system and $x = 1; y = 2$ is still valid. Thus we could still say that $\boldsymbol{b}$ is still in the column space of $\boldsymbol{A}$.

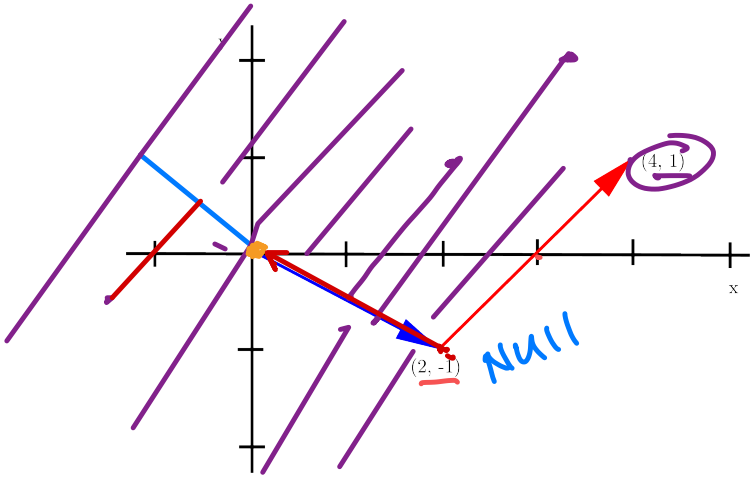

Fig. 6.8: The column view of a system of equations shows a linear combination of vectors. The solution is the scaling of each vector such that the head of the last vector is at the head of the vector on the right side of the equation.

In the Column Space of A

Consider the columns of matrix $\boldsymbol{A}$ as a set of vectors. The set of vectors that can be formed from linear combinations of this set of vectors is called the *span* of the columns of $\boldsymbol{A}$. A vector that is part of the span of the columns of $\boldsymbol{A}$ is said to be *in the column space* of $\boldsymbol{A}$.

6.4.6 When Does A Solution Exist?

Not every matrix equation of the form $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ has a unique solution. There must be an equal number of independent equations as there are unknown variables. A few tests can help to determine if a unique solution exists. The matrix in question here has m rows and n columns (m-by-n).

Note: Solutions to systems of equations of the form $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, may:

1. not exist
2. be an exact, unique solution
3. be an infinite set of vectors
4. be only an approximation

`rank`

The rank of a matrix is *the number of non-zero pivots during elimination*, which is also the number of independent equations. Rank is never more than the smaller dimension of the matrix. A square matrix with rank equal to the number of rows and columns is said to be *full rank*. Full rank matrices have all non-zero pivots.

When a row or column is a linear combination of others rows or columns, then a pivot will be zero and the rank is reduced.

MATLAB has a `rank` function which takes a matrix as input.

Solution Requirements for $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$

1. The $\boldsymbol{A}$ matrix is usually square ($m = n$).
2. The square matrix $\boldsymbol{A}$ must be full rank, $rank(\boldsymbol{A}) = m = n$. When $m > n$, then it is required that $rank(\boldsymbol{A}) = min(size(A))$.
3. The determinant of $\boldsymbol{A}$ may not equal zero. This is a well known test for $\boldsymbol{A}$ being invertible (not singular). It is also can be used as a solution test when the size of $\boldsymbol{A}$ is relatively small, but it is slow for larger matrices. The previous tests of rank are sufficient.

This example does not have a solution.

```
>> A = [1 2 3; 0 3 1; 1 14 7]
A =
     1     2     3
     0     3     1
     1    14     7

% Third row of A is a linear combination of rows 1 and 2
>> 4*A(2,:) + A(1,:)
ans =
     1    14     7

% A is not full rank
>> rank(A)
ans =
     2

% Not invertible
>> det(A)
ans =
     0

>> b = [1; 2; 3];

>> % This won't go well ...
>> A \ b
Warning: Matrix is singular to working precision.
ans =
     NaN
    -Inf
     Inf
```

Note: Based on the RREF of the augmented matrix and the application, a row or column might be removed and then the system solved as either an under-determined or over-determined system.

This example will go better. Notice that it is not necessary to calculate the determinant of $\boldsymbol{A}$. Sufficient condition is determined by the rank: $rank(\boldsymbol{A}) = rank([\boldsymbol{A}\ \boldsymbol{b}]) = m = n$. Note that the rank of the augmented matrix is only needed to determine if an exact solution existed for an over-determined matrix ($m > n$), but it also verifies that vector $\boldsymbol{b}$ is in the column space of a square matrix $\boldsymbol{A}$.

```
>> A = randi(10, 3, 3) - 4
A =
     5     6    -1
     6     3     2
    -2    -3     6
>> b = randi(10, 3, 1) - 2
b =
     8
     0
     8
>> rank(A)          % full rank
ans =
     3

% The next line is not needed because the matrix is square and full␣
 ↪rank.
>> rank([A b])      % b is in the column space of A.
ans =
     3
>> x = A \ b
x =
   -2.8889
    4.1481
    2.4444
```

Under-determined case:

When $m < n$, there are not enough equations and no unique solution exists. In this case, the $\boldsymbol{A}$ matrix is called *under-determined*. Although a unique solution does not exist, it is possible to find an infinite set of solutions, such as all of the points on a line or a plane. This will be discussed in *Under-determined Systems and RREF*.

Over-determined case:

When $m > n$, there are more equations than unknowns. In this case, the $\boldsymbol{A}$ matrix is called *over-determined*.

Pre-multiplying both $\boldsymbol{A}$ and $\boldsymbol{b}$ by $\boldsymbol{A^T}$ makes the left side of the equation a square matrix, which is needed for the elimination algorithms of either the inverse, `inv` , or left–divide, `\` ,

operations, i.e., $\boldsymbol{A}^T\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$. In fact, when $m > n$, the left–divide operator automatically pre-multiplies by $\boldsymbol{A}^T$ to make the matrix square.

In some cases, when $\boldsymbol{b}$ is in the column space of $\boldsymbol{A}$, then an exact solution exists. That is, $\boldsymbol{b}$ can be written as a linear combination of the columns of $\boldsymbol{A}$. You can test for this by verifying that $rank([\boldsymbol{A}\ \boldsymbol{b}]) = rank(\boldsymbol{A})$.

When $\boldsymbol{b}$ is *not* in the column space of $\boldsymbol{A}$, the only solution available is an approximation. We will discuss the over-determined case in more detail in *Over-determined Systems and Vector Projections*.

Note: Now complete *Matrix Homework*.

A
(4 x 3)
rank(A) = 3

[A b]
4 x 4
rank([A b])
= 3 - exact
= 4 - approximation