10 Systems of linear equations and Gaussian elimination

10.1 Linear equations

An equation can have more than one **unknown**, as in

$$x^2 + y^2 = 5$$

An equation with one or more unknowns is called **linear** if the unknowns appear on their own, without exponent (that is to say, with exponent 1). In other words, products of unknowns are not allowed. The example above is not linear (but "quadratic") because it contains the product of x with itself and also the product of y with itself. The following *is* a linear equation (with three unknowns)

$$3x - 2y + z = 7$$

A general linear equation with n unknowns can be written as

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$$

where it is understood that the x's are the unknowns, the a's and b are **placeholders** or **parameters** or **coefficients** which in any concrete instance would be actual numbers.

A linear equation which contains more than one variable will usually have many solutions. For example, the equation

$$2x - y = 0$$

has the solutions $x = 1, y = 2$, and $x = 2, y = 4$, and $x = 3, y = 6$, and so on. In order to pin this down to *exactly one solution* we need *as many equations as there are unknowns*. For example, the system of linear equations

$$\begin{aligned} 2x - y &= 0 \\ x + y &= 6 \end{aligned}$$

has only *one* assignment for x and y which makes *both* equations *simultaneously* true, namely, $x = 2, y = 4$.

10.2 Gaussian elimination

In the language of Computer Science, Gaussian elimination is best described as a *recursive algorithm*. It has a **base case** and a **general case**:

Base Case There is only one equation and one unknown: $ax = b$. This can be solved directly: $x = b/a$.

General Case There are n equations and each contains n unknowns: Then we use the first equation to **eliminate** the first unknown in the remaining $n-1$ equations. Those new $n-1$ equations then only contain $n-1$ unknowns and Gaussian elimination can be applied recursively to those.

Once Gaussian elimination returns the values for those $n-1$ unknowns, they can be substituted into the first equation in which then only one unknown remains. This can be solved with the Base Case.

Time for an example:

92

$$\begin{aligned} x_1 + 5x_2 - 2x_3 &= -11 && \text{(Eq. 1)} \\ 3x_1 - 2x_2 + 7x_3 &= 5 && \text{(Eq. 2)} \\ -2x_1 - x_2 - x_3 &= 0 && \text{(Eq. 3)} \end{aligned}$$

This is the General Case as there is more than one unknown. We use equation 1 to eliminate x_1 from equations 2 and 3. We do this by subtracting 3 times equation 1 from equation 2, and adding twice equation 1 to equation 3:

93

Now we apply Gaussian elimination recursively to the smaller system consisting of equations 2 and 3 which each contain two unknowns, and temporarily forget about equation 1. Once more we have to follow the General Case: we use the second equation to eliminate x_2 in the third. For this we add 9 times equation 2 to 17 times equation 3:

94

We get one equation (the third one) with *one* unknown. It can be solved by the Base Case: $x_3 = -32/32 = -1$.

Now we go back one step in the recursion and substitute $x_3 = -1$ in the second equation and get:

95

The second equation can now be solved by the Base Case and we get $x_2 = (38+13)/(-17) = -3$.

We return to the very first step in the algorithm and replace x_2 and x_3 in the first equation by the values that we have computed:

96

Simplifying the first equation gives $x_1 = -11-2+15 = 2$. All three unknowns have been computed; the algorithm stops.

10.3 A more compact notation

The unknowns themselves are not affected by Gaussian elimination as all computation happens on the numeric coefficients in front of them. This suggest to only write out the coefficients and leave the unknowns implicit. In our example, the picture looks like this

97

$$\left(\begin{array}{rrr|r} 1 & 5 & -2 & -11 \\ 3 & -2 & 7 & 5 \\ -2 & -1 & -1 & 0 \end{array}\right)$$

We include the vertical line to remind us that the last column refers to the right hand sides of the given equations, not to another unknown.

Rather than go through the same example again, we do Gaussian elimination on a second example in this new notation.

$$\left(\begin{array}{rrr|r} 3 & 1 & 5 & 3 \\ -3 & 1 & -2 & -5 \\ 3 & -1 & 7 & 10 \end{array}\right)$$

Eliminating the first variable in equations 2 and 3:

98

Eliminating the second variable in equation 3:

99

Simplifying the last equation:

100

We can now read off that $x_3 = 1$ and as described above, use this value in the first two equations, etc. Alternatively, we can continue working in the compact notation: Using the last line we eliminate the entries in the third column in lines one and two.

101

Now we use the second line and use it to eliminate the second entry in the first row.

102

We finish by dividing the first line by 6 and the second line by 2 so as to get all ones on the diagonal:

103

We can read off the solution by putting back the unknowns:

104

$$\begin{pmatrix} x_1 & & & = & 1/6 \\ & x_2 & & = & -5/2 \\ & & x_3 & = & 1 \end{pmatrix}$$

Summary and useful hints

- Every unknown corresponds to precisely one column of entries in the compact notation. The last column corresponds to the numbers on the right-hand side of the equations.
- Be careful with the signs of the coefficients when translating, and when copying from one step in the algorithm to the next.
- Always work with the compact notation, not the explicit equations containing variables or you will inevitably get confused.
- Neat work will help you avoid common mistakes, such as dropping a minus sign and mis-copying entries.
- Avoid fractions until the very end. For example, in

  $$\left(\begin{array}{cc|c} 2 & -3 & 5 \\ 3 & 7 & -1 \end{array}\right)$$

  don't rewrite to

  $$\left(\begin{array}{cc|c} 2 & -3 & 5 \\ 0 & 7-\frac{3}{2}\times(-3) & -1-\frac{3}{2}\times 5 \end{array}\right) = \left(\begin{array}{cc|c} 2 & -3 & 5 \\ 0 & \frac{23}{2} & -\frac{17}{2} \end{array}\right)$$

  by subtracting $\frac{3}{2}$ of the first line, but first multiply suitably as in

  $$\left(\begin{array}{cc|c} 2 & -3 & 5 \\ 6 & 14 & -2 \end{array}\right)$$

  and then subtract an *integer multiple* (here 3 times the first line):

  $$\left(\begin{array}{cc|c} 2 & -3 & 5 \\ 0 & 1-3\times(-3) & -2-3\times 5 \end{array}\right) = \left(\begin{array}{cc|c} 2 & -3 & 5 \\ 0 & 23 & -17 \end{array}\right)$$

- Know your sign rules:
  - subtracting a negative number from a negative number:

    $$-x-(-y) \text{ is the same as } -x+y \text{ which is the same as } y-x$$

  - multiplying negative numbers:

    $$(-x)\times(-y) \text{ is the same as } x\times y$$

- Don't even think about using the highschool "substitution method" if there are more than two equations!

- Despite all the good advice I am trying to give, it's still very easy to make mistakes in Gaussian elimination, so when you have worked out a solution *check* it by substituting the values back into the equations. It takes very little time.
- Abstracting away from the actual values, the first phase of Gaussian elimination leads to a regular even-spaced "staircase" that separates the zeros created by the algorithm from the rest:

$$\left(\begin{array}{cccc|c} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right) \longrightarrow \cdots \longrightarrow \left(\begin{array}{cccc|c} \bullet & * & * & * & * \\ 0 & \bullet & * & * & * \\ 0 & 0 & \bullet & * & * \\ 0 & 0 & 0 & \bullet & * \end{array}\right)$$

The term often used in books for this pattern is **echelon form**. In the second phase, one creates zeros above the staircase. This can be pictured as follows:

$$\left(\begin{array}{cccc|c} \bullet & * & * & * & * \\ 0 & \bullet & * & * & * \\ 0 & 0 & \bullet & * & * \\ 0 & 0 & 0 & \bullet & * \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} \bullet & * & * & 0 & * \\ 0 & \bullet & * & 0 & * \\ 0 & 0 & \bullet & 0 & * \\ 0 & 0 & 0 & 1 & * \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} \bullet & * & 0 & 0 & * \\ 0 & \bullet & 0 & 0 & * \\ 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 1 & * \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & * \\ 0 & 1 & 0 & 0 & * \\ 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 1 & * \end{array}\right)$$

10.4 The special cases

The presentation of Gaussian elimination as a recursive algorithm makes it very easy to analyse the special cases. All special cases are to do with zeros appearing in the "wrong place". We discuss each in turn and give an example. The last special case will force us to generalise the Gaussian elimination algorithm.

Special Base Case 1: contradictory equation. The base case concerns one equation with one unknown: $ax = b$. It can happen that $a = 0$, so the equation really reads $0 = b$. If, furthermore, it is the case that $b \neq 0$, then the equation is *contradictory* and can not be solved. In this case, the algorithm should exit and indicate that a solution cannot be found, for example, by raising an exception. If the original system had several equations, and the base case was reached through a series of recursive calls, then the overall result is still that there is **no solution**. Example:

105

$$\begin{array}{rcrcl} x_1 & - & 2x_2 & = & 3 \\ 2x_1 & - & 4x_2 & = & 7 \end{array} \longrightarrow$$

Special Base Case 2: irrelevant equation. This is like the previous case but in fact both sides are zero, that is, in $ax = b$ both a and b are zero and the equation reduces to $0 = 0$. This means that *the equation does not constrain the value of x in any way*. The answer of the algorithm should therefore be that this unknown **can be chosen freely**. Example:

106

$$\begin{array}{rcrcl} x_1 & - & 2x_2 & = & 3 \\ 2x_1 & - & 4x_2 & = & 6 \end{array} \longrightarrow$$

In this example the second equation is redundant (it is exactly twice the first equation), so in actual fact we are left with only one equation for two unknowns. We can choose x_2 freely to be any number and once we have done this, we have to set x_1 to $3+2x_2$. We write this as

$$\begin{array}{rcl} x_1 & = & 3+2x_2 \\ x_2 & : & \text{chosen freely} \end{array}$$

Special Recursive Case 1: can't use the first equation for eliminating the first unknown. If the coefficient of the first unknown in the first equation is zero, then no matter how often the first equation is added to another one, the first variable of that other one is not affected.

In this case, we simply exchange the first equation with another one for which the first coefficient is *not* zero, and run the algorithm on this rearranged system. (In fact, it is useful to "pick" the most suitable equation for elimination anyway. If we do computation on paper, then we would like the relevant coefficient to be close to 1; if it is done on a computer, then one picks the one with the *largest* lead coefficient because this keeps the round-off errors down.) Example:

107

$$\left(\begin{array}{ccc|c} 0 & -2 & 3 & 3 \\ 2 & -1 & -2 & 6 \\ -5 & -2 & -1 & -3 \end{array}\right) \quad \longrightarrow$$

Special Recursive Case 2: Can't use any of the given equations to eliminate the first unknown. This is like "Special Base Case 2", in that it means that the first variable is not constrained at all by the given system. The algorithm should report back that the first variable can be chosen freely. Example:

108

$$\left(\begin{array}{ccc|c} 1 & -2 & 1 & 3 \\ 2 & -4 & -2 & 7 \\ -5 & 10 & -1 & -9 \end{array}\right) \quad \longrightarrow$$

However, the example also shows that this brings us to a situation where we have *two* equations for *one* unknown — a case we have not had to consider before. So this special case forces us to consider *more general* systems of linear equations, where the number of unknowns does not necessarily match the number of equations.

General Gaussian elimination Luckily, the algorithm does not need much adjustment. What we have called the "Recursive Case" or "General Case" stays the same and we only have to reconsider the Base Case. This now splits into two:

Base Case 1 Only one unknown is left over but there may be more than one equation. In this case we solve each equation independently and compare the answers; if they agree, then that is what we return. If they disagree, then the system has no solution (which we can communicate back by raising an exception).

Base Case 2 Only one equation is left over but more than one unknown appears in it (let's say m many). In this case $m-1$ unknowns can be chosen freely and the other one is computed from those choices and the equation. (It does not matter which $m-1$ unknowns are chosen and which is computed.)

The second Base Case should be illustrated with an example. Suppose we end up in a situation where only this equation remains

$$x-2y+z=3$$

The answer to return to the previous level of the recursion should be:

109

Generalising Gaussian elimination in this way does not affect the strategies we follow in the special cases discussed in this chapter. We can, however, be more precise about what to do in "Special Recursive Case 2" above. In this case we remember that the first variable is not constrained and run the algorithm recursively on the n equations for only $n-1$ unknowns. When we get an answer from the recursive call we report this back up and also report that the first variable can be chosen freely.

Special cases in the compact notation. The above analysis is attractive as it explains exactly what needs to be done in each special case; in fact, we could use it to implement Gaussian elimination on a computer. For working on paper, however, a global geometric description is more helpful.

Remember that without the special cases, Gaussian elimination leads to a regular even-spaced "staircase":

$$\left(\begin{array}{cccc|c} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right) \quad \longrightarrow \quad \cdots \quad \longrightarrow \quad \left(\begin{array}{cccc|c} \bullet & * & * & * & * \\ 0 & \bullet & * & * & * \\ 0 & 0 & \bullet & * & * \\ 0 & 0 & 0 & \bullet & * \end{array}\right)$$

To have no special case occur during the elimination is the same as saying that the entries represented as bullets are different from zero. The special cases have the effect that some steps are wider than others (but the height of the steps is still at most one); we get the *echelon form* for general systems:

$$\left(\begin{array}{ccccc|c} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{array}\right) \quad \longrightarrow \quad \cdots \quad \longrightarrow \quad \left(\begin{array}{ccccc|c} 0 & \bullet & * & * & * & * \\ 0 & 0 & 0 & \bullet & * & * \\ 0 & 0 & 0 & 0 & \bullet & * \\ 0 & 0 & 0 & 0 & 0 & ? \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

As before, the bullets indicate entries different from zero, and the asterisks arbitrary entries. If the question mark is nonzero, then there is no solution. If it is zero, then we have infinitely many solutions because one or several variables can be chosen freely. The ones that can be chosen freely are those where the staircase does not drop down one level (x_1 and x_3 in the example). The other ones are computed but their value depends on what was chosen for the free ones.

To give a numeric example, the above final echelon form could have the entries

$$\left(\begin{array}{ccccc|c} 0 & 2 & -3 & 7 & 0 & 4 \\ 0 & 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & 0 & 0 & -3 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

We record the result of the computation as

$$\begin{array}{lcl} x_1 & : & \text{chosen freely} \\ x_2 & = & 2+\frac{3}{2}x_3 \\ x_3 & : & \text{chosen freely} \\ x_4 & = & 0 \\ x_5 & = & -2 \end{array}$$

Extended Gaussian elimination. Once we have reached echelon form, and once we have checked that the system is not contradictory, we can continue to eliminate entries *above* the pivots (i.e., the bullets in our diagrams). In the examples above, for example, we could carry on in the following way (assuming the question mark is a zero):

$$\left(\begin{array}{ccccc|c} 0 & \bullet & * & * & * & * \\ 0 & 0 & 0 & \bullet & * & * \\ 0 & 0 & 0 & 0 & \bullet & * \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \quad \longrightarrow \quad \cdots \quad \longrightarrow \quad \left(\begin{array}{ccccc|c} 0 & 1 & * & 0 & 0 & * \\ 0 & 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

and this is of help for writing down the general solution. In the numerical example we get:

$$\left(\begin{array}{ccccc|c} 0 & 2 & -3 & 7 & 0 & 4 \\ 0 & 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & 0 & 0 & -3 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \quad \longrightarrow \quad \left(\begin{array}{ccccc|c} 0 & 1 & -\frac{3}{2} & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

Other methods. In school you may have learned that besides the elimination method, there is also the *substitution method* for solving systems of linear equations. In fact, it is not really different, but actually just another way of writing down the steps of the algorithm. However, the chance of making an arithmetic error with the substitution method is *much greater* than with Gaussian elimination, so I do not recommend it!

You may also have learned about determinants and *Cramer's rule* which gives the solutions as the quotient of two determinants. Computationally, this is a *bad idea* for three reasons: Cramer's rule cannot cope with special cases, and it requires many more arithmetic operations than Gaussian elimination, and those operations are numerically unstable.

Finally, I must admit that Gaussian elimination is also not ideal from a computational point of view. When we have a system with many variables (say, more than 2,000) then round-off errors tend to accumulate leading to completely wrong "solutions". Instead, people use *iterative* methods which approximate the solution vector step by step. A discussion of these is beyond the scope of this module.

Systems of linear equations over other fields. If we reflect on the work we have done in this chapter then we may notice that we only used the *four basic arithmetic operations*, addition, subtraction, multiplication and (in the very last step of Gaussian elimination) division. From this we can conclude that Gaussian elimination can be employed for systems of equations *over an arbitrary field*, for example, the finite field $\mathrm{GF}(2)$ discussed previously.

Exercises

For each of the following systems of linear equations, transcribe into the compact notation and solve by Gaussian elimination:

1. $$\begin{array}{rcrcrcr} x_1 & + & 2x_2 & + & x_3 & = & -1 \\ 2x_1 & - & x_2 & + & x_3 & = & 1 \\ -2x_1 & + & x_2 & - & 2x_3 & = & 2 \end{array}$$

2. $$\begin{array}{rcrcrcr} & & x_1 & - & 2x_2 & = & 0 \\ 2x_1 & - & 2x_2 & - & 3x_3 & = & 0 \\ 4x_2 & - & 3x_3 & - & 4x_4 & = & 0 \\ 6x_3 & - & 4x_4 & - & 5x_5 & = & 0 \\ & & 8x_4 & - & 5x_5 & = & 1 \end{array}$$

3. $$\begin{array}{rcrcrcr} 2x_1 & - & x_2 & - & x_3 & = & 3 \\ x_1 & + & 2x_2 & + & 2x_3 & = & -1 \\ 3x_1 & + & x_2 & + & x_3 & = & 3 \end{array}$$

4. $$\begin{array}{rcrcrcr} -2x_1 & + & 2x_2 & - & x_3 & = & 4 \\ 3x_1 & + & 2x_2 & + & 2x_3 & = & -1 \\ -x_1 & - & 4x_2 & - & x_3 & = & -3 \end{array}$$

5. The following system of equations is to be read as being over $\mathrm{GF}(2)$. Compute *all* solutions.

$$\begin{array}{rcl} x_1+x_2+x_3+x_4+x_5 & = & 1 \\ x_1+x_3+x_5 & = & 1 \\ x_1+x_4 & = & 1 \end{array}$$

Practical advice

In the exam, I expect you to be able to solve a general system of linear equations, whether it is over the ordinary numbers or over the finite field $\mathrm{GF}(2)$. In particular, you must be able to

- translate from the explicit notation with unknowns (as in Box 92) to the compact notation introduced in Section 10.3, Box 97, and back;
- recognise when a system is contradictory and has no solution;
- recognise when there are infinitely many solutions and clearly write down which variables can be freely chosen and which have to be computed and how. If the field is $\mathrm{GF}(2)$, then there are only *two choices* for the freely chosen variables, so we can actually list them all.

Please remember...

- Always work with the compact notation and transform the number scheme until the general staircase pattern (the echelon form) has been achieved.
- Always report the result in the form outlined above at the end of Section 10.4.