1 GSW… Gaussian Elimination

Gaussian elimination is probably the simplest technique for solving a set of simultaneous linear equations, such as:

$$\begin{aligned} y_1 &= A_{1,1}x_1 + A_{1,2}x_2 + A_{1,3}x_3 + \ldots + A_{1,n}x_n \\ y_2 &= A_{2,1}x_1 + A_{2,2}x_2 + A_{2,3}x_3 + \ldots + A_{2,n}x_n \\ &\ldots \\ y_m &= A_{m,1}x_1 + A_{m,2}x_2 + A_{m,3}x_3 + \ldots + A_{m,n}x_n \end{aligned} \tag{0.1}$$

more usually written using vectors to represent the terms y_m and x_n and a matrix to represent all the coefficients a_{ij}, as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \ldots \\ y_m \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & \ldots & A_{1,n} \\ A_{2,1} & A_{2,2} & \ldots & A_{2,n} \\ A_{3,1} & A_{3,2} & \ldots & A_{3,n} \\ \ldots & \ldots & \ldots & \ldots \\ A_{m,1} & A_{m,2} & \ldots & A_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} \tag{0.2}$$

or just as:

$$\mathbf{y} = \mathbf{A}\,\mathbf{x} \tag{0.3}$$

Gaussian elimination is in many ways the ‘brute force’ approach to solving systems of simultaneous linear equations. It is time-consuming, especially for large matrices, but the rules are simple to understand and easy to program, and it is reliable.

1.1 Elementary Row Operations

The technique of Gaussian elimination relies on the fact that there are certain operations that can be done on the set of equations that do not change the value of the unknown vector **x**. Consider the set of simultaneous equations:

$$\begin{aligned} 5 &= x_1 - 2x_2 \\ 4 &= 2x_1 - x_2 \end{aligned} \tag{0.4}$$

which can be represented in matrix form as:

$$\begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & -2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{0.5}$$

The operations that don’t change **x** are known as *elementary row operations*. These elementary row operations have to be applied to both the matrix **A** and the known vector **y** (in this case the vector [5; 4]) and consist of:

- Swapping any two rows
- Multiplying one row by a constant term
- Adding a linear multiple of one row to another row

1.1.1 Swapping Rows

Clearly, I can swap the order of the equations, and the component values x_1 and x_2 of **x** remain the same:

$$\begin{aligned} 4 &= 2x_1 - x_2 \\ 5 &= x_1 - 2x_2 \end{aligned} \tag{0.6}$$

That's obvious I hope. In terms of the matrix form of the equations, this is equivalent to swapping the elements of the corresponding rows of the matrix **A**, and the corresponding elements of the vector **y**, but not the elements of the vector **x**. In this case, we'd get:

$$\begin{bmatrix} 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{0.7}$$

1.1.2 Multiplying Rows by a Constant

I can also multiply any of the equations by any constant value, for example multiply the second equation by minus two, to get:

$$\begin{aligned} 4 &= 2x_1 - x_2 \\ -10 &= -2x_1 + 4x_2 \end{aligned} \tag{0.8}$$

or in matrix form:

$$\begin{bmatrix} 4 \\ -10 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{0.9}$$

and again, the values of the elements of **x** have not changed: multiplying an entire equation by minus two does not change the relationship between **x** and **y** in the equation. Note that again, I have to make the change to the row of **A** and the corresponding element of **y**, but the vector **x** remains unchanged.

1.1.3 Adding Rows Together

If $A = B$ and $C = D$ then clearly $A + C = B + D$. In the same way, I can add any two of the equations together, and replace one of the original equations with the sum. Suppose I added the first and second equations together, and replaced the second equation with the sum:

$$\begin{aligned} 4 &= 2x_1 - x_2 \\ -6 &= \quad\quad 3x_2 \end{aligned} \tag{0.10}$$

Again, in terms of the matrix representation, I've changed the elements of **A** and **y** into:

$$\begin{bmatrix} 4 \\ -6 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{0.11}$$

1.1.4 Hang On a Minute…

You might have just spotted that we now know the value of x_2. All we have to do is divide the new second equation by three, and we'll get $x_2 = -2$. Then, a simple substitution process into the remaining first equation produces $x_1 = 1$. We've solved for **x** without having to find the inverse matrix.

This process is called back-substitution (since it starts by finding the last element of **x** and then works backwards to the first), and it can be done whenever the matrix has an upper-triangular form (i.e. no non-zero elements below the leading diagonal). Another example: a four-by-four matrix in upper triangular form would in general look something like this:

$$\begin{bmatrix} U_{1,1} & U_{1,2} & U_{1,3} & U_{1,4} \\ 0 & U_{2,2} & U_{2,3} & U_{2,4} \\ 0 & 0 & U_{3,3} & U_{3,4} \\ 0 & 0 & 0 & U_{4,4} \end{bmatrix}$$

which corresponds to the equations:

$$\begin{aligned} y_1 &= U_{1,1}x_1 + U_{1,2}x_2 + U_{1,3}x_3 + U_{1,4}x_4 \\ y_2 &= \phantom{U_{1,1}x_1 + {}} U_{2,2}x_2 + U_{2,3}x_3 + U_{2,4}x_4 \\ y_3 &= \phantom{U_{1,1}x_1 + U_{1,2}x_2 + {}} U_{3,3}x_3 + U_{3,4}x_4 \\ y_4 &= \phantom{U_{1,1}x_1 + U_{1,2}x_2 + U_{1,3}x_3 + {}} U_{4,4}x_4 \end{aligned} \qquad (0.12)$$

With a set of equations in this form, it's easy to get the value of x_4 from the last equation, and once knowing that you can determine x_3 from the second-last equation, and then x_2, and so on.

This is pretty much what Gaussian elimination is all about: using these three elementary row operations to get the matrix **A** into this useful upper-triangular form, and then finding the value of **x** without having to calculate a matrix inverse at any stage. Of course, Gaussian elimination can be used for other things as well: it can find matrix inverses, determine the rank of a matrix, and perform matrix decomposition. There are a couple of tricks to know about along the way, but the basic ideas are pretty simple.

1.1.5 The Extended Matrix Form

You might have noticed that the elementary row operations could not just be done on the matrix **A**, they had to be done on the vector **y** as well. In practice, we often define an *extended* or *augmented matrix* form, which combines the matrix **A** and the vector **y** into one larger matrix, with one more column than **A**. For example, for a 3-by-3 matrix **A**, this would produce the extended matrix:

$$\left[\begin{array}{ccc|c} A_{1,1} & A_{1,2} & A_{1,3} & y1 \\ A_{2,1} & A_{2,2} & A_{2,3} & y2 \\ A_{3,1} & A_{3,2} & A_{3,3} & y3 \end{array}\right] \qquad (0.13)$$

Now the elementary matrix operations can be performed on the rows of this extended matrix, without having to worry about remembering to do them to both **A** (in unextended form) and **y**. This is easier to program.

1.1.6 Elementary Column Operations

One final point before I finish on elementary operations and get on with the algorithm: there are elementary column operations as well. You can swap any of the columns of the matrix **A**, but this time you have to swap over the corresponding elements in the rows of the vector **x**, and leave **y** alone. For example, we could write our original two equations (0.4) as:

$$\begin{aligned} 5 &= -2x_2 + x_1 \\ 4 &= -x_2 + 2x_1 \end{aligned} \tag{0.14}$$

which in matrix form would be:

$$\begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} \tag{0.15}$$

We've just swapped over the two columns in the matrix, and the rows of the elements in **x**. This is a special case of a more general result: take any two matrices **A** and **B**, and if you swap the *n*th and *m*th column of **A** and the *n*th and *m*th row of **B**, the product **AB** does not change[1].

1.2 The Basic Gaussian Elimination Algorithm

The idea of Gaussian elimination is to perform elementary row operations (mostly adding multiples of each equation to all of the equations below), in order to eliminate the entries in the matrix **A** below the leading diagonal. For example, take the set of three linear simultaneous equations:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.16}$$

In terms of how the algorithm is often programmed, the next step is to express the matrix **A** and vector **y** in terms the extended matrix form by adding the column vector **y** to the right of the matrix **A**:

$$\left[\begin{array}{ccc|c} A_{1,1} & A_{1,2} & A_{1,3} & y1 \\ A_{2,1} & A_{2,2} & A_{2,3} & y2 \\ A_{3,1} & A_{3,2} & A_{3,3} & y3 \end{array}\right] \tag{0.17}$$

(the vertical line after the third column in the extended matrix indicates where the original matrix **A** and column vector **y** join up; it has no other significance, and usually isn't included when we write the extended matrix).

Working in terms of this extended matrix, the first step is to remove any terms in x_1 from the second equation ($y_2 = A_{2,1}x_1 + A_{2,2}x_2 + A_{\ 2,3}x_3$). This can be done by subtracting $A_{2,1}/\ A_{1,1}$ times the first equation from the second equation, to leave the set of equations (written in this new extended matrix form):

[1] See the problems and solutions for a proof of this.

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & y_1 \\ 0 & A_{2,2} - A_{1,2}\left(A_{2,1}/A_{1,1}\right) & A_{2,3} - A_{1,3}\left(A_{2,1}/A_{1,1}\right) & y_2 - y_1\left(A_{2,1}/A_{1,1}\right) \\ A_{3,1} & A_{3,2} & A_{3,3} & y_3 \end{bmatrix} \tag{0.18}$$

Although this is easier to write if I let $b = A_{2,1}/A_{1,1}$.

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & y_1 \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} & y_2 - by_1 \\ A_{3,1} & A_{3,2} & A_{3,3} & y_3 \end{bmatrix} \tag{0.19}$$

Then, we remove any terms in x_1 from the third equation, by subtracting $c = A_{3,1}/A_{1,1}$ times the first equation from the third equation:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & y_1 \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} & y_2 - by_1 \\ 0 & A_{3,2} - cA_{1,2} & A_{3,3} - cA_{1,3} & y_3 - cy_1 \end{bmatrix} \tag{0.20}$$

Next, we subtract $d = (A_{3,2} - cA_{1,2}) / (A_{2,2} - bA_{1,2}) = (A_{3,2} - A_{1,2}(A_{3,1}/A_{1,1})) / (A_{2,2} - A_{1,2}(A_{2,1}/A_{1,1}))$ times the second equation from the third equation:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & y_1 \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} & y_2 - by_1 \\ 0 & 0 & A_{3,3} - cA_{1,3} - d\left(A_{2,3} - bA_{1,3}\right) & y_3 - cy_1 - d\left(y_2 - by_1\right) \end{bmatrix} \tag{0.21}$$

and the first three columns, which correspond to the manipulated matrix **A**, are now in upper triangular form. Writing this system of equations back into the familiar non-extended form here for clarity, gives:

$$\begin{bmatrix} y_1 \\ y_2 - by_1 \\ y_3 - cy_1 - d\left(y_2 - by_1\right) \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} \\ 0 & 0 & A_{3,3} - cA_{1,3} - d\left(A_{2,3} - bA_{1,3}\right) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.22}$$

These three simultaneous equations can be written:

$$\begin{aligned} x_1 &= \frac{1}{A_{1,1}}\left(y_1 - A_{1,2}x_2 - A_{1,3}x_3\right) \\ x_2 &= \frac{1}{A_{2,2} - bA_{1,2}}\left(y_2 - by_1 - A_{2,3} - bA_{1,3}x_3\right) \\ x_3 &= \frac{1}{A_{3,3} - cA_{1,3} - d\left(A_{2,3} - bA_{1,3}\right)}\left(y_3 - cy_1 - d\left(y_2 - by_1\right)\right) \end{aligned} \tag{0.23}$$

and these can be easily solved by *back-substitution* (that is, first finding the value of x_3 from the third equation, then substituting this value into the second equation to find the value of x_2, then substituting both x_2 and x_3 into the first equation to find x_1. We've now found the vector **x**, but avoided the need to explicitly invert the matrix. That's good: that saves time.

1.2.1 An Example of Gaussian Elimination

As an example of this process in action, consider the simultaneous equations:

$$\begin{aligned} 9 &= x_1 + 2x_2 - x_3 \\ -1 &= 2x_1 - x_2 + 2x_3 \\ 4 &= -x_1 + x_2 - 3x_3 \end{aligned} \tag{0.24}$$

in matrix form, these can be written:

$$\begin{bmatrix} 9 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.25}$$

and this gives the extended matrix form:

$$\begin{bmatrix} 1 & 2 & -1 & 9 \\ 2 & -1 & 2 & -1 \\ -1 & 1 & -3 & 4 \end{bmatrix} \tag{0.26}$$

Subtract twice the first equation (row) from the second equation (row):

$$\begin{bmatrix} 1 & 2 & -1 & 9 \\ 0 & -5 & 4 & -19 \\ -1 & 1 & -3 & 4 \end{bmatrix} \tag{0.27}$$

then add the first equation (row) to the third equation (row):

$$\begin{bmatrix} 1 & 2 & -1 & 9 \\ 0 & -5 & 4 & -19 \\ 0 & 3 & -4 & 13 \end{bmatrix} \tag{0.28}$$

then add 3/5 of the second equation (row) to the last equation (row):

$$\begin{bmatrix} 1 & 2 & -1 & 9 \\ 0 & -5 & 4 & -19 \\ 0 & 0 & -8/5 & 8/5 \end{bmatrix} \tag{0.29}$$

If we like, we can normalise the matrix by dividing each row by the leading element (the first non-zero element) in the row:

$$\begin{bmatrix} 1 & 2 & -1 & 9 \\ 0 & 1 & -4/5 & 19/5 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{0.30}$$

and writing this back in the more familiar form using a non-extended matrix, we get:

$$\begin{bmatrix} 9 \\ 19/5 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -4/5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.31}$$

from which can see that the original equations have been converted into the form:

$$\begin{aligned} 9 &= x_1 + 2x_2 - x_3 \\ \frac{19}{5} &= x_2 - \frac{4}{5}x_3 \\ -1 &= x_3 \end{aligned} \tag{0.32}$$

which are easy to solve sequentially, starting from the bottom and working up (*back-substitution*):

$$\begin{aligned} x_3 &= -1 \\ x_2 &= \frac{19}{5} + \frac{4}{5}x_3 = \frac{19}{5} - \frac{4}{5} = \frac{15}{5} = 3 \\ x_1 &= 9 - 2x_2 + x_3 = 9 - 6 - 1 = 2 \end{aligned} \tag{0.33}$$

The technique can easily be extended to any number of simultaneous equations, and typically[2] requires less than one-half of the number of operations required to invert the matrix.

1.3 Pivoting: Improving the Gaussian Elimination Algorithm

Having said that the Gaussian elimination technique is reliable, I should point out that there are some circumstances in which the simple algorithm described above just doesn't work. Fortunately, there is a simple modification to the algorithm that fixes the problem in most cases. In the general system of three equations described above, the problem is the term *d*.

In the third step of the algorithm, we had to subtract an amount $d = (A_{3,2} - cA_{1,2}) / (A_{2,2} - bA_{1,2}) = (A_{3,2} - A_{1,2}(A_{3,1}/A_{1,1})) / (A_{2,2} - A_{1,2}(A_{2,1}/A_{1,1}))$ times the second equation from the third equation. If $A_{2,2} - A_{1,2}(A_{2,1}/A_{1,1}) = 0$, then this is impossible. It requires a division by zero.

For example, consider the set of equations:

$$\begin{aligned} -1 &= x_1 + 3x_2 - x_3 \\ 7 &= 2x_1 + 6x_2 + x_3 \\ 6 &= x_1 + x_2 + 2x_3 \end{aligned} \tag{0.34}$$

Perform the first two steps of the Gaussian elimination technique to remove the coefficients of x_1 in the second and third equations, and we get:

[2] I have to say 'typically' since the exact cost of doing these calculations is dependent on the hardware being used to do the calculations. There's a lot of clever hardware around now that can speed up this sort of calculation, but only if the calculations are done in the way the hardware is expecting.

$$\begin{aligned}-1 &= x_1 + 3x_2 - x_3\\ 9 &= 0 + 0 + 3x_3\\ 7 &= 0 - 2x_2 + 3x_3\end{aligned} \tag{0.35}$$

and it's now impossible to subtract a multiple of the second equation from the third equation to eliminate the term $-2x_2$ from the third equation. The solution is obvious to us (although perhaps not to a computer, unless we program it to look out for these cases): all we need to do is swap the second and third equations:

$$\begin{aligned}-1 &= x_1 + 3x_2 - x_3\\ 7 &= 0 - 2x_2 + 3x_3\\ 9 &= 0 + 0 + 3x_3\end{aligned} \tag{0.36}$$

and without any further processing, the matrix is in upper triangular form, and we can carry on with the algorithm.

Even if $A_{2,2} - A_{1,2}(A_{2,1}/A_{1,1})$ was not quite zero but just very small, the simple version of the algorithm would require multiplying the second equation by a very large number before subtracting from the third equation. Any small amount of rounding error in the arithmetic caused by working with finite precision[3], or errors in the values of the terms in the matrix **A** or the vector **y** caused by noise, and this process could magnify this error, and result in a large error in the result for **x**. (Such sets of equations are known as 'ill-conditioned systems'.)

Again, the solution is to introduce another step into the Gaussian elimination algorithm: before eliminating the terms in x_n from the lowest $(N - n)$ equations, sort the lowest $(N - n + 1)$ equations so that the coefficient of x_n with the largest absolute value is on the n^{th} row (where N is the total number of equations).

This is exactly what we have just done in the case where $N = 3$ and $n = 2$: after the first elimination stage, the largest coefficient of x_2 in the lowest 2 equations was the –2 in the third equation: so we swapped this third equation with the second equation and proceed. This process is sometimes called *pivoting*[4].

1.4 Inverting Matrices Using Gaussian Elimination

The Gaussian elimination algorithm above successfully solves most systems of linear equations without inverting the matrix. If we really wanted to invert the matrix, we can do this by extending the algorithm.

Instead of starting with the equations in the form:

[3] For example with a limited number of bits of resolution in a fixed-point DSP chip, or an FPGA implementation.

[4] Technically, this process of exchanging rows so that the row with the largest element in the column ends up on the main diagonal is called *partial pivoting*. Full pivoting would allow the order of the columns to be changed as well, and can provide greater accuracy in the result when the accuracy of the calculations is limited. This however requires the order of the components of the unknown vector **x** to be changed too, and requires significantly more processing power.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.37}$$

we can multiply the left-hand side by the unit matrix before we start:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.38}$$

and then instead of modifying the terms in the vector **y** as we combine the equations, we change the terms in the unit matrix in the same way as the terms in the matrix **A**. For example, the first step in the Gaussian elimination becomes:

$$\begin{bmatrix} 1 & 0 & 0 \\ -b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.39}$$

which represents exactly the same set of equations as:

$$\begin{bmatrix} y_1 \\ y_2 - by_1 \\ y_3 \end{bmatrix} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.40}$$

in the format we were using before. We can write this using the extended matrix format as well, but this time we start with an extended matrix of:

$$\left[\begin{array}{ccc|ccc} A_{1,1} & A_{1,2} & A_{1,3} & 1 & 0 & 0 \\ A_{2,1} & A_{2,2} & A_{2,3} & 0 & 1 & 0 \\ A_{3,1} & A_{3,2} & A_{3,3} & 0 & 0 & 1 \end{array}\right] \tag{0.41}$$

which produces, after the first step, an extended matrix of:

$$\left[\begin{array}{ccc|ccc} A_{1,1} & A_{1,2} & A_{1,3} & 1 & 0 & 0 \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} & -b & 1 & 0 \\ A_{3,1} & A_{3,2} & A_{3,3} & 0 & 0 & 1 \end{array}\right] \tag{0.42}$$

and after the third step:

$$\left[\begin{array}{ccc|ccc} A_{1,1} & A_{1,2} & A_{1,3} & 1 & 0 & 0 \\ 0 & A_{2,2} - bA_{1,2} & A_{2,3} - bA_{1,3} & -b & 1 & 0 \\ 0 & 0 & A_{3,3} - cA_{1,3} - d\left(A_{2,3} - bA_{1,3}\right) & -c + bd & -d & 1 \end{array}\right] \tag{0.43}$$

Before, all we wanted to do was solve the simultaneous equations, so we stopped here. However, to find the inverse, we need to continue, and start working back up from the bottom, removing all the terms in the first three columns of this extended matrix (the part corresponding to the original matrix **A**) above the leading diagonal. The first step is to remove the term in $A_{1,3}$ from the first row by subtracting $e = A_{1,3}$ / $(A_{33} - cA_{1,3} - d(A_{2,3} - bA_{1,3}))$ times the third equation:

$$\left[\begin{array}{ccc|ccc} A_{1,1} & A_{1,2} & 0 & 1-e(bd-c) & de & -e \\ 0 & A_{2,2}-bA_{1,2} & A_{2,3}-bA_{1,3} & -b & 1 & 0 \\ 0 & 0 & A_{3,3}-cA_{1,3}-d\left(A_{2,3}-bA_{1,3}\right) & -c+bd & -d & 1 \end{array}\right] \quad (0.44)$$

and so on, until the first three columns on this extended matrix (which are the manipulated version of the original matrix **A**) are in diagonal form (i.e. they only have non-zero elements on the leading diagonal). Then divide each row by the value of its diagonal term, and we're left with an extended matrix of the form:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & w_{11} & w_{12} & w_{13} \\ 0 & 1 & 0 & w_{21} & w_{22} & w_{23} \\ 0 & 0 & 1 & w_{31} & w_{32} & w_{33} \end{array}\right] \quad (0.45)$$

which represents a set of simultaneous linear equations in the form:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{23} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (0.46)$$

and clearly, since **Wy** = **x** and **y** = **Ax**, the matrix **W** = $\mathbf{A}^{-1}$. This process is known as the *Gauss-Jordan* method.

I'll illustrate this with the same example as before. After the Gaussian elimination algorithm for the set of equations:

$$\begin{aligned} 9 &= x_1 + 2x_2 - x_3 \\ -1 &= 2x_1 - x_2 + 2x_3 \\ 4 &= -x_1 + x_2 - 3x_3 \end{aligned} \quad (0.47)$$

we got the series of equations (written in non-extended format this time):

$$\begin{bmatrix} 1 & 0 & 0 \\ 2/5 & -1/5 & 0 \\ 1/8 & -3/8 & -5/8 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -4/5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (0.48)$$

Now we continue by adding one times the third equation to the first equation to eliminate the term at the top-right of the matrix **A**:

$$\begin{bmatrix} 9/8 & -3/8 & -5/8 \\ 2/5 & -1/5 & 0 \\ 1/8 & -3/8 & -5/8 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & -4/5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (0.49)$$

then adding 4/5 of the third equation to the second equation:

$$\begin{bmatrix} 9/8 & -3/8 & -5/8 \\ 1/2 & -1/2 & -1/2 \\ 1/8 & -3/8 & -5/8 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.50}$$

and finally subtracting twice the second equation from the first equation:

$$\begin{bmatrix} 1/8 & 5/8 & 3/8 \\ 1/2 & -1/2 & -1/2 \\ 1/8 & -3/8 & -5/8 \end{bmatrix} \begin{bmatrix} 9 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.51}$$

(This corresponds to an extended matrix of:

$$\begin{bmatrix} 1 & 0 & 0 & 1/8 & 5/8 & 3/8 \\ 0 & 1 & 0 & 1/2 & -1/2 & -1/2 \\ 0 & 0 & 1 & 1/8 & -3/8 & -5/8 \end{bmatrix} \tag{0.52}$$

if we were using the extended notation.) So, we have our inverse:

$$\begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 1 & -3 \end{bmatrix}^{-1} = \frac{1}{8} \begin{bmatrix} 1 & 5 & 3 \\ 4 & -4 & -4 \\ 1 & -3 & -5 \end{bmatrix} \tag{0.53}$$

If we have to pivot, we just pivot the extended matrix: swapping over the rows of the extended matrix. If we end up having to divide by zero anyway, then this indicates that the matrix does not have an inverse.

1.5 Solving a Series of Systems of Simulations Equations

Quite often, we have the problem of solving a whole series of systems of simultaneous equations **y** =**Ax**, with the same value of **A**, but different values of **y**. This happens, for example, when we have a signal transmitted over a linear distorting channel, and we know the channel response (**A**) and the received signal (**y**), and we want to find out what the transmitted signal was (**x**). If the channel does not change, the same value of **A** is used over and over again with each new value of **y**.

We don't really want to have to do a full Gaussian elimination each time, for the new value of **y**. We could do a full inversion of the matrix and calculate $\mathbf{A}^{-1}$, but there is another way: a slight twist on the Gaussian elimination technique that makes solving the second, third and all subsequent sets of equations much easier. It starts off just like an attempt to invert the matrix, but it doesn't go the whole way.

The idea is to write the equations as:

$$\mathbf{Iy} = \mathbf{Ax} \tag{0.54}$$

where **I** is the unit matrix, and then instead of making changes in the value of **y** as we follow the process of Gaussian elimination, we make the changes to **I** instead. This is exactly the same idea as described above for calculating the inverse matrix.

For example, consider the same set of equations as we had before (the ones that required pivoting), only this time we want to keep the value of **y** as a variable, since it's going to take a different value for each set of simultaneous equations:

$$\begin{aligned} y_1 &= x_1 + 3x_2 - x_3 \\ y_2 &= 2x_1 + 6x_2 + x_3 \\ y_3 &= x_1 + x_2 + 2x_3 \end{aligned} \tag{0.55}$$

written in this new matrix format, this becomes:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -1 \\ 2 & 6 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.56}$$

The first step is to subtract twice the first equation from the second equation:

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -1 \\ 0 & 0 & -1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.57}$$

Next, we have to subtract the first equation from the second equation:

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -1 \\ 0 & 0 & -1 \\ 0 & -2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.58}$$

And now we see the need for the pivot. Again, we can do this by swapping the rows in the two matrices, rather than the rows in the right-hand matrix and the vector **y**, (this is just equivalent to writing the three simultaneous equations in a different order):

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 3 & -1 \\ 0 & -2 & 3 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{0.59}$$

Now the difference from the matrix inversion process: we stop here. We've transformed the original set of equations into the form:

$$\mathbf{By} = \mathbf{Ux} \tag{0.60}$$

where **U** is an upper-triangular matrix. For any new value of **y**, we can simply work out **By** since both are known, then determine **x** using the usual process of back-substitution. In the case where **A** is a sparse matrix (i.e. most terms are zero), this can be much faster than working out a full matrix inverse.

(There are more efficient ways to deal with series of sets of simultaneous equations, particularly when the matrix **A** has certain useful properties: see the chapter on Sets of Systems for more details on these techniques.)

1.6 Calculating Matrix Rank

Gaussian elimination can also be used to calculate the rank of a matrix. After converting the matrix into upper-triangular form, count how many non-zero elements there are on the leading diagonal. The result is the rank of the matrix.

Intuitively, this is reasonable: the rank of a matrix is the number of linearly independent rows in the matrix. The Gaussian elimination technique aims to simplify the matrix as much as possible by subtracting multiples of the rows of a matrix from each other. If any rows are linear combinations of the other rows, Gaussian elimination will remove them entirely.

For example, consider the matrix:

$$\begin{bmatrix} 1 & 2 & 3 & -1 \\ -2 & -4 & 1 & 0 \\ 1 & 2 & 3 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{0.61}$$

the first step in the Gaussian elimination is to add twice the first row to the second row:

$$\begin{bmatrix} 1 & 2 & 3 & -1 \\ 0 & 0 & 7 & -2 \\ 1 & 2 & 3 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{0.62}$$

then subtract the first row from the third row, and add the first row to the fourth row:

$$\begin{bmatrix} 1 & 2 & 3 & -1 \\ 0 & 0 & 7 & -2 \\ 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \end{bmatrix} \tag{0.63}$$

now we'll need to do a pivot, since the element $A_{2,2}$ is zero:

$$\begin{bmatrix} 1 & 2 & 3 & -1 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & -2 \end{bmatrix} \tag{0.64}$$

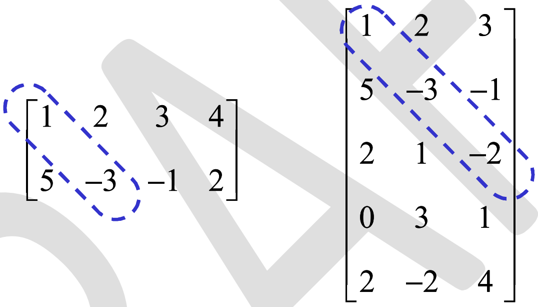
luckily, all the terms in the second column under $A_{2,2}$ are already zero, so we can move on to the third column. Once again, we need to do a pivot, since the term $A_{3,3}$ is zero but there are non-zero terms beneath it (the 7, in this case):

$$\begin{bmatrix} 1 & 2 & 3 & -1 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 7 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{0.65}$$

and now we stop: the matrix is in upper-triangular form. We can count the number of non-zero elements along the leading diagonal: in this case, three. Hence this matrix has rank three[5].

This process works with any matrices, not only square ones. Rectangular matrices have ranks too, and the rank can be determined from looking at the number of non-zero elements along the diagonal from the top-left down after a Gaussian elimination process. The only difference is that in the case of non-rectangular matrices, this doesn't lead to the element on the bottom-right.

This is why the rank of a matrix can never be greater than the lower of the number of rows and number of columns. There aren't any more elements on the leading diagonal.



$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & -3 & -1 & 2 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 5 & -3 & -1 \\ 2 & 1 & -2 \\ 0 & 3 & 1 \\ 2 & -2 & 4 \end{bmatrix}$$

Figure 1-1 The Leading Diagonal in Non-Square Matrices

1.6.1 Systems with Redundant Information

This method of calculating the rank also solves the problem of knowing which equations to ignore when there is redundant information. Just go through a process of Gaussian elimination, and then ignore all the equations with a zero element on the main diagonal.

For example, the case we had in the chapter on Linear Algebra was:

$$\begin{aligned} -1 &= x_1 + 2x_2 \\ 3 &= 2x_1 - x_2 \\ 1 &= -x_1 - 2x_2 \end{aligned} \tag{0.66}$$

Writing this in matrix format gives:

[5] Since this is a four-by-four matrix with rank three, it cannot be inverted. If you tried to calculate the inverse of this matrix, the next step would be to subtract a multiple of the fourth row to get rid of the term -1 in the top right: and that's impossible. Square matrices which do not have full rank (i.e. the rank is less than the number of rows and columns) do not have inverses.

$$\begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & -1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{0.67}$$

which gives the extended form of the matrix as:

$$\left[\begin{array}{cc|c} 1 & 2 & -1 \\ 2 & -1 & 3 \\ -1 & -2 & 1 \end{array}\right] \tag{0.68}$$

Go through the first steps of the Gaussian elimination algorithm on this matrix, and we get:

$$\left[\begin{array}{cc|c} 1 & 2 & -1 \\ 0 & -5 & 5 \\ 0 & 0 & 0 \end{array}\right] \tag{0.69}$$

The bottom row is all zeros. That means that there is no information in this third equation at all any more, it's just:

$$0 = 0x_1 + 0x_2 \tag{0.70}$$

Throw this row away. That leaves the square system, which is now already in upper-triangular form:

$$\left[\begin{array}{cc|c} 1 & 2 & -1 \\ 0 & -5 & 5 \end{array}\right] \tag{0.71}$$

and this can be readily solved using the back-substitution technique, which here gives:

$$\begin{aligned} -1 &= x_1 + 2x_2 \\ 5 &= \quad -5x_2 \end{aligned} \tag{0.72}$$

so:

$$\begin{aligned} x_2 &= -1 \\ x_1 &= -1 - 2x_2 = -1 + 2 = 1 \end{aligned} \tag{0.73}$$

1.7 Tutorial Questions

1) Prove that any elementary row operation on the extended matrix [**A** | **y**] does not change the values of **x** in the equation **y** = **Ax**.

2) Prove that any elementary row operation on the extended matrix [**A** | **I**] does not change the values of **x** in the equation **Iy** = **Ax**.

3) Prove that for any two matrices **A** and **B**, swapping any two columns of **A** and the same two rows of **B** does not change the product **AB** (provided the product exists).

4) Solve the simultaneous equations using the Gaussian elimination technique:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 - 2 &= 0 \\ x_1 + 2x_2 - x_3 + 2 &= 0 \\ x_1 - x_2 - 7x_3 + 2 &= 0 \end{aligned}$$

5) Can you solve the following set of simultaneous equations? If not, why not?

$$\begin{aligned} 1 &= x_1 + 2x_2 + 3x_3 \\ 3 &= x_1 - 2x_2 + x_3 \\ 0 &= x_1 + 6x_2 + 5x_3 \end{aligned}$$

6) Invert the matrix:

$$\begin{bmatrix} -1 & 7 & -3 \\ 2 & -2 & 0 \\ 0 & -3 & 3 \end{bmatrix}$$

7) Invert the matrix:

$$\begin{bmatrix} 1 & -1 & 1 \\ -2 & 2 & 0 \\ 0 & 2 & -2 \end{bmatrix}$$

and show that your answer is correct.

8) Solve the following set of simultaneous equations:

$$\begin{aligned} x_1 - x_2 + x_3 &= 1 \\ -x_1 + x_2 &= 2 \\ 3x_1 - x_2 - x_3 &= 3 \\ 2x_2 - 2x_3 &= 6 \end{aligned}$$

(Yes, there are four equations, but only three elements in **x**. Try a Gaussian elimination and see what happens.)

9) Find the rank of the following matrices:

$$\begin{bmatrix} 1 & 3 \\ 2 & -2 \\ 4 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 3 & 2 & 1 \\ 3 & -1 & -4 & 2 \\ -1 & 7 & 8 & 0 \end{bmatrix}$$

10) What's the rank of a vector?

11) Convert the following series of simulations equations into the form **By** = **Ux**, where **U** is an upper-triangular matrix:

$$\begin{bmatrix} 2 \\ -4 \\ -5 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 2 \\ 1 & -3 & -2 \\ -2 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$