LINEAR ALGEBRA: BEWARE!

MATH 196, SECTION 57 (VIPUL NAIK)

You might be expecting linear algebra to be a lot like your calculus classes at the University. This is probably true in terms of the course structure and format. But it's not true at the level of subject matter. Some important differences are below.

- Superficially, linear algebra is a lot easier, since it relies mostly on arithmetic rather than algebra. The computational procedures involve the systematic and correct application of processes for adding, subtracting, multiplying, and dividing numbers. But the key word here is *superficially*.
- Even for the apparently straightforward computational exercises, it turns out that people are able to do them a lot better if they understand what's going on. In fact, in past test questions, people have often made fewer errors when doing the problem using full-scale algebraic symbol manipulation rather than the synthetic arithmetic method.
- One important difference between linear algebra and calculus is that with calculus, it's relatively easy to understand ideas *partially*. One can obtain much of the basic intuition of calculus by understanding graphs of functions. In fact, limit, continuity, differentaaition, and integration all have basic descriptions in terms of the graph. Note: these aren't fully rigorous, which is why you had to take a year's worth of calculus class to cement your understanding of the ideas. But it's a *start*. With linear algebra, there is no single compelling visual tool that connects all the ideas, and conscious effort is needed even for a partial understanding.
- While linear algebra lacks any *single* compelling visual tool, it requires *either* considerable visuo-spatial skill *or* considerable abstract symbolic and verbal skill (or a suitable linear combination thereof). Note the gap here: the standard computational procedures require only arithmetic. But getting an understanding requires formidable visuo-spatial and/or symbolic manipulation skill. So one can become a maestro at manipulating matrices without understanding anything about the meaning or purpose thereof.
- Finally, even if you master linear algebra, the connection of linear algebra to its applications is relatively harder to grasp than the connection of calculus to its applications. Applying multivariable calculus is easy: *marginal rates of change equal (partial) derivatives*. Summing up over a continuously variable parameter equals integration. One can get quite far with just these two ideas. With linear algebra, on the other hand, there is no punchline. There's no easy way of spotting when and how a given situation in economics or computer science or any other branch of mathematics requires the use of linear algebra.

Keeping all the above in mind, you can treat this course in either of two ways. You can look at it as a bunch of arithmetic procedures (with the occasional algebraic insight behind a procedure). From that perspective, it's a "grind"-course: relatively low-skilled, but requires a lot of gruntwork and concentration. Or, you can look at it as a bunch of difficult ideas to be mastered, with a few arithmetic procedures to codify the execution of these ideas. From that perspective, it's an intellectually challenging course, and if you succeed, a satisfying one.

In practice, I would recommend seeing the course as a mix. Make sure you master the basic computational procedures. I'll try to keep them to the bare minimum you need to attain familiarity with the structures involved. The homeworks (particularly the routine homeworks) will serve that role. And if I fall short on the explanations in class (I hope I don't), the book will fill the gap.

And give a decent shot to trying to understand the concepts. I can't guarantee success. In fact, as mentioned above, you're much more likely to take away zero conceptual knowledge from a linear algebra course than you are from a calculus course. But at least if you try, you have *some* chance. Give it a shot. The quizzes will help you with that.

LINEAR FUNCTIONS: A PRIMER

MATH 196, SECTION 57 (VIPUL NAIK)

Executive summary

Words ...

(1) Linear models arise both because some natural and social phenomena are intrinsically linear, and because they are computationally tractable and hence desirable as approximations, either before or after logarithmic and related transformations.
(2) Linear functions (in the affine linear sense) can be characterized as functions for which all the second-order partial derivatives are zero. The second-order pure partial derivatives being zero signifies linearity in each variable holding the others constant (if this condition is true for each variable separately, we say the function is (affine) multilinear). The mixed partial derivatives being zero signifies *additive separability* in the relevant variables. If this is true for every pair of input variables, the function is completely additively separable in the variables.
(3) We can use logarithmic transformations to study multiplicatively separable functions using additively separable functions. For a few specific functional forms, we can make them linear as well.
(4) We can use the linear paradigm in the study of additively separable functions where the components are known in advance up to scalar multiples.
(5) If a function type is linear in the parameters (not necessarily in the input variables) we can use (input,output) pairs to obtain a system of linear equations in the parameters and determine the values of the parameters. Note that a linear function of the variables with no restrictions on the coefficients and intercepts must also be linear in the parameters (with the number of parameters being one more than the number of variables). However, there are many nonlinear functional forms, such as polynomials, that are linear in the parameters but not in the variables.
(6) Continuing the preceding point, the number of well-chosen (input,output) pairs that we need should be equal to the number of parameters. Here, the "well-chosen" signifies the absence of dependencies between the chosen inputs. However, choosing the bare minimum number does not provide any independent confirmation of our model. To obtain independent confirmation, we should collect additional (input,output) pairs. The possibility of modeling and measurement errors may require us to introduce error-tolerance into our model, but that is beyond the scope of the current discussion. We will return to it later.

Actions ...

(1) One major stumbling block for people is in writing the general functional form for a model that correctly includes parameters to describe the various degrees of freedom. Writing the correct functional form is half the battle. It's important to have a theoretically well-grounded choice of functional form and to make sure that the functional form as described algebraically correctly describes what we have in mind.
(2) It's particularly important to make sure to include a parameter for the *intercept* (or *constant term*) unless theoretical considerations require this to be zero.
(3) When dealing with polynomials in multiple variables, it is important to make sure that we have accounted for all possible monomials.
(4) When dealing with piecewise functional descriptions, we have separate functions for each piece interval. We have to determine the generic functional form for each piece. The total number of parameters is the sum of the number of parameters used for each of the functional forms. In particular, if the nature of the functional form is the same for each piece, the total number of parameters is (number of parameters for the functional form for each piece) $\times$ (number of pieces).

1. A brief summary that reveals nothing

In the natural and social sciences, "linearity" is a key idea that pops up repeatedly in two ways:

(1) A lot of natural and social phenomena are intrinsically linear, i.e., the mathematical formulations of these involve linear functions.
(2) A lot of natural and social phenomena allow for linear approximation. Even though the actual description of the phenomenon is nonlinear, the linear approximation serves well for descriptive and analytic purposes.

Understanding how linear structures behave is thus critical to understanding many phenomena in the natural and social sciences, and more to the point, critical to understanding the mathematical models that have (rightly or wrongly) been used to describe these phenomena.

2. Additively separable, multilinear, and linear

2.1. Terminology: linear and affine.

A function f of one variable x is termed linear if it is of the form $x \mapsto mx + c$ for real numbers m (the slope) and c (the intercept). In later usage, we will sometimes restrict the term "linear" to situations where the intercept is zero. To make clear that we are taking of the more general version of linear that allows a nonzero intercept, it is better to use the term "affine" or "(affine) linear." The choice of word may become clearer later in the course. In contrast, "homogeneous linear" explicitly signifies the absence of a constant term.

2.2. The case of functions of two variables.

Consider a function F of two variables x and y.

We say that F is:

- *additively separable* if we can find functions f and g such that $F(x, y) = f(x) + g(y)$. Under suitable connectedness and differentiability assumptions, this is equivalent to assuming that F_{xy} is identically zero. The differentiability assumption is necessary to make sense of F_{xy}, and the connectivity assumption is necessary in order to argue the converse (i.e., going from $F_{xy} = 0$ to F being additively separable).

  Conceptually, additive separability means that the variables do not "interact" in the function. Each variable produces its own contribution, and the contributions are then pooled together. In the polynomial setting, this would mean that there are no polynomials that are mixed products of powers of x with powers of y.
- *(affine) linear in x* if, for each fixed value $y = y_0$, the function $x \mapsto F(x, y_0)$ is a linear function of x. Under reasonable connectivity assumptions, this is equivalent to assuming that F_{xx} is the zero function.

  What this means is that once we fix y, the function has constant returns in x. The graph of the function restricted to $y = y_0$ (so it is now just a function of x) looks like a straight line.
- *(affine) linear in y* if, for each fixed value $x = x_0$, the function $y \mapsto F(x_0, y)$ is a linear function of y. Under reasonable connectivity assumptions, this is equivalent to assuming that F_{yy} is the zero function.

  What this means is that once we fix x, the function has constant returns in y. The graph of the function restricted to $x = x_0$ (so it is now just a function of y) looks like a straight line.
- *(affine) multilinear* if it is linear in x and linear in y.

  This means that the function has constant returns in each variable individually. However, it does not mean that the function has constant returns in the variables together, because the returns to one variable may be influenced by the value of the other.
- *(affine) linear* if it is both (affine) multilinear and additively separable. Under connectivity and differentiability assumptions, this is equivalent to saying that *all* the second-order partial derivatives are zero.

  Additive separability basically guarantees that the returns on one variable are not affected by the other, and therefore, the function must have constant returns in the variables combined.

  F must be of the form $F(x, y) = ax + by + c$ with a, b, c real numbers. The graph of such a function is a plane. The values a and b are the "slopes" in the respective variables x and y and the value c is the intercept. If we are making the graph $z = F(x, y)$, the intersection of the graph (a plane) with the z-axis gives the value of the intercept.

Here are some examples. Note that when we say "linear" below we mean affine linear:

Function	Additively separable?	Linear in x?	Linear in y	Multilinear?	Linear?
$x^2 - \sin y + 5$	Yes	No	No	No	No
xe^y	No	Yes	No	No	No
$(x+1)(y-1)$	No	Yes	Yes	Yes	No
$\cos x + y - 1$	Yes	No	Yes	No	No
$2x + 3y - 4$	Yes	Yes	Yes	Yes	Yes

2.3. **Extending to multiple variables.** A function F on n variables $x_1, x_2, \dots, x_n$ is termed affine linear if there exist real numbers $a_1, a_2, \dots, a_n, c$ such that we can write:

$$F(x_1, x_2, \dots, x_n) := a_1x_1 + a_2x_2 + \dots + a_nx_n + c$$

The case $c = 0$ is of particular interest because it means that the origin gets sent to zero. We shall talk in more detail about the significance of that condition later.

Once again, linear functions of multiple variables can be broken down in terms of the following key attributes:

- Additive separability in every pair of variables, and in fact, additive separability "overall" in all the variables. This is equivalent (under various connectedness and differentiability assumptions) to the second-order mixed partial derivative in every pair of variables being zero.
- Multilinearity: For each variable, the function is (affine) linear with respect to that variable, holding other variables constant. This is equivalent (under various connectedness and differentiability assumptions) to the second-order pure partial derivative in every variable being zero.

We could have functions that are additively separable and not multilinear. This means that the function is a sum of separate functions of the individual variables, but the separate functions are not all linear. We could also have functions that are multilinear but not additively separable. For instance, $x_1x_2 + x_2x_3 + x_1x_2x_3$ is (affine) multilinear but not additively separable.

2.4. **Real-world importance.** Both additive separability and multilinearity are simplifying assumptions with real-world significance. Suppose we are trying to model a real-world production process that depends on the labor inputs of two individuals. Let x and y be the number of hours each individual devotes to the production process. The output is a function $F(x, y)$.

The additive separability assumption would mean that the inputs of the two workers do not interact, i.e., the hours put in by one worker do not affect the marginal product per hour of the other worker. The multilinearity assumption says that holding the hours of the other worker constant, the output enjoys constant returns to hours for each worker. If we assume both of these, then F is linear, and we can try to model it as:

$$F(x, y) = ax + by + c$$

with a, b, c as real numbers to be determined through empirical observation. c in this situation represents the amount produced if neither individual puts in any work, and we may take this to be 0. a and b represent the "productivity" values of the two respective workers.

Note that both additive separability and the multilinearity are pretty strong assumptions that are not usually satisfied. In the production function context:

- The two alternatives to additive separability (no interaction) are *complementarity* (positive second-order mixed partial) and *substitution* (negative second-order mixed partial).
- The two alternatives to linearity (constant returns) in a particular variable are *increasing returns* (positive second-order pure partial) and *decreasing returns* (negative second-order pure partial).

3. Situations reducible to the linear situation

3.1. **Multiplicatively separable functions.** If linearity requires such strong assumptions in even simple contexts, can we really expect it to be that ubiquitous? Not in a literal sense. However, some very specific nonlinear functional forms can be changed to linear functional forms by means of the logarithmic transformation.

We say that a function $G(x, y)$ of two variables x and y is *multiplicatively separable* if $G(x, y) = f(x)g(y)$ for suitable functions f and g. Assuming that G, f, and g are positive on our domain of interest, this can be rewritten as:

$$\ln(G(x, y)) = \ln(f(x)) + \ln(g(y))$$

Viewed *logarithmically*, therefore, multiplicatively separable becomes additively separable. Of course, as noted above, additive separability is not good enough for linearity.

Of particular interest are functions of the form below, where c, x, and y are positive:

$$G(x, y) = cx^a y^b$$

Note that *Cobb-Douglas production functions* are of this form.

Taking logarithms, we get:

$$\ln(G(x, y)) = \ln c + a \ln x + b \ln y$$

Thus, $\ln(G)$ is a linear function of $\ln x$ and $\ln y$. Taking logarithms on everything, we thus get into the linear world.

3.2. **Additively separable situation where the component functions are known in advance up to scalar multiples.** Suppose $f_1, f_2, \dots, f_n$ are known functions, and we are interested in studying the possibilities for a function F of n variables of the form:

$$F(x_1, x_2, \dots, x_n) = a_1 f_1(x_1) + a_2 f_2(x_2) + \dots + a_n f_n(x_n) + c$$

The key is that $f_1, f_2, \dots, f_n$ are known in advance. In this situation, we can "replace" $x_1, x_2, \dots, x_n$ by $f_1(x_1), f_2(x_2), \dots, f_n(x_n)$, treating the latter as the new variables. With these as the new inputs, the output is a linear function, and all the techniques of linearity apply.

4. The computational attraction of linearity

The discussion so far has concentrated on why linearity is a powerful conceptual assumption, and also on why that assumption may or may not be justified. This also gives us some insight into why linear functions might be computationally easier to handle. It's now time to look more explicitly at the computational side.

The introduction to your textbook (Bretscher) mentions the *streetlight strategy*: a person lost his keys, and was hunting for them under the streetlight, not because he thought that was the most likely location he might have dropped them, but because the light was good there, so he had a high chance of finding the keys if indeed the keys were there. Although this is often told in the form of a joke, the streetlight strategy is a reasonable strategy and is not completely baseless. Linear algebra is a very powerful streetlight that illuminates linear functions very well. This makes it very tempting to try to use linear models. To an extent, the best way to deal with this temptation is to yield to it. But not completely. Naive "linear extrapolation" can have tragic consequences.

I'd love to say more about this, but that's what your whole course is devoted to. I will briefly allude to some key aspects of linearity and its significance.

4.1. **Parameter determination.** So, you've decided on a particular model for a phenomenon. Perhaps, you've decided that the amount produced is a linear function of the hour counts x and y of the form:

$$F(x, y) = ax + by + c$$

The problem, though, is that you don't yet know the values of a, b, and c, the *parameters* in the model. The model is too theoretical to shed light on these values.

In order to use the model for numerically accurate predictions, you need to figure out the values of the parameters. How might you do this? Let's think more clearly. The simplest rendering of a function is:

$$\text{Input variables} \overset{\text{the function}}{\to} \text{Output variables}$$

Now, however, if only a *generic form* for the function is known, and the values of some parameters are not yet known, then the "machine" needs to be fed with both the input variables and the parameter values. We can think of this as:

$$\text{Input variables} + \text{Parameter values} \overset{\text{generic form}}{\to} \text{Output variables}$$

Now, suppose we know the values of the outputs for some specific choices of inputs. Each such piece of information gives an equation in terms of the parameter values. With enough such equations, we hope to have enough information to deduce the parameter values.

As a general rule, keep in mind that:

Dimension of solution space to constraint set = (Dimension of original space) - (Number of independent constraints)

In other words:

Dimension of solution space = (Number of variables) - (Number of equations)

In our case, we are trying to solve equations in the parameters, and we want a "zero-dimensional" solution space, i.e., we want to determine the parameter uniquely. Thus, what we want is that:

0 = (Number of parameters) - (Number of input-output pairs)

Note here that the parameters become the variables that we are trying to find, but they are different from what we usually think of as the variables, namely the inputs to the function. This could be a source of considerable confusion for people, so it's important to carefully understand this fact.

Note also something about the jargon of *input-output pair*: a single input-output pair includes the values of each of the inputs, plus the output. If the function has n inputs and 1 output, an "input-output pair" would be a specification of $n + 1$ values, including n inputs and 1 output.

Once we have found the values of the parameters, we treat them as those constant values and no longer treat them as variables. Our treating them as variables is a provisonal step in finding them.

In other words, in order to determine the parameters, we need to have as many input-output pairs as the number of parameters.

However, if we have exactly as many input-output pairs as the number of parameters, we will get the parameters but we have no additional confirmation, no sanity check, that our model is indeed correct. In order to provide additional confirmation, it will be necessary to have *more* input-output pairs than the number of parameters that need to be determined. If this "overdetermined" system still gives unique solutions, then indeed this provides some confirmation that our model was correct.

A number of issues are not being addressed here. The most important is the issue of modeling errors and measurement errors. The former refers to the situation where the model is an approximation and not exactly correct, while the latter refers to a situation where the input-output pairs are not measured with full accuracy and precision. Usually, both errors are present, and therefore, we will not expect to get exact solutions to overdetermined systems. We need to adopt some concept of error-tolerance and provide an appropriate mathematical formalism for it. Unfortunately, that task requires a lot more work, so we shall set it aside for now.

Instead, let us return to the question of parameter determination, assuming that everything works exactly. Consider our model:

$$F(x, y) = ax + by + c$$

This model has two inputs (x and y) and one output ($F(x, y)$), and it has three parameters a, b, and c. Specifying an input-output pair is equivalent to specifying the output $F(x_0, y_0)$ for a particular input $x = x_0, y = y_0$.

Right now, our goal is to determine the parameters a, b, and c. What we need are specific observations of input-output pairs. Suppose we have the following observations:

$$\begin{aligned} F(1,2) &= 12 \\ F(2,3) &= 17 \\ F(3,5) &= 25 \end{aligned}$$

These observations may have been gathered empirically: these might have been the number of hours put in by the two workers, with the corresponding outputs, in past runs of the production process.

We now plug in these input-output pairs and get a system of linear equations:

$$\begin{aligned} a + 2b + c &= 12 \\ 2a + 3b + c &= 17 \\ 3a + 5b + c &= 25 \end{aligned}$$

Note that this is a system of linear equations *in terms of the parameters*, i.e., the variables for this system of linear equations are the parameters of our original functional form.

If we solve the system, we will get the unique solution $a = 2$, $b = 3$, $c = 4$.

This gives us the parameters, assuming we are absolutely sure our model is correct. What, however, if we want independent confirmation? In that case, we'd like another data point, i.e., another input-output pair, that agrees with these parameter values. Suppose we were additionally given the data point that $F(1,1) = 9$. This would be a validation of the model.

Suppose, however, that it turns out that $F(1,1) = 10$. That would suggest that the model is wrong. Could it be rescued partially? Yes, it might be rescuable if we assume some errors and approximations in our modeling and measurement process. On the other hand, an observation like $F(1,1) = 40$ (if actually correct and not simply a result of a "typo" or other weird measurement error) would likely mean that the model is close to useless.

We needed three input-output pairs to determine the parameters uniquely because there are three parameters in the generic form. To provide additional confirmation, we need more input-output pairs.

4.2. **Linear in parameters versus linear in variables.** There is a subtle but very important distinction to keep in mind. When we use input-output pairs to form equations in the parameters that we then solve, the nature of those equations depends, obviously, on the nature of the functional form. However, it depends on the way the function depends on the *parameters*, not on the way the function depends on the *input variables*. For instance, consider a function:

$$F(x,y) := axy + be^{x \sin y} + cx^2y^2$$

This function is definitely not linear in x or y, nor is it additively separable. However, the function is linear in the parameters a, b, and c. Thus, the system of equations we set up using (input, output) pairs for the function is a system of linear equations.

If a function is linear in the inputs, then assuming no nonlinear constraints relating the parameters, it is also linear in the parameters (note that the number of parameters is one more than the number of variables, and that's the number of observations we need to determine the parameters uniquely). However, as observed above, it is very much possible to be linear in the parameters but not in the variables. For this reason, linear algebra methods for parameter estimation can be applied to some functional forms that are not linear in the input variables themselves.

4.3. **Generating equations using methods other than input-output values.** The typical method used to generate equations for parameter determination is input-output value pairs. There are, however, other methods, based on other pieces of information that might exist about the function.

These may include, for instance, (input, derivative of output) pairs, or (average of inputs, average of outputs) pairs, or other information.

The case of the differential equation, which we might return to later, is illustrative. The *initial value specification* used for the solution of a differential equation usually involves an input value and the value of the output and many derivatives of the output at a single point.

4.4. **Once the parameters are determined.** Once the parameters for the model have been determined, we can proceed to use the usual methods of multivariable calculus to acquire a deeper understanding of what the function does. If the function is also linear in the input variables, then we can use the techniques of linear algebra. Otherwise, we are stuck with general techniques from multivariable calculus.

5. Creating general functional forms and counting parameters

One major stumbling block for people is in writing the general functional form for a model that correctly includes parameters to describe the various degrees of freedom. Writing the correct functional form is half the battle. It's important to have a theoretically well-grounded choice of functional form and to make sure that the functional form as described algebraically correctly describes what we have in mind.

Note also that it's important to maintain mentally the distinction between the number of parameters and the number of inputs.

5.1. **Some examples for functions of one variable and multiple variables.**

- A polynomial function $f(x)$ of one variable x that is given to have degree $\leq n$: The general functional form is:

$$a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

  The number of parameters here is $n+1$. Therefore, the number of well-chosen input-output pairs we expect to need is $n+1$. In fact, in this case, choosing $n+1$ inputs always allows us to determine the polynomial functon uniquely. This is related to the Lagrange interpolation formula and also to the idea of the Vandermonde matrix and Vandermonde determinant. Note that in order to obtain additional confirmation of the model, we need to have $n+2$ or more input-output pairs.
- A polynomial function $f(x,y)$ of two variables x and y, of total degre $\leq n$.

  The polynomial is obtained by taking linear combinations of monomials of the form x^iy^j where $i+j \leq n$. The number of such monomials depends on n, and the process involves just writing out all the monomials. For instance, in the case $n = 2$, the monomials are 1, x, y, x^2, xy, y^2, and the generic functional form is:

$$a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2$$

  The number of parameters is 6 in this case ($n = 2$), so we need 6 well-chosen inputs to find the parameters, and we need 7 or more to both find the parameters and obtain independent confirmation of the model.

  In general, the number of parameters is $(n+1)(n+2)/2$. This formula is not obvious, but we can work it out with some effort. Rather than memorize the formula, try to understand how we would go about writing the generic functional form.
- A polynomial function of multiple variables: The idea of the preceding point generalizes, but the number of parameters now grows considerably. In general, it grows both with the number of variables and with the degree of the polynomial that we are trying to use.
- A trigonometric function of the form $a\sin x + b\cos x + C$. This has three parameters a, b, and C. If $C = 0$ for theoretical reasons, then we have only two parameters.
- A sum of trigonometric functions, such as:

$$f(x) := a_1\sin(m_1x) + a_2\sin(m_2x)$$

  This is not linear in the parameters m_1 and m_2, but it is linear in the parameters a_1 and a_2. If we already know m_1 and m_2, we can use (input,output) pairs to find a_1 and a_2.

  This type of function can arise when considering musical sounds that involve multiple frequencies being played simultaneously, as is common with many musical instruments (fundamental and other modes of vibration for guitar strings, for instance, and the use of chords in music, etc.)

- A sum of exponential functions, such as:

$$f(x) := a_1 e^{m_1 x} + a_2 e^{m_2 x}$$

  This is not linear in the parameters m_1 and m_2, but it is linear in the parameters a_1 and a_2. If we already know m_1 and m_2, we can use (input,output) pairs to find a_1 and a_2.

  This type of situation arises when multiple exponential trends arising from different causes or sources get combined.

5.2. **Examples involving piecewise descriptions.** In some cases, instead of assuming a single functional form throughout the domain, we assume a function with a piecewise description, i.e., the function differs on different pieces of the domain. For functions of one variables, these pieces could be intervals. For functions of two variables x and y, these pieces could be rectangular regions, or otherwise shaped regions, in the xy-plane (where the domain of the function resides).

The function description on each piece involves a functional form that has some parameters. The total number of parameters needed for the functional description overall is the sum of the number of parameters needed on each piece. If the functional form is the same on all pieces, then the number of parameters needed is (number of parameters needed for the functional form) $\times$ (number of pieces).

For instance, consider a function of one variable that has a piecewise quadratic description with the domain $[0, 4]$ having four pieces: $[0, 1]$, $[1, 2]$, $[2, 3]$, and $[3, 4]$. For each piece, the functional form needs 3 parameters (on account of being quadratic). There are four such pieces, so we need a total of $3 \times 4 = 12$ parameters. If we assume continuity at the transition points 1, 2, and 3, then each of the continuity conditions gives an equation relating the descriptions, so we effectively have $12 - 3 = 9$ degrees of freedom.

In this case, a particular (input,output) pair would be helpful in so far as it helps pin down the parameters for the interval where the input lives. For inputs that are transition points, we can use its containment in either interval to generate an equation.

There is a lot more that can be said here, but since this topic is relatively tangential to the course as a whole, we will stop here. It may be beneficial to look at the relevant quiz and homework problems for a more detailed discussion of some variations of this setup.

EQUATION-SOLVING WITH A SPECIAL FOCUS ON THE LINEAR CASE: A PRIMER

MATH 196, SECTION 57 (VIPUL NAIK)

Executive summary

Words ...

(1) We need to solve equations both for determining the parameters in a general functional description of a model, and for using existing models to estimate the values of real-world variables. Equations are also useful in solving max/min problems.
(2) When evaluating equation-solving algorithms for suitability, we should consider time, memory, parallelizability, precision, and elegance.
(3) The dimension of the solution space to a system of equations that has no redundancies or inconsistencies is expected to be (number of variables) - (number of equations). If there are fewer equations than variables, we expect that the system is *underdetermined.* If there are more equations than variables, we expect that the system is *overdetermined.* Generally, this means that either some of the equations are redundant (i.e., they do not provide additional information) or the equations are inconsistent.
(4) For diagonal systems of equations, i.e., systems of equations where each equation involves only one variable, we can solve the equations separately, then string together the coordinates of the solution. The process is parallelizable. The solution set satisfies a *rectangular completion property* (note: this isn't a standard term, it's just a shorthand for something that's hard to describe without full symbolism; you might prefer to think of it as a *coordinate interchange property* if that name connects better with the concept for you).
(5) For triangular systems of equations, we need to solve the equations in a particular order, and if we do so, we have to solve a series of equations in one variable. The solution set need not satisfy the rectangular completion property. In case of multiple solutions, we need to make cases and branch each case. Parallelization is possible between the branches but not between the equations themselves.
(6) The same remarks apply for triangular systems of linear equations, except that there is no branching to worry about.
(7) When we manipulate equations (by adding, subtracting, and multiplying) to obtain new equations, we must remember to keep the old equations around to protect against loss of information. However, keeping *all* the old equations around can result in a memory overload. The general idea is to keep enough of the old equations around that the other old equations that we are discarding can be recovered from the new system. In other words, we want our transformations to be *reversible.*
(8) One manipulation technique is to use one of the equations to eliminate one of the variables by expressing it in terms of the other variables. If we are able to do this systematically all the way through, we would have reduced the original system to a triangular system.
(9) Another manipulation technique is to add and subtract multiples of one equation to another equation. We generally keep the equation that is being multiplied before the addition, and discard the equation that is not being multiplied by anything, because that guarantees reversibility. If the value we are multiplying with is a nonzero constant, we can also keep the other equation instead.
(10) We typically add and subtract equations with the explicit goal of eliminating variables or eliminating difficult expressions in terms of the variables.

Actions ...

(1) Please be sure not to throw out information when manipulating the system. When in doubt, store more rather than less.

(2) Please remember that, as soon as you discover an inconsistency, you can abandon the equation-solving attempt (or the branch you are in). All previous solutions that you have found don't amount to anything. In a diagonal system, if any one equation in the system has no solution for its corresponding variable, the system as a whole has no solution. In a triangular system, as soon as we discover an equation that has no solution, discard the branch (from the most recent branching point onward).

1. Equation-solving: why should I care?

In the context of creating and using models, equation-solving occurs in two contexts:

(1) Using input-output pairs (or *observations*) to determine or estimate parameters.
(2) Using outputs from known sets of functions to determine the values of certain inputs (a sort of *triangulation process*).

In application (1), we get a linear equation if and only if the generic functional forms are linear in the parameters (they need not be linear in the inputs). In application (2), we get a linear equation for each functional form that is linear in the inputs.

Apart from model construction and analysis, equation-solving is also key to optimization problems (finding maxima and minima of functions): a first-cut shortlist for the points of local maximum and minimum for a function is the set of critical points, which is obtained by setting all the partial derivatives to equal zero.

Equation-solving is key to going from theoretical models to concrete expressions.

The purpose of this course (and more so of this part of the course) is to become adept at solving linear equations. However, this is a good opportunity to reflect on the general methods of solving equation of all sorts, including equations that are not linear.

2. What should we worry about when (creating procedures for) solving equations?

Before proceeding, it might be worthwhile reviewing the list of things to be watchful for when solving equations and creating procedures for the task of solving equations:

- *Time* is a major constraint. There's a lot more to life than solving equations. The faster, the better.
- *Memory* is another constraint. Manipulating large systems of equations can take a lot of space. Lots of memory usage can also affect time indirectly. Ideally, our equation-solving process should use as little "working memory" as possible.
- *Parallelizability* is great. Parallelizability means that if multiple processors are available, the task can be split across them in a meaningful manner that cuts down on the time required. Some kinds of procedures are parallelizable, or partially so. Others are not parallelizable.
- *Numerical precision* is another issue, particularly in cases where the numbers involved are not rational numbers. The degree of precision with which we measure solutions is important. Lots of major errors occur because of inappropriate truncation. Hypersensitivity to small errors is a bad idea.
- *Elegance and code simplicity*: Another criterion that is arguably important is the complexity of the code that powers the algorithm. Given two algorithms that take about the same amount of time to run, the algorithm that is based on easier, simpler code is preferable, since the code is easier to maintain, tweak, and debug. Elegance is also valuable for aesthetic reasons, which you may or may not care about.

3. Dimension and codimension theory

Brief recap of dimension theory: suppose we start with $\mathbb{R}^n$. We now impose m mutually independent constraints in the form of equations. In other words, we start with all points in $\mathbb{R}^n$, and each constraint whittles down our set of points. The general expectation is that each scalar equation should whittle down the dimension of the set by 1. Thus, the dimension of the subset of $\mathbb{R}^n$ satisfying m independent constraints should be $n - m$.

We can think of the original n (the dimension of the variable space) as the number of variables and the value m (the number of constraints) as the number of equation. Thus:

Dimension of solution space = Number of variables - Number of equations (generically)

In particular:

- If $m < n$, we generically expect that the solution space is positive-dimensional. In other words, we should expect lots of points, possibiy infinitely many of them, that satisfy the system of constraints. We say that such a system is *underdetermined*.

  The key word is *generically*. If the constraints happen to not be independent (either because they are *redundant* or because they are *inconsistent*) a lot of weird things could happen to the solution set. Inconsistency could result in an empty solution set. Redundancy could result in a solution set of higher dimension, because we effectively "wasted" some constraints.
- If $m = n$, we generically expect that the solution space is zero-dimensional. In other words, we should expect to get finitely many solution candidates, or at any rate, a discrete solution set. In the linear situation, the solution will be unique, but for polynomial situations, we could get finitely many solutions without having a unique solution.
- If $m > n$, the solution space is expected to have negative dimension and we say that the system is *overdetermined*. What this essentially means is that it is not really possible for the constraints to be independent. If there is a there there, so to speak, then the system will be consistent and we will still get solutions. Otherwise, the system will not be consistent and we will get an empty solution set.

4. Diagonal systems

Consider a situation with three variables x, y, and z. Suppose we are given the three equations:

$$\begin{aligned} x^2 &= 2 \\ \sin y &= 0 \\ 2z - z^2 &= -10 - z \end{aligned}$$

In principle, this is a system of three equations in three variables. In practice, the system is particularly nice, because each equation "talks about" only one variable. Thus, we can solve each equation as an equation in one variable.

The first equation pins down the value of x to $x = \pm\sqrt{2}$. Applied as a constraint on the entire xyz-space $\mathbb{R}^3$, this brings us down from all of $\mathbb{R}^3$ to the union of the two planes $x = \sqrt{2}$ and $x = -\sqrt{2}$, both parallel to the yz-plane.

The second equation tells us that y is an integer multiple of π. Note that the solution set, though infinite, is discrete and zero-dimensional. (The precise meaning of these concepts is beyond our present scope). Viewed as an equation on $\mathbb{R}^3$, the solution set to this is the union of planes of the form $y = n\pi, n \in \mathbb{Z}$, each parallel to the xz-plane. If we view this together with the first equation, then we have *two constraints* in $\mathbb{R}^3$ and should expect a one-dimensional solution set. Indeed we do get such a solution set: the set of all lines of the form $x = \pm\sqrt{2}, y = n\pi, n \in \mathbb{Z}$.

The third equation is an equation purely in terms of z. If we rearrange and solve, we get $z = 5$ or $z = -2$. The solution set in $\mathbb{R}^3$ for this alone is a union of the two planes $z = 5$ and $z = -2$. The overall solution set of all three equations together is the intersection of all three solution sets, namely, the zero-dimensional (but infinite) set of points of the form:

$$\{(x, y, z) : x = \pm\sqrt{2}, y = n\pi, n \in \mathbb{Z}, z = 5 \text{ or } z = -2\}$$

We call equation systems of this type diagonal systems of equations. Note the following interesting things about diagonal systems of equations:

- The solution procedure for diagonal systems is simply to solve each of the equations individually as an equation in one variable, then string together the solutions in coordinates.

  This is not to say or imply that a diagonal system can always be solved. Diagonal systems can also be difficult to solve. However, the difficulty with diagonal systems does not lie with the entanglement of the variables. Rather, the difficulty, if there is any, must be with solving one of the equations in one variable. There is no additional complexity arising from the presence of multiple variables, since they do not interact at all in the equations.

Further, even if one or more of the individual equations in the diagonal system does not admit an exact solution using known methods, it may be possible to determine approximate solutions using numerical analysis methods. If approximate solutions for each variable can be determined, this gives approximate solutions for the system as a whole.

- The solution set also has a rectangular completion property as follows: if two points are in it, then so are any other points where we swap in and out the coordinates of the point. In the three-dimensional case, if (x_1, y_1, z_1), and (x_2, y_2, z_2) are solutions, so are (x_1, y_2, z_1), (x_2, y_2, z_1) and all the other points we can get in this fashion.

Exercise for the reader: Evaluate the procedure for solving diagonal systems for time, memory, parallelizability, and numerical precision.

Answer:

- *Time*: In a serial computation context, it is basically the sum of the times taken to solve each equation individually.
- *Memory*: If we solve the equations serially, the memory overhead at any point in time is:
  - For the equations already solved: Store the solution set.
  - For the equation being solved: The full working memory needed for its solution procedure.
  - For the equations yet to be solved: Store the equations.
- *Parallelizability*: The system is massively parallelizable. Each equation can be solved independently of the others.
- *Numerical precision*: This is a relative non-issue for the single variable to multivariable transition.
- *Elegance and code simplicity*: Assuming we can invoke modular code to solve the equations in one variable, the code to solve a diagonal system in terms of solving the individual equations is quite simple.

4.1. Diagonal systems of linear equations.

A diagonal system of linear equations is particularly easy to solve. Unlike the general case, we know that each linear equation in one variable can be one of these three things:

- It has a unique solution
- It is inconsistent, i.e., it simplifies to $0 = 1$.
- It conveys no information, i.e., it simplifies to $0 = 0$.

If, for our linear system, the "inconsistent" case arises for any equation, then there are no solutions to the system. Otherwise, in each coordinate, we either have a unique possibility or the whole real line. Overall, then, our solution set will be an affine subspace (whatever that means) parallel to one of the coordinate subspaces, where some coordinates are completely and uniquely fixed, and other coordinates are completely free to vary.

If a variable does not appear in any equation, it is free to vary over all reals. For completeness, we may throw in a "conveys no information" equation of the form $0x_i = 0$ for each such variable.

5. Non-diagonal systems: general remarks

5.1. The entanglement of variables.

Conceptually, we can think of diagonal systems as *disentangled* systems of equations: the entanglements between the variables have been removed so that we can concentrate on each variable individually. For general systems of equations, there exist entanglements between the variables. Consider, for instance:

$$\begin{aligned} x^2 - 2xy &= y^2 + z^2 - 3xyz \\ -x^3 - 4y^3 &= -5 \\ xyz &= 1 + (x-1)z^2 \end{aligned}$$

This kind of system is extremely difficult because the variables are entangled with each other. It is not easy to look at this and solve for one variable at a time, because *every* equation is a jumble of the variables.

What we need to do is *rethink* the system in a way that the variables are more clearly separated. The goal is to move to an *equivalent* system that is more separated. We will now proceed to see a number of techniques used for this end. Before that, however, we need to note a few caveats.

5.2. **Protecting against the loss of information.** Let's look at a particular system of equations in two variables that actually came up naturally when trying to find critical points for a function in a multivariable calculus course:

$$\begin{aligned} 4x^3 - 2xy^2 &= 0 \\ -2x^2y + 4y^3 &= 0 \end{aligned}$$

Here is what some people did: multiply the first equation by x, the second equation by y, and subtract. The mixed products cancel under subtraction, and we are left with:

$$4x^4 - 4y^4 = 0$$

This simplifies to giving the union of the line $y = x$ and the line $y = -x$.

Is that the solution set to the original system? Definitely, any solution to the original system must satisfy the new equation $4x^4 - 4y^4 = 0$, and hence, must be on at least one of the two lines. However, the new equation does not convey all the information stored in the original pair of equations. It is a single constraint, whereas the original system had two constraints. Thus, it is possible (and in this case true) that the solution set to the new equation is strictly bigger than the solution set to the old system.

What should we do instead? There are many alternate approaches to solving the question, but this approach can be easily fixed. All we need to do is remember to keep the old system alongside the new equation. Thus, instead of just finding the solutions to the new equation, we test these solutions against the old system. In this case, testing either of the solution lines against either of the equations in the old system gives us a unique solution $x = 0$, $y = 0$.

So, one approach we can use is that even as we try to add, subtract, and manipulate our old system to create a new system, we preserve all the old equations, to avoid any loss of information. This approach, however, is not scalable, because it leads to "equation bloat" as a number of redundant equations are kept on the books for fear of losing something valuable. Usually, it suffices to keep *some* of the old equations on the books, just enough that we do not lose information, but not enough to create unnecessary redundancy and not enough to take too much space.

The key idea is to think of the process as transforming the entire system together, and then ask of a transformation: is it *reversible*? Can we recover the old system from the new one? Certain kinds of transformation rules are reversible by nature. If we stick to those, we can do the manipulations mechanically and not have to explicitly think about loss of information.

Alternative solution approach for the example used. An alternative solution approach is to note that the solutions to the two equations are respective:

$$\begin{aligned} 4x^3 - 2xy^2 = 0 &\iff x = 0 \text{ or } y^2 = 2x^2 \\ -2x^2y + 4y^3 = 0 &\iff y = 0 \text{ or } x^2 = 2y^2 \end{aligned}$$

Let's try all four possibilities:

- $x = 0$ and $y = 0$: This is one solution.
- $x = 0$ and $x^2 = 2y^2$: This solves to $x = 0$, $y = 0$.
- $y^2 = 2x^2$ and $y = 0$: This solves to $x = 0$, $y = 0$.
- $y^2 = 2x^2$ and $x^2 = 2y^2$: Combining, we get $4x^2 = x^2$, so $x = 0$. Plug in and get $y = 0$. Thus, this solves to $x = 0$, $y = 0$.

All four cases end up giving the same point solution $x = 0$, $y = 0$. The fact that all cases end up giving the same solution is a special feature of this system of equations, and is not illustrative of any more general principle.

6. Triangular systems

6.1. **An example and the general idea.**

We've already talked about diagonal systems, but there is another kind of system that bears special mention: *triangular system.*

A triangular system of equations is a system where we can arrange the variables so that there is one equation involving only the first variable, another equation involving only the first two, another equation involving only the first three, and so on. For instance, this system is triangular:

$$\begin{aligned} x^2 &= 2 \\ 2x + x^2y &= x^3y^2 \\ xy + xz + yz &= 0 \end{aligned}$$

The first equation involves only x, hence can be solved for x. The second equation involves only x and y. We can plug in solution(s) to the first equation into the second equation, making it an equation in one variable y. Solve this for y. Having got possible values of x and y, we can plug these into the third equation to solve for z.

Note that the existence of multiple solutions at any stage means that we need to make cases for the next stage. In our case, the first equation has two solutions $x = \sqrt{2}$ and $x = -\sqrt{2}$. We have to make two branches for the entire rest of the system based on what solution we choose. Eventually, we will pool together all solutions. Similarly, if in either branch, we get multiple solutions for y, we must branch further. The presence of such branching renders the later coordinates potentially dependent on the earlier coordinates. Thus, the solution set need not have the rectangular completion property observed for diagonal systems.

Solution details for this example. The first equation gives two cases: $x = \sqrt{2}$ and $x = -\sqrt{2}$. Let's consider the subcase $x = \sqrt{2}$. Plugging this into the second equation, we get:

$$2\sqrt{2} + 2y = 2\sqrt{2}y^2$$

Simplify and solve this quadratic equation to get:

$$y = \sqrt{2} \text{ or } y = -1/\sqrt{2}$$

We can further make branches based on the value of y. If we choose $y = \sqrt{2}$, the third equation becomes:

$$2 + 2\sqrt{2}z = 0$$

This simplifies to $z = -1/\sqrt{2}$.

Alternatively, if we choose $y = -1/\sqrt{2}$, we get:

$$-1 + z/\sqrt{2} = 0$$

This simplifies to $z = \sqrt{2}$.

Thus, we get two solutions from the subcase $x = \sqrt{2}$:

- $x = \sqrt{2}, y = \sqrt{2}, z = -1/\sqrt{2}$
- $x = \sqrt{2}, y = -1/\sqrt{2}, z = \sqrt{2}$

Now, consider the case $x = -\sqrt{2}$. In this case, the second equation becomes:

$$-2\sqrt{2} + 2y = -2\sqrt{2}y^2$$

Simplify and solve this quadratic equation to get:

$$y = -\sqrt{2} \text{ or } y = 1/\sqrt{2}$$

If we choose $y = -\sqrt{2}$, we get:

$$2 - 2\sqrt{2}z = 0$$

This simplifies to $z = 1/\sqrt{2}$.

If we choose $y = 1/\sqrt{2}$, we get:

$$-1 - (1/\sqrt{2})z = 0$$

This solves to $z = -\sqrt{2}$.

Thus, we get two solutions from the subcase $x = -\sqrt{2}$:

- $x = -\sqrt{2}, y = -\sqrt{2}, z = 1/\sqrt{2}$
- $x = -\sqrt{2}, y = 1/\sqrt{2}, z = -\sqrt{2}$

Overall, we get four solutions:

- $x = \sqrt{2}, y = \sqrt{2}, z = -1/\sqrt{2}$
- $x = \sqrt{2}, y = -1/\sqrt{2}, z = \sqrt{2}$
- $x = -\sqrt{2}, y = -\sqrt{2}, z = 1/\sqrt{2}$
- $x = -\sqrt{2}, y = 1/\sqrt{2}, z = -\sqrt{2}$

It is clear by looking at this solution set that it does not satisfy the "rectangular completion property." Specifically, we cannot swap values of coordinates between different solutions and still be guaranteed to have a solution.

6.2. **Evaluating the pros and cons of triangular systems.** Both diagonal systems and triangular systems are "essentially solved" types of systems of equations. In both types of systems, we have a strategy to reduce a multivariable system to single variable equations. There are a few ways, however, in which triangular systems are more complicated than diagonal systems. Some of these are stated below:

- In a diagonal system, the individual equations in the separate variables can be solved separately. In other words, the computations can be carried out in *parallel* and the solutions pooled together. Diagonal systems are massively parallelizable.

  For a triangular system, on the other hand, we need to solve the equations *serially* in a very particular order. We first solve the equation with only one variable, then the equation with that and another variable, and so on. Since the process of solving the second equation relies on the solution to the first, the process of solving triangular systems is inherently harder to parallelize.
- In a diagonal system, there is no branching into cases. In a triangular system, on the other hand, multiple solutions to one equation lead to a branching of cases for solving the remaining equations. Note that these branches can be dealt with through parallel processing, but the degree of branching required is generally *a priori* unclear since it depends at each stage on how many solutions each particular equation will turn out to have.

Note that in the case of a triangular system of *linear* equations, the branching issue does not arise, because at each stage, we must get a unique solution. The first issue, namely the issue of non-parallelizability, is still an important problem.

Exercise for the reader: Evaluate the solution procedure for triangular systems for time, memory, parallelizability, and numerical precision.

- *Time*: Depends both on the number of equations and on the number of solutions to each equation. Consider the case where the triangular system has n equations that are of degrees $m_1, m_2, \dots, m_n$ respectively. The total number of equations we effectively need to solve could be as large as $1 + m_1 + m_1m_2 + \dots + m_1m_2 \dots m_{n-1}$.

  We could do a "depth-first" or a "breadth-first" algorithm.
- *Memory*: If proceeding serially, we need to store the solution strings accumulated so far, the working memory for whatever current equation we are working on, and all the equations we are yet to work on. The amount of information that needs to be stored will, like time, depend on the number of solutions to the first few equations.

- *Parallelizability*: We cannot parallelize the system itself. However, we could parallelize the branching process. Every time we encouter multiple solutions, we create multiple new processor nodes and seed them with the information on the solution string so far. If we use this process, the time for every serial strand involves solving n equations. However, the unpredictability in the number of processors needed and the complexity of passing information around are important problems.
- *Numerical precision*: This could be an issue, particularly with polynomials of high degree where precise algebraic expressions for the solutions do not exist. Unlike the diagonal case, the problem is that we need solution values to some equations to plug into other equations. Thus, numerical imprecision for some solutions could lead us to go awry later. Or to put it another way, *errors could compound.*
- *Elegance and code simplicity*: The code becomes complicated because it involves recursion and/or branching. Note that this complexity does not arise in the case of a triangular system of linear equations. Parallelizing the code imposes additional coding challenges.

6.3. **The no solutions case.** For a triangular system, if you obtain an equation that has no solutions, that means that the sub-branch you are working in has no solution. Look for the most recent branching point, and discard all work below that. For instance, consider:

$$\begin{aligned} x^2 &= 3 \\ y^2 &= 2 \\ z &= x + y \\ w^2 &= z \end{aligned}$$

If we choose $x = -\sqrt{3}$, $y = \sqrt{2}$, then $z = \sqrt{2} - \sqrt{3}$, but $w^2 = z$ has no solution. This means that there are no solutions in the branch $x = -\sqrt{3}$, $y = \sqrt{2}$. Similarly, there are no solutions in the branch $x = -\sqrt{3}$, $y = -\sqrt{2}$, so we need to discard those branches.

6.4. **Triangular systems of linear equations.** As noted above, triangular systems of linear equations need to be solved serially, but there are no major branching issues.

Recall the following three cases for a linear equation in one variable:

- It has a unique solution
- It is inconsistent, i.e., it simplifies to $0 = 1$.
- It conveys no information, i.e., it simplifies to $0 = 0$.

As long as our solution process sticks with the first case, we will keep building a unique solution string for the variables solved for so far. If we ever hit the second case, there is no solution, and the process can terminate. If we hit the third case for some variable, then that variable is free to vary over all reals. Subsequent variables will not be determined as unique real numbers, but as linear expressions in that variable, which we can treat as a parameter. What we will get, essentially, is a *family* of solutions. Consider, for instance, the system:

$$\begin{aligned} 2x_1 &= 6 \\ x_1 + x_2 + x_3 &= 7 \\ x_1 - x_2 + x_3 - x_4 &= 10 \end{aligned}$$

There is a missing equation relating x_1 and x_2, so we can fill in such a missing "conveys no information" equation, basically saying that x_2 is free to take any value. We get $x_1 = 3$. The variable x_3 is $7 - x_1 - x_2 = 4 - x_2$. We cannot simplify it further because x_2 is not unique.

We can now solve the fourth equation to write x_4 and we get:

$$x_4 = -3 - 2x_2$$

Thus, our general solution is the line:

$$\{(3, x_2, 4 - x_2, -3 - 2x_2) : x_2 \in \mathbb{R}\}$$

Note that the dimension of this solution set is 1, and it involves a free parameter. This is to be expected because we had four variables but only three equations. Geometrically, it is a line in $\mathbb{R}^4$.

Keep in mind the nature of the solution set you see above. You'll see this sort of description repeatedly, in somewhat different forms, throughout the course. Make sure you have a clear understanding of what it means.

6.5. **A cool fact about linear systems.** One cool thing about systems of linear equations is that every system of linear equations is equivalent to a triangular system. This statement deserves more elaboration, which we shall provide later.

7. Typical manipulation techniques

7.1. **Eliminating a variable through substitution.** In this technique, we solve for one variable in terms of the others from one equation, then replace that variable by the expression in terms of the others into all the other equations. Note that this process:

- reduces the number of variables by one
- reduces the number of equations by one (namely, the equation that we manipulate to solve for one variable in terms of the others).

The simultaneous reduction in the number of variables and the number of equations is no coincidence. Recall that the (expected) dimension of the solution space is given by (number of variables) - (number of equations). The dimension of the solution space should be invariant under manipulation processes. When we eliminate a variable, we also eliminate an equation.

7.1.1. *Example 1 (not solved completely).* Consider the example:

$$\begin{aligned} x^2 - 2xy &= y^2 + z^2 - 3xyz \\ -x^3 - 4y^3 &= -5 \\ xyz &= 1 + (x-1)z^2 \end{aligned}$$

Note that the third equation is linear in y, so we can simplify it to get y in terms of x and z, then eliminate y. Explicitly:

$$y = \frac{1 + (x-1)z^2}{xz}$$

Note that there is a caveat case where the expression does not work. This is the case $x = 0$ and $z = \pm 1$. This case would need to be dealt with separately to see if it is consistent with the other equations. Setting that aside for now, we can plug in the expression for y in the other two equations to get:

$$\begin{aligned} x^2 - 2x\frac{1 + (x-1)z^2}{xz} &= \left(\frac{1 + (x-1)z^2}{xz}\right)^2 + z^2 - 3x\left(\frac{1 + (x-1)z^2}{xz}\right)z \\ -x^3 - 4\left(\frac{1 + (x-1)z^2}{xz}\right)^3 &= -5 \end{aligned}$$

Note that this system is still too complicated.

7.1.2. *Example 2 (solved completely, but only one case can be handled by hand calculation).* In a simpler world, we can keep eliminating variables one by one, then we ultimately get one equation in one variable, that we can solve to get its value. Now, we need to find the other variables. This process now works as a triangular system! We now turn to an example of such a system.

$$\begin{aligned} x+y+z &= 6 \\ x^2+2y+3z &= 16 \\ x^2+y^2+z^2 &= 14 \end{aligned}$$

Use the first equation to eliminate z by writing it as $z = 6 - x - y$. Now, the second and third equation become:

$$\begin{aligned} x^2+2y+3(6-x-y) &= 16 \\ x^2+y^2+(6-x-y)^2 &= 14 \end{aligned}$$

We can use the first of these to eliminate y by getting $y = x^2 - 3x + 2$. Thus, we are left with one equation:

$$x^2 + (x^2-3x+2)^2 + (6-x-x^2+3x-2)^2 = 14$$

This simplifies to:

$$2(3+2x+5x^2-5x^3+x^4) = 0$$

This is a degree four polynomial equation in one variable. It turns out that it has two real roots and two complex roots. The real roots are $x = 3$ (determinable by inspection) and a more complicated one whose approximate vaue is 2.546 (that requires considerably computation or help from a graphing calculator or computational software). Consider the two cases:

- $x = 3$: We can back-substitute in the expression for y in terms of x and then in the expression for z in terms of y to get $y = 2$ and $z = 1$. Thus, one solution is $x = 3$, $y = 2$, $z = 1$.
- $x \approx 2.546$: This solution (which is difficult to compute by hand) is $x \approx 2.546$, $y \approx 0.844$, $z \approx 2.61$. The values of y and z are obtained by back-substitution.

In fact, properly considered, our system should be viewed as the following triangular system:

$$\begin{aligned} 2(3+2x+5x^2-5x^3+x^4) &= 0 \\ y &= x^2-3x+2 \\ z &= 6-x-y \end{aligned}$$

Note that this eliminate-and-substitute approach can rarely be done all the way to the point that we get a triangular system. In the linear situation, however, this can always be done.

7.1.3. *A linear example.* We will see a lot of linear examples, so we will be brief here.

$$\begin{aligned} x+y+z &= 6 \\ 2x+3y+5z &= 16 \\ x-y+3z &= -2 \end{aligned}$$

We can use the first equation to obtain:

$$z = 6-x-y$$

We can substitute in the other two equations:

$$\begin{aligned} 2x + 3y + 30 - 5x - 5y &= 16 \\ x - y + 18 - 3x - 3y &= -2 \end{aligned}$$

The system simplifies to:

$$\begin{aligned} 3x + 2y &= 14 \\ 2x + 4y &= 20 \end{aligned}$$

We can use the first equation to write $y = (14 - 3x)/2$. We can then plug into the third equation and obtain:

$$2x + 2(14 - 3x) = 20$$

This simplifies to $x = 2$. We substitute $y = (14 - 3x)/2 = 4$ and $z = 6 - x - y = 0$. The unique solution is thus $x = 2$, $y = 4$, $z = 0$.

7.2. **Adding and subtracting equations.** Given two equations:

$$\begin{aligned} F(x_1, x_2, \dots, x_n) &= 0 \\ G(x_1, x_2, \dots, x_n) &= 0 \end{aligned}$$

Suppose λ is a real number. We can replace this system by the system:

$$\begin{aligned} F(x_1, x_2, \dots, x_n) + \lambda G(x_1, x_2, \dots, x_n) &= 0 \\ G(x_1, x_2, \dots, x_n) &= 0 \end{aligned}$$

Basically, we have added a multiple of the second equation to the first equation. Note that it is *not* enough to just store the equation:

$$F(x_1, x_2, \dots, x_n) + \lambda G(x_1, x_2, \dots, x_n) = 0$$

because as a *single* equation, this loses some of the information present in the original system. However, if we retain G alongside, we can recover the original system from the new system. The reason is that starting with the new system, subtracting λ times the second equation from the first recovers the first equation of the old system.

Note that if $\lambda \neq 0$, we could also store $F + \lambda G$ and F together instead of $F + \lambda G$ and G. However, it is better to store G. If we do this, it more readily generalizes to the case where λ is replaced by some function $H(x_1, x_2, \dots, x_n)$. Explicitly:

$$\begin{aligned} F(x_1, x_2, \dots, x_n) &= 0 \\ G(x_1, x_2, \dots, x_n) &= 0 \end{aligned}$$

is equivalent to:

$$\begin{aligned} F(x_1, x_2, \ldots, x_n) + H(x_1, x_2, \ldots, x_n)G(x_1, x_2, \ldots, x_n) &= 0 \\ G(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

for any function $H(x_1, x_2, \ldots, x_n)$.

Suppose we multiply *both* F and G by constants and then add. If the coefficient on F in the sum is always nonzero, we can store the new equation alongside G and throw out F. If the coefficient on G in the sum is always nonzero, we can store the new equation alongside F and throw out G. If both coefficients are always nonzero, we can store the new equation alongside whichever one of F and G we please, and throw out the other.

7.3. **Adding and subtracting with the goal of eliminating.** We can in principle add and subtract equations in any number of ways, but only some of the ways of doing so are *useful*. One laudable goal is to add and subtract in a manner that eliminates a variable or an expression. Consider, for instance, the system:

$$\begin{aligned} x + \cos(y - x^2) &= 0 \\ y + x\cos(y - x^2) &= \pi/2 \end{aligned}$$

In this system, what's bugging us is the presence of $\cos(y - x^2)$, which is a trigonometric function of a polynomial, and is hard to work with. Let's try to eliminate this expression by adding and subtracting the equations. We can subtract x times the first equation from the second equation, but importantly, keep this alongside the first equation. Remember, we keep the equation which *is* being multiplied, so that we can recover the original system. We have:

$$\begin{aligned} x + \cos(y - x^2) &= 0 \\ y - x^2 &= \pi/2 \end{aligned}$$

We can now use the second equation to eliminate y by expressing it in terms of x, and get $y = x^2 + (\pi/2)$. Plugging thus into the first equation, we get:

$$x + \cos(\pi/2) = 0$$

Since $\cos(\pi/2) = 0$, we get $x = 0$, and plugging back gives $y = \pi/2$. Thus, the unique solution is $x = 0$, $y = \pi/2$.

7.4. **Linear systems: elimination is always possible.** Suppose we have a linear system of equations. One of the nice things about such a system is that it is always possible to add and subtract equations to eliminate variables. In fact, we can construct a systematic procedure to solve systems of linear equations by first adding and subtracting them in order to convert to a triangular system, then solving the resultant triangular system. We will see this procedure, called *Gauss-Jordan elimination*, in a subsequent lecture.

GAUSS-JORDAN ELIMINATION: A METHOD TO SOLVE LINEAR SYSTEMS

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 1.2.

Executive summary

(1) A system of linear equations can be stored using an *augmented matrix*. The part of the matrix that does not involve the constant terms is termed the *coefficient matrix*.
(2) Variables correspond to columns and equations correspond to rows in the coefficient matrix. The augmented matrix has an extra column corresponding to the constant terms.
(3) In the paradigm where the system of linear equations arises from an attempte to determine the parameters of a model (that is linear in the parameters) using (input,output) pairs, the parameters of the model are the new variables, and therefore correspond to the columns. The (input,output) pairs correspond to the equations, and therefore to the rows. The input part controls the part of the row that is in the coeficient matrix, and the output part controls the augmenting entry.
(4) If the coefficient matrix of a system of simultaneous linear equations is in reduced row-echelon form, it is easy to read the solutions from the coefficient matrix. Specifically, the non-leading variables are the parameters, and the leading variables are expressible in terms of those.
(5) In reduced row-echelon form, the system is inconsistent if and only if there is a row of the coefficient matrix that is all zeros with the corresponding augmented entry nonzero. Note that if the system is *not* in reduced row-echelon form, it is *still* true that a zero row of the coefficient matrix with a nonzero augmenting entry implies that the system is inconsistent, but the converse does not hold: the system may well be inconsistent despite the absence of such rows.
(6) In reduced row-echelon form, if the system is consistent, the dimension of the solution space is the number of non-leading variables, which equals (number of variables) - (number of nontrivial equations). Note that the all zero rows give no information.
(7) Through an appropriately chosen sequence of row operations (all reversible), we can transform any linear system into a linear system where the coefficient matrix is in reduced row-echelon form. The process is called Gauss-Jordan elimination.
(8) The process of Gauss-Jordan elimination happens entirely based on the coefficient matrix. The final column of the augmented matrix is affected by the operations, but does not control any of the operations.
(9) There is a quicker version of Gauss-Jordan elimination called Gaussian elimination that converts to row-echelon form (which is triangular) but not reduced row-echelon form. It is quicker to arrive at, but there is more work getting from there to the actual solution. Gaussian elimination is, however, sufficient for determining which of the variables are leading variables and which are non-leading variables, and therefore for computing the dimension of the solution space and other related quantities.
(10) The arithmetic complexity of Gauss-Jordan elimination, in both space and time terms, is polynomial in n. To get a nicely bounded bit complexity (i.e., taking into account the sizes of the numbers blowing up) we need to modify the algorithm somewhat, and we then get a complexity that is jointly poynomial in n and the maximum bit length.
(11) Depending on the tradeoff between arithmetic operations and using multiple processors, it is possible to reduce the arithmetic complexity of Gauss-Jordan elimination considerably.
(12) Gauss-Jordan elimination is not conceptually different from iterative substitution. Its main utility lies in the fact that it is easier to code since it involves only numerical operations and does not require the machine to understand equation manipulation. On the other hand, because the operations are

purely mechanical, it may be easier to make careless errors because of the lack of "sense" regarding the operations. To get the best of both worlds, use the Gauss-Jordan elimination procedure, but keep the symbolic algebra at the back of your mind while doing the manipulations.

(13) The Gauss-Jordan elimination process is only one of many ways to get to reduced row-echelon form. For particular structures of coefficient matrices, it may be beneficial to tweak the algorithm a little (for instance, swapping in an easier row to the top) and save on computational steps. That said, the existence of the Gauss-Jordan elimination process gives us a guarantee that we can reach our goal by providing one clear path to it. If we can find a better path, that's great.

(14) We can use the Euclidean algorithm/gcd algorithm/Bareiss algorithm. This is a variant of the Gauss-Jordan algorithm that aims for the same end result (a reduced row-echelon form) but chooses a different sequencing and choice of row operations. The idea is to keep subtracting multiples of rows from one another to get the entries down to small values (hopefully to 1), rather than having to divide by the leading entry. It's not necessary to master this algorithm, but you might find some of its ideas helpful for simplifying your calculations when solving linear systems.

1. Notational setup

1.1. **Augmented matrix and coefficient matrix.** Storing equations requires storing letters, operations, and equality symbols. For linear equations, however, there is a predictable manner in which the operations are arranged. Hence, linear equations can be stored and manipulated more compactly simply by storing certain *numbers*, namely the coefficients of the variables and the constant term, in a structured form.

1.1.1. *Example 1.* For instance, the system:

$$\begin{aligned} 2x + 3y + z &= 5 \\ x - 4y + 5z &= 7 \\ 11x + 3y + 6z &= 13 \end{aligned}$$

can be described using this box of numbers, called the *augmented matrix*:

$$\begin{bmatrix} 2 & 3 & 1 & \mid & 5 \\ 1 & -4 & 5 & \mid & 7 \\ 11 & 3 & 6 & \mid & 13 \end{bmatrix}$$

Note that this makes sense *only* if we agree beforehand what it's being used for. Explicitly, each row represents one equation, with the first entry representing the coefficient of x, the second entry representing the coefficient of y, the third entry representing the coefficient of z, and the final entry representing the constant term *placed on the opposite side of the equality symbol*.

1.1.2. *Example 2.* Note that, unlike the example used above, some linear systems may be written in a way where we cannot just "read off" the augmented matrix from the system, because the variables appear in different orders. The first step there is to rearrange each equation so that we can read off the coefficients. For instance, consider the system:

$$\begin{aligned} x - 3y + 5z &= 17 \\ y - 2(z - x) &= 5 \\ x + y + 2(z - 3y + 4) &= 66 \end{aligned}$$

Each equation is linear in each of the variables x, y, and z. However, the variables appear in different orders in the equations. Moreover, the third equation contains multiple occurrences of y and also contains constant terms on both sides. We need to simplify and rearrange the second and third equation before being able to read off the coefficients to construct the augmented matrix. Here's what the system becomes:

$$\begin{aligned} x - 3y + 5z &= 17 \\ 2x + y - 2z &= 5 \\ x - 5y + 2z &= 58 \end{aligned}$$

The augmented matrix can now be read off:

$$\begin{bmatrix} 1 & -3 & 5 & \mid & 17 \\ 2 & 1 & -2 & \mid & 5 \\ 1 & -5 & 2 & \mid & 58 \end{bmatrix}$$

1.1.3. *Return to Example 1 for the coefficient matrix.* There is a related concept to the augmented matrix, namely the *coefficient matrix*. The coefficient matrix ignores the constant terms (the last column in the augmented matrix). Recall our first example:

$$\begin{aligned} 2x + 3y + z &= 5 \\ x - 4y + 5z &= 7 \\ 11x + 3y + 6z &= 13 \end{aligned}$$

The coefficient matrix for this is:

$$\begin{bmatrix} 2 & 3 & 1 \\ 1 & -4 & 5 \\ 11 & 3 & 6 \end{bmatrix}$$

1.2. **Dimensions of the coefficient matrix and augmented matrix.** If there are n variables and m equations, then the coefficient matrix is a $m \times n$ matrix: m rows, one for each equation, and n columns, one for each variable. The augmented matrix is a $m \times (n+1)$ matrix. The additional column is for the constant terms in the system.

For a typical precisely determined linear system, we expect that the number of equations equals the number of variables. Thus, if there are n variables, we expect n equations. The coefficient matrix is thus expected to be a $n \times n$ matrix. A matrix where the number of rows equals the number of columns is termed a *square matrix*, so another way of putting it is that the coefficient matrix is a square matrix. The augmented matrix is expected to be a $n \times (n+1)$ matrix, with the extra column needed for the constant terms.

Explicitly:

- Variables correspond to columns, so (number of variables) = (number of columns in coefficient matrix) = (number of columns in augmented matrix) - 1.
- Rows correspond to equations, so (number of equations) = (number of rows in coefficient matrix) = (number of rows in augmented matrix).
- (Number of variables) - (Number of equations) = (number of columns in coefficient matrix) - (number of rows in coefficient matrix). In an ideal world, this number should be the dimension of the solution space. The world is not always ideal, something we shall turn to in a while.

1.3. **The application to parameter estimation (you won't understand this completely just by reading, so pay attention in class and/or return to this later again).** Recall that if we have a functional model that is linear in the parameters (though not necessarily in the variables) we can use (input,output) pairs to construct a system of linear equations that we can then solve for the parameters.

We now note that:

- The coefficient matrix is completely determined by the *input* part of the input-output pairs.
- The additional column in the augmented matrix that stores the constant terms of the linear system is completely determined by the *output* part of the input-output pairs.

This is significant in a number of ways. As we will see in matrix theory, a lot of the basic nature of a linear system is controlled by the coefficient matrix. Roughly, if the coefficient matrix satisfies certain conditions, then the system has no redundancy and no inconsistency. If the coefficient matrix does not satisfy these conditions, then the system has potential for either redundancy or inconsistency. Which of these cases occurs depends on the outputs obtained.

We will also study techniques to solve linear systems that do the bulk of their processing on the coefficient matrix alone. An advantage of using such techniques is that much of the solution process can be run even without knowing the outputs, and the part of the algorithm that relies on the outputs can be done quickly. This is useful for two reasons: *amortization in repeated application*, and *faster processing once the outputs are known*.

To be more explicit: In some cases, the same model is being used for many different situations with different parameter values. For instance, a particular model of the economy may be used for different historic eras and different countries, where we expect the parameter values to be different. We can thus standardize a choice of inputs in input-output pairs that is common to all the uses of the model, then preprocess the solution procedure, and thus apply this to each of the model situations quickly on receiving the outputs. Similar remarks apply to models such as weather and traffic systems that have a fixed pattern of input collection at periodic intervals and need to rapidly process the outputs obtained at a particular interval to estimate the parameters involved.

2. Row reduction

Row reduction is a process for manipulating a system of linear equations to obtain an equivalent system of linear equations that is easier to solve. In this context, *equivalent* means that the solutions to the old system are *precisely* the same as the solutions to the old systems. Another way of thinking of this is that we can deduce the new system from the old system *and* we can deduce the old system from the new system. In other words, all the transformations that we do on the system of equations need to be *reversible transformations*.

We saw in a previous lecture that given a system of equations:

$$\begin{aligned} F(x_1, x_2, \ldots, x_n) &= 0 \\ G(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

and a real number λ, we can transform this to a new system:

$$\begin{aligned} F(x_1, x_2, \ldots, x_n) + \lambda G(x_1, x_2, \ldots, x_n) &= 0 \\ G(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

This type of transformation is sometimes called a *shear operation* based on a geometric interpretation. The shear operation by λ is reversible because its inverse is the shear operation by $-\lambda$.

We also saw that shear operations (and more generally, adding and subtracting equations) are typically used in order to eliminate certain variables or certain expressions involving variables.

The shear will be the most important and nontrivial of the three types of *elementary operations* that we use in order to simplify linear systems. The three operations are:

(1) Multiply or divide an equation by a nonzero scalar.
(2) Add or subtract a multiple of an equation to another equation (the shear operation).
(3) Swap two equations.

Interestingly, all these operations can be done purely as operations on the augmented matrix, without having to (tediously) write down the variables and operational symbols. The operations on the augmented matrix work in the same way. Explicitly, the three operations are:

(1) Multiply or divide a row by a nonzero scalar.
(2) Add or subtract a multiple of a row to another row (the shear operation).
(3) Swap two rows.

2.1. **Reduced row-echelon form.** The reduced row-echelon form can be thought of as trying to capture the idea of a triangular system where some equations are missing. It further restricts attention to particularly nice triangular forms that cannot be made to look nicer.

A matrix is said to be in reduced row-echelon form if it satisfies the following conditions:

(1) For every row, the left-most nonzero entry, if any, is a 1. We will call this the *pivotal* entry for the row and the column where this appears the pivotal column. It is possible for a row to be all zeros.
(2) The pivotal column for a later (lower) row is a later (more to the right) column, if it exists. If a given row is all zeros, all later rows are also all zeros.
(3) Any column containing a pivotal 1 must have all other entries in it equal to 0. This has two subparts:
    (a) All entries in the column below the pivotal 1 are equal to 0 (note that this is redundant if we assume both 1 and 2).
    (b) All entries in the column above the pivotal 1 are equal to 0.

Of the above, 3(b), while part of the definition of reduced row-echelon form, is not necessary for the system to be triangular. Condition 1 is also not strictly necessary, i.e., we can relax that condition and use the term *pivotal entry* for the left-most nonzero-valued entry in the row. The main advantage of 1 and 3(b) is mostly in terms of computational simplicity, both as far as the process for obtaining the echelon form is concerned (something we discuss in a subsequent section), and as far as using it to find the actual solution is concerned.

Conditions 2 and 3(a) are the conditions that really enforce the triangular nature of the system.

2.2. **Solving a linear system whose coefficient matrix is in reduced row-echelon form.** If the coefficient matrix for a system of linear equations is in reduced row-echelon form, then we say that the *leading variable* (also called the pivotal variable) for a particular equation is the variable corresponding to the pivotal column for that row.

Consider the following linear system in four variables x_1, x_2, x_3, and x_4:

$$\begin{aligned} x_1 - 2x_2 + x_4 &= 5 \\ x_3 + 5x_4 &= 31 \end{aligned}$$

The coefficient matrix for this system is as follows:

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

The leading 1 for the first row occurs in the first column. The leading 1 for the second row occurs in the third column. Note that the first nonzero entry in each row is a 1, and columns for later rows are later columns. This confirms conditions 1 and 2. Condition 3 is also clearly true here.

To solve a system of this sort, we start from the variables corresponding to the later columns and move backward to the variables corresponding to the earlier columns. First, consider the variable x_4, corresponding to the last column. This column is not pivotal for any row. So, x_4 is free to vary over all reals. It is helpful to name it using a letter customarily used to describe a parameter, such as the letter t.

The column for x_3 is pivotal for the second equation. Thus, the second equation can be used to deduce the value of x_3 based on the values of all the later variables. In this case, the equation is:

$$x_3 + 5x_4 = 31$$

Note that $x_4 = t$, so we get:

$$x_3 = 31 - 5t$$

x_2 does not appear as a leading variable for any row. Thus, it is again free to vary over all reals. Let s be the freely varying value of x_2.

This leaves x_1, which is pivotal for the first equation. We have the equation:

$$x_1 - 2x_2 + x_4 = 5$$

This gives:

$$x_1 - 2s + t = 5$$

This simplifies to:

$$x_1 = 5 + 2s - t$$

Thus, our solution set is:

$$\{(5 + 2s - t, s, 31 - 5t, t) : s, t \in \mathbb{R}\}$$

Note that if we had a third equation in the system which read:

$$0x_4 = 1$$

the coefficient matrix would be:

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

and the system would be inconsistent. On the other hand, if the third equation reads:

$$0x_4 = 0$$

the coefficient matrix would be the same as above, but the system is consistent, albeit the third equation conveys no information and might as well be dropped.

The following are clear from these examples for a *system in reduced row-echelon form*:

- The number of leading variables for a system in reduced row-echelon form can be given by the formula:
  Number of leading variables = (Number of rows that are not identically zero) = (Total number of rows) - (Number of rows that are identically zero)
  And therefore:
  Number of non-leading variables = (Number of columns) - (Number of rows that are not identically zero)
- For the system to be consistent, all rows with zeros for the coefficient matrix should also have zero as the augmented entry. Note that if there is no row of all zeros in the coefficient matrix, the system is automatically consistent. We can think of the rows that are identically zero as redundant and uninformative, and they can be discarded with no loss of information.
- Assuming the system is consistent, the "dimension" of the solution space is the number of freely varying parameters, which is equal to the number of variables that are *not* pivotal. Thus, we have:
  Dimension of solution space = (Number of columns) - (Number of rows that are not identically zero)
  Another way of putting it is that:
  Dimension of solution space = (Number of variables) - (Number of equations that actually give us new information)
- In the particular case that there are no rows that are identically zero in the coefficient matrix, we simply get:
  Dimension of solution space = (Number of variables) - (Number of equations)
  Note that this agrees with our general theoretical understanding.

2.3. **Converting a system into reduced row-echelon form.** Recall our three types of reversible operations on systems of linear equations:

(1) Multiply or divide an equation by a nonzero scalar.
(2) Add or subtract a multiple of an equation to another equation.
(3) Swap two equations.

These correspond to similar operations on the augmented matrix:

(1) Multiply or divide a row by a nonzero scalar.
(2) Add or subtract a multiple of a row to another row.
(3) Swap two rows.

Our *end goal* is to convert the system to reduced row-echelon form. Our *permissible moves* are the above three types of reversible operations. We now need to figure out a *general procedure* that, starting with any system, can identify the appropriate sequence of permissible moves to reach our end goal. The choice of moves will obviously depend on the system we start with. We want to make the process as algorithmically straightforward as possible, so that we ultimately do not have to think much about it.

The operations that we perform are done on the augmented matrix, but they are *controlled* by the coefficient matrix, in the sense that our decision of what operation to do at each stage is completely governned by the coefficient matrix. We do not ever actually need to look at the right-most column in the decision process for the operation.

Here is the procedure.

- We scan the first column, then the second column and so on till we find a column with a nonzero entry. As soon as we have discovered such a column, look for the top-most nonzero entry in that column. If that is not in the first row, then *swap* the first row with the row where it occurs. Note that we perform the swap in the augmented matrix, so we also move the entries of the right-most column. Recall that swapping equations (i.e., swapping rows) is a reversible transformation that preserves the set of solutions.
- We now have a situation of (possibly) some all-zero columns on the left edge, then a column where the top entry (first row) is nonzero. Now, if this top entry is not 1, *divide* that equation by the top entry. In the augmented matrix, this involves dividing the row by a nonzero number.
- What's left now is to clear all the nonzero stuff that may occur *below* the first row in that particular column. We can clear this stuff out by subtracting suitable multiples of the first row from each of the later rows.

Once we have done this clearing out, we ignore the first row, and look for the earliest column that has a nonzero entry in some row other than the first row. Note that that earliest column must be strictly later than the pivotal column for the first row. Having done so, swap it with the second row, then divide to make the pivotal entry 1. Now, clear all the nonzero entries in its colum below and above it by subtracting a suitable multiple of the second row from each of the other rows. Keep repeating a similar process. Explicitly:

- If the first i rows have been dealt with, and the pivotal column for the i^{th} row is the j^{th} column, we look for the first column with nonzero stuff below the i^{th} row (this column must have a number bigger than j). We swap the first row where there is a nonzero entry in that column (among those below the i^{th} row) with the $(i+1)^{th}$ row).
- We now divide out by the leading entry of that row to make it 1.
- Finally, we clear out all entries in the column of that row to 0s by subtracting a suitable multiple of the row.

This process, completed to the end, gives a reduced row-echelon form. If at any stage you get to the situation where a particular row of the coefficient matrix is all zeros, all later rows are all zeros. Reduced row-echelon form has been reached. If any of the corresponding augmented matrix column values is nonzero, then the system is inconsistent. If they are all zero, then the system is consistent, and these last few equations can be discarded.

Note that the column-scanning happens completely in the coefficient matrix, and the judgment of whether we have reached reduced row-echelon form is based completely on the coefficient matrix. However, the operations are also done to the last column of the augmented matrix, which is not part of the coefficient matrix. Note that the last column does not *control* any of the operations and does not affect how the operations are decided. In fact, we could determine the entire sequence of operations, and the final reduced row-echelon form of the coefficient matrix, even without knowledge of the last column. Then, when we get access to the last column (the outputs) we can quickly apply the sequence of operations to this to get the new system, that we then proceed to solve.

We will review in class the example from the book.

2.4. **Cleaning your room.** A full proof that this process works "as advertised" would be too tedious to write down, though it should be conceptually clear. We need to check that the strategy we have outlined above is a *fully general strategy* that is guaranteed to give us a reduced row-echelon form regardless of the system that we start with.

Imagine that you are cleaning your room. You clean one corner of the room. There is a lot of junk there, but you mostly just dump all the junk in another corner which you haven't yet gotten around to cleaning. This doesn't really make the rest of the room dirtier, because the rest of the room was pretty dirty to begin with.

Now, you gradually proceed to more corners of the room. At each stage, as you are cleaning that corner, you can afford to do anything to the parts of the room you have not yet gotten around to cleaning. But you should not dump any stuff in the part you have already cleaned. If you do so, then you will end up having to return and re-clean the part of the room you already cleaned. This is something that you want to avoid. If you can demonstrate that each part of the room you clean makes that part clean and preserves the cleanliness of earlier parts, then you are doing well.

The appropriate analogues to parts of the room in this case are columns of the matrix. In each iteration of the process, one or more columns (traversing from left to right) get "cleaned up." In the process of cleaning up these columns, a lot of changes are made to later columns. But we didn't have any guarantee of cleanliness for those columns. They were a mess anyways. The key point is that while cleaning up a column, *earlier columns* (the ones we already cleaned up) should not get sullied. This indeed is something you should check carefully (in order to get an understanding of why the process works; it's not necessary to check this while executing the process). It's easy to see that these earlier columns are unaffected by swapping and by scalar multiplication. They are also unaffected by the shears because the value being added or subtracted is zero for these columns.

2.5. **Gaussian elimination and row-echelon form.** Recall the three conditions that we used to declare a matrix to be in reduced row-echelon form:

(1) For every row, the left-most nonzero entry, if any, is a 1. We will call this the *pivotal* entry for the row and the column where this appears the pivotal column.
(2) The pivotal column for a later (lower) row is a later (more to the right) column, if it exists. If a given row is all zeros, all later rows are also all zeros.
(3) Any column containing a pivotal 1 must have all other entries in it equal to 0. This has two subparts:
    (a) All entries in the column below the pivotal 1 are equal to 0.
    (b) All entries in the column above the pivotal 1 are equal to 0.

We had mentioned a while back that condition 3(b) is less important. If we do not insist on condition 3(b), we will still end up with a "triangular" system from which the solutions can be constructed with relative ease. A matrix that satisfies conditions 1, 2, and 3(a) is said to be in *row-echelon form* (without the qualifier "reduced").

We can solve systems of linear equations by simply targeting to convert them to row-echelon form, rather than reduced row-echelon form. Some authors use the term *Gaussian elimination* for this process (as opposed to Gauss-Jordan elimination for converting to reduced row-echelon form). Gaussian elimination saves somewhat on the number of operations that needs to be done in row reduction, but a little extra effort needs to be spent constructing the solution. The main saving in row reduction happens at the stage where we are clearing the nonzero entries in the same column as a pivotal 1. With Gauss-Jordan elimination (working towards the *reduced* row-echelon form), we need to clear entries both below and above the 1. With Gaussian elimination (working towards the row-echelon form), we only need to clear entries below the 1.

3. Computational considerations

Recall from an earlier discussion that we evaluate algorithms using five broad criteria:

- *Time* is a major constraint. There's a lot more to life than solving equations. The faster, the better.
- *Memory* is another constraint. Manipulating large systems of equations can take a lot of space. Lots of memory usage can also affect time indirectly. Ideally, our equation-solving process should use as little "working memory" as possible.

- *Parallelizability* is great. Parallelizability means that if multiple processors are available, the task can be split across them in a meaningful manner that cuts down on the time required. Some kinds of procedures are parallelizable, or partially so. Others are not parallelizable.
- *Numerical precision* is another issue, particularly in cases where the numbers involved are not rational numbers. The degree of precision with which we measure solutions is important. Lots of major errors occur because of inappropriate truncation. Hypersensitivity to small errors is a bad idea.
- *Elegance and code simplicity*: Another criterion that is arguably important is the complexity of the code that powers the algorithm. Given two algorithms that take about the same amount of time to run, the algorithm that is based on easier, simpler code is preferable, since the code is easier to maintain, tweak, and debug. Elegance is also valuable for aesthetic reasons, which you may or may not care about.

We will now discuss Gaussian elimination and Gauss-Jordan elimination from the perspective of each of these.

3.1. **Time.** If you've tried doing a few problems using Gauss-Jordan elimination, time feels like a major constraint. Indeed, the process feels deliberately long and tedious. You might wonder whether there exist other *ad hoc* approaches, such as substituting from one equation into the other, that work better.

In fact, there is no known method that works better than Gaussian elimination in general (though it can be tweaked at the margins, and there are special matrix structures where shorter methods apply). The "substitute from one equation into another" approach ends up being computationally the same as Gaussian elimination. The main reason why Gaussian elimination is preferred is that it is programmatically easier to code row operations rather than actual manipulation rules for equations (as substitution rules would do).

We first try to compute the *number of row operations* involved in Gaussian elimination and Gauss-Jordan elimination. For simplicity, we will assume that we are working with an equal number of variables and equations n. We note that:

- There are n rows that we need to fix.
- For each row, the process for fixing that row involves a scanning (finding nonzero entries), an interchange operation (that is one row operation), a scalar multiplication (that is another row operation) and up to $n-1$ row operations that involve clearing out the nonzero entries in the pivotal column. This suggests an upper bound of $n+1$ row operations. In fact, the upper bound is n, because if there is an interchange operation, then one of the entries is already zero, so we do not need to clear that out.

Overall, then, we could require n^2 row operations. It is easy to construct an example matrix for every n where this upper bound is achieved.

Note that if we are doing Gaussian elimination rather than Gauss-Jordan elimination, the worst case for the number of row operations would be somewhat lower: $n + (n-1) + \cdots + 1 = n(n+1)/2$. This is about half the original number. However, the saving that we achieve does not change the number of row operations in the "big O notation" terms. The worst-case number of row operations remains $\Theta(n^2)$ in that notation.

Let us now proceed to the *arithmetic complexity* of Gauss-Jordan elimination. Arithmetic complexity is time complexity measured in terms of the total number of arithmetic operations involved, where an arithmetic operation includes addition, subtraction, multiplication, or division of two numbers at a time. Each of the three types of row operations involves at most $2n$ arithmetic operations, so since the row operation complexity is $\Theta(n^2)$, this gives an upper bound of $O(n^3)$ on the arithmetic complexity. It is also relatively easy to see that this is the best we can achieve order-wise, although the constant can be somewhat improved once we look at the actual details of the operations (namely, because the nature of our past operations guarantees some entries to already be zero, we can spare ourselves from doing those operations). Thus, the arithmetic complexity is $\Theta(n^3)$. Note that the complexity incurred from other bookkeeping and tracking operations does not affect the complexity estimate.

Similar complexity estimates are valid for systems where the number of rows differs from the number of columns.

Arithmetic complexity is, however, potentially misleading because, it is not true that "an addition is an addition." Addition of large, unwieldy integers is more time-consuming than addition of small integers.

Integer multiplication similarly becomes more complicated the larger the integers are. If dealing with non-integers, we have to worry about precision: the more decimal places we have, the more time-consuming the operations are (this also affects the precision/accuracy evaluation, which we will turn to later).

One of the problems is that even if the original matrix we start with has small integer entries, row reduction can lead to larger integer entries in magnitude, and repeatedly doing so can get us out of control. Consider, for instance, the matrix:

$$\begin{bmatrix} 1 & 3 \\ 3 & -1 \end{bmatrix}$$

The first row is already good, so we clear out the 3 in the first column that occurs below the 1. To do this, we subtract 3 times the first row from the second row. We get:

$$\begin{bmatrix} 1 & 3 \\ 0 & -10 \end{bmatrix}$$

Note that we ended up with a large magnitude integer -10, much bigger than the entries in the matrix we started with. The problem could keep getting compounded in bigger matrices. Further, note that converting leading entries to 1 by division also brings up the possibility of having to deal with rational numbers that are not integers, even if we started completely with integers.

There is a variation of Gauss-Jordan elimination called the *Bareiss algorithm* that does a good job of controlling the size of entries at all intermediate steps of the algorithm. The algorithm bounds the bit complexity of the algorithm to a polynomial in n and the bit length of the entries. Discussion of the algorithm in full is beyond the scope of this course. Nonetheless, we will explore some ideas related to this algorithm to help keep the coefficient matrix as close to integers as possible and make your life easier, even though we'll avoid the algorithm in full (see Section 4.4 for a few details).

3.2. **Space requirements.** As was the case with time, space requirements can be interpreted in two ways: the *arithmetic space requirement* and the *bit space requirement.* Arithmetically, at every stage, we need to store a matrix of the same size as the original augmented matrix, plus a much smaller amount of tracking information that helps us remember how far in the algorithm we have gotten. In other words, the space requirement for the algorithm for a situation with n variables and n equations is $\Theta(n^2)$.

Note that this assumes that space is *reusable.* If space is not reusable, then the space requirement would be more complicated.

The computation also becomes more complicated if we consider the issue of the numbers in the matrix becoming larger in magnitude or getting to involve fractions. The *bit complexity* estimate will therefore be somewhat higher, though it will still be polynomial in n and L, where L is an upper bound on the bit length of all the augmented matrix entries.

3.3. **Parallelizability.** The algorithm is partly parallelizable. Specifically, the process of clearing the nonzero entries in a column with a pivotal entry can be done in parallel: each row can be done separately. Moreover, within each row, the different entry subtractions can be done in parallel. In other words, we could be doing all the $O(n^2)$ operations corresponding to clearing out a pivotal column more or less simultaneously.

This parallelization is most useful in situations where read/write access to an always-updated augmented matrix is a lot cheaper than actually performing arithmetic operations on the entries. In other words, if the cost of having many different processors have read/write access to the matrix is much less than the time cost of doing an arithmetic operation, then this parallelization makes sense. With such parallelization, the total arithmetic complexity could become as low as $O(n)$. However, these assumptions are not typically realistic.

There are other ways of parallelizing which include the use of block matrices, but these involve modifications to the algorithm and are beyond the present scope.

3.4. **Precision and accuracy.** If we are working with rational entries we can maintain perfect precision and accuracy. This is because all entries in a process starting with rational entries will remain rational. If we wish, we could first convert the entries to integers and then use the Bareiss algorithm to avoid entries that become too complicated.

Precision becomes an issue when the entries are not rationals, but observed real numbers without closed-form expressions, and we are truncating them to a manageable level. The problem arises most for matrices where the leading entry in a row is very small in magnitude, so dividing out by it means multiplying by a huge number, thus amplifying any measurement error present in that and remaining entries. There are certain kinds of matrices for which this problem does not arise, and there are modifications of Gauss-Jordan elimination that (a) keep the problem in check, and (b) failing that, warn us in case it cannot be kept in check.

3.5. **Elegance and code simplicity.** Gauss-Jordan elimination is conceptually not too hard, albeit it is still somewhat tricky to code. The main advantage of coding Gauss-Jordan elimination as opposed to "substitute and solve" approaches, even though they are computationally equivalent, is that the latter approaches require coding symbolic manipulations of equations, which is trickier. Gauss-Jordan elimination, on the other hand, converts those symbolic manipulations of equations into addition and subtraction of numbers, and the algorithm does not require the computer to "understand" how to manipulate an equation.

3.6. **Amortization when multiple systems share the same coefficient matrix.** We discussed earlier that when the same coefficient matrix appears in multiple systems, with the final column (the outputs) of the systems not known in advance, it makes sense to "preprocess" the system by converting the coefficient matrix to reduced row-echelon form and recording all the row operation steps performed. Then, when given access to the output column, we can simply apply the operations step-by-step to that output column alone (the coefficient matrix having already been dealt with) and read off the solution.

The arithmetic complexity of the operations done on the output column alone is the number of row operations done in Gauss-Jordan elimination, which is $\Theta(n^2)$ assuming a worst-case scenario. Thus, the fixed pre-processing cost is $\Theta(n^3)$ but the marginal time cost for each new output is $\Theta(n^2)$.

4. Human computer tips

You're a human computer. Yes, you! `https://en.wikipedia.org/wiki/Human_computer`

In other words, you need to figure out how to do computations effectively and correctly.

The peril of being a human computer is that you are more liable to make careless errors. The advantage of being a human computer is that you can often rely on visual heuristics that mechanical computers cannot really understand. Let's use our advantages to help guard against the perils.

That said, to a large extent, the tips for smart computation and clear solution-writing (including how much detail to show) are closely linked with the tips for programming an appropriate algorithm for a computer to execute. Some of the ideas we discussed in the previous section will reappear in a somewhat different form here.

4.1. **Writing the operation clearly.** For every operation or sequence of operations that you do, write the operation or sequence clearly. In case of a sequence, write the operations in sequence. You may write the operation above a transition arrow, or on the right of the matrix where you are adding and subtracting. We'll see soon how much we can combine operations, and your approach to writing operations should take that into account.

One advantage of clearly writing the row operation steps is in terms of *pre-processing*: if we ever need to solve a linear system with the same coefficient matrix but a different augmenting column, we can easily see all the row operation steps and simply apply them to the new augmenting column, getting to the answer much more quickly than if we had not written down the steps.

4.2. **Should the steps be combined? Yes, as long as they are parallelizable.** One can consider two extremes in terms of how extensively you write your steps. At one extreme, you rewrite the entire augmented matrix *after every row operation*, stating the row operation used at the transformation at each stage. The problem with this is that you have to write a lot of stuff. Since there are $\Theta(n^2)$ row operations, you end up writing $\Theta(n^2)$ matrices. In other words, a lot of space gets used up.

At the other extreme, you just write one matrix with an erasable pencil, and keep erasing and updating entries each time you do a row operation. The problem with this is that it can make you feel giddy.

Is there a middle ground? Yes. The idea is to essentially follow the first approach, *except that we do not separately write steps for operations that are done in parallel.* Explicitly, this means that we can combine all the clearing steps for a single pivotal column together. Let us say that at some stage, our augmented matrix reads as:

$$\begin{bmatrix} 1 & 2 & 3 & \mid & 4 \\ 2 & 5 & 1 & \mid & 1 \\ 5 & 7 & 8 & \mid & -2 \end{bmatrix}$$

We need to clear out the entries in the first column below the pivotal 1. There are two operations we need to perform: $R2 \to R2 - 2R1$ (i.e., we sutract twice the first row from the second row), and $R3 \to R3 - 5R1$ (i.e., we subtract 5 times the first row from the third row). We can directly write down the augmented matrix obtained after *both* these operations are performed, instead of writing down the full new augmented matrix after one operation and then again the one obtained after the second. Note that you should also write down clearly *all the row operations that are being done simultaneously*, so in this case, you should write down $R2 \to R2 - 2R1$ and $R3 \to R3 - 5R1$ (some people write these expressions on the immediate right of the matrix, some write then above the transtion arrow).

The reason why parallel steps can be combined is that doing so does not involve any erasing and rewriting, since the steps do not influence one another. On the other hand, a sequence such that $R2 \to R2 - 2R1$ followed by $R1 \to R1 - R2$ should be done step-by-step because the second step is dependent on the outcome of the first one. Note that with this approach, you only end up writing $\Theta(n)$ matrices, and about $\Theta(n^3)$ numbers, which makes sense because $\Theta(n^3)$ is the arithmetic complexity of the procedure.

In conceptual terms, parallelizable operations can be done together because they commute with each other. It does not matter in what order we do the operations. For instance, in the above example, the operations $R2 \to R2 - 2R1$ and $R3 \to R3 - 3R1$ yield the same final result regardless of the order in which they are performed.

4.3. **Noting excellent rows to swap to the top.** A literal reading of the Gauss-Jordan algorithm would suggest that we look for the *earliest* row with a nonzero entry in the relevant column. However, it typically makes more sense to look at all the rows and pick the one that's most likely to minimize further computation. A good row is one where the leading entry is as close to 1 as possible already, and as many of the other entries are 0 (and if not 0, at least small) as possible. If you find that type of row, swap that to the top.

For instance, consider the augmented matrix:

$$\begin{bmatrix} 4 & 7 & 5 & \mid & 12 \\ 5 & 4 & 3 & \mid & 10 \\ 1 & 0 & 2 & \mid & 3 \end{bmatrix}$$

One can start blindly here: divide the first row by 4, and proceed to clear the remaining entries in the first column. Alternatively, one can proceed smartly: interchange the first and third row. Note how much easier the remaining row clearing operations become if we use the latter approach.

Of course, sometimes, all the good stuff doesn't come together, and in that case, one has to prioritize (for instance, one row might be mostly zeros but its leading entry may be large, another row might lead with a 1 but have messy later entries). The goal here isn't to come up with a foolproof way to avoid further work. Rather, the goal is to identify a few methods that might on occasion save some precious time.

4.4. **Keeping the entries as close to integers as possible.** Yes, the official version of Gauss-Jordan elimination tells you to divide and make the pivotal entry 1. In practice, this can cause a headache. It's generally advisable to use other techniques. Did you know that, in practice, one can dispense of division almost entirely in the quest to get a pivotal 1? That does entail doing more subtraction steps. But it's often worth it, if it means we can keep everything integral. This is the secret of the Bareiss algorithm, but we'll just catch a glimpse here.

For instance, consider the system:

$$\begin{aligned} 3x + y &= 4 \\ 5x + 7y &= 44 \end{aligned}$$

$$\begin{bmatrix} 3 & 1 & \mid & 4 \\ 5 & 7 & \mid & 44 \end{bmatrix}$$

The standard Gauss-Jordan method would involve dividing the first row by 3. Let's try to do it another way.

We first do $R2 \to R2 - R1$, and obtain:

$$\begin{bmatrix} 3 & 1 & \mid & 4 \\ 2 & 6 & \mid & 40 \end{bmatrix}$$

We now do $R1 \to R1 - R2$, and obtain:

$$\begin{bmatrix} 1 & -5 & \mid & -36 \\ 2 & 6 & \mid & 40 \end{bmatrix}$$

Note that we have managed to make the column entry 1 without doing any division. We now do $R2 \to R2 - 2R1$ and obtain:

$$\begin{bmatrix} 1 & -5 & \mid & -36 \\ 0 & 16 & \mid & 112 \end{bmatrix}$$

Now this R2 can be divided by 16 (we have no real choice) but luckily everything remains integral, and we obtain:

$$\begin{bmatrix} 1 & -5 & \mid & -36 \\ 0 & 1 & \mid & 7 \end{bmatrix}$$

We finally do $R1 \to R1 + 5R2$ and obtain:

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

Thus, $x = -1$ and $y = 7$.

The puzzling part of the process is the first two subtractions. How did we think of them? The secret sauce is something called the *gcd algorithm* (this is the algorithm used to compute the gcd of two numbers): keep applying the Euclidean algorithm to find quotients and remainders. If you wish to learn how that works, feel free to read it up. Otherwise, just try to get an intuitive sense of how to do it in easy cases.

4.5. **Keeping a number sense.** We mentioned above that dealing with numbers makes things a lot easier to code than handling full-scale algebraic manipulations. Note, however, that what's simple for computers need not be simple for humans, particularly humans who understand algebra. I've often observed that people can sometimes make fewer careless errors when solving systems of linear equations using symbolic manipulation of the equation (i.e., writing down the full equations and adding, subtracting, and substituting) rather than performing row operations on the augmented matrix, even though the row operations on the matrix are tantamount to equation manipulation. This may have something to do with the degree of natural structural understanding and care that people exhibit: when manipulating equations, you have more of a sense of whether what you're doing is right or wrong. When adding and subtracting numbers, the process feels more mechanical and you may therefore be less attentive. Ideally, you'd want the best of both worlds: just do row operations on the matrices, but still have a "number sense" that alerts you to when things are going wrong.

Perhaps the best way to achieve this is to periodically "make sense" of the numerical operations that we performend in terms of the symbolic algebra. In addition, the tips for good presentation and avoiding careless errors given above should help.

LINEAR SYSTEMS AND MATRIX ALGEBRA

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 1.3.

Executive summary

(1) The *rank* of a matrix is defined as the number of nonzero rows in its reduced row-echelon form, and is also equal to the number of leading variables. The rank of a matrix is less than or equal to the number of rows. It is also less than or equal to the number of columns.
(2) The rank of the coefficient matrix of a system of simultaneous linear equations describes the number of independent equational constraints in the system.
(3) How far the coefficient matrix is from having full column rank determines the dimension of the solution space if it exists.
(4) How far the coefficient matrix is from having full row rank determines the probability that the system is consistent, roughly speaking. If the coefficient matrix has full row rank, then the system is consistent for all outputs. Otherwise, it is consistent only for some outputs and inconsistent for others.
(5) For a consistent system, the dimension of the solution space equals (number of variables) - (rank).
(6) There is a concept of "expected dimension" which is (number of variables) - (number of equations). Note that if the system does not have full row rank, the expected dimension is less than the actual dimension (if consistent). The expected dimension can be thought of as averaging the actual dimensions over all cases, where inconsistent cases are assigned dimensions of $-\infty$. This is hard to formally develop, so we will leave this out.
(7) There are various terms commonly associated with matrices: zero matrix, square matrix, diagonal, diagonal matrix, scalar matrix, identity matrix, upper-triangular matrix, and lower-triangular matrix.
(8) A vector can be represented as a row vector or a column vector.
(9) We can define a dot product of two vectors and think of it in terms of a sliding snake.
(10) We can define a matrix-vector product: a product of a matrix with a column vector. The product is a column vector whose entries are dot products of the respective rows of the matrix (considered as vectors) with the column vector.
(11) Matrix-vector multiplication is linear in the vector.
(12) A linear system of equations can be expressed as saying that the coefficient matrix times the input vector column (this is the column of unknowns) equals the output vector column (this is the column that would be the last column in the augmented matrix).

1. Rank of a matrix

The rank of the coefficient matrix is an important invariant associated with a system of linear equations.

1.1. **Rank defined using reduced row-echelon form.**

If a matrix is already in reduced row-echelon form, its rank is defined as the number of leading variables, or equivalently, as the number of rows that are not all zero.

For an arbitrary matrix, we first convert it to reduced row-echelon form, then measure the rank there. Although we didn't establish it earlier, the reduced row-echelon form is unique, hence the rank defined this way makes sense and is also unique.

Note that when we used row reduction as a process to solve systems of linear equations, we had a coefficient matrix as well as an augmented matrix. The operations were determined by the coefficient matrix, but we performed them on the augmented matrix. For our discussion of rank here, we are concerned *only* with the coefficient matrix part, not with the augmenting column.

Note that:

Rank of a matrix $\leq$ Total number of rows in the matrix

Also:

Rank of a matrix $\leq$ Total number of columns in the matrix

Combining these, we obtain that:

Rank of a matrix $\leq$ Minimum of the total number of rows and the total number of columns in the matrix

Conceptually, the rank of the coefficient matrix signifies the number of independent equational constraints in there. The above two statements are interpreted as saying:

Number of independent equational constraints $\leq$ Total number of equational constraints

Number of independent equational constraints $\leq$ Total number of variables

The former makes obvious sense, and the latter makes sense because (in case the system is consistent) the dimension of the solution space is the difference (Total number of variables) - (Number of independent equational constraints).

Combining these, we obtain that:

Number of independent equational constraints $\leq$ Minimum of the number of variables and number of equations

A couple more terms:

- A matrix is said to have *full row rank* if its rank equals its number of rows. If the coefficient matrix of a system of simultaneous linear equations has full row rank, this is equivalent to saying that all the equations are independent, i.e., the system of equations has no redundancies and no inconsistencies.
- A matrix is said to have *full column rank* if its rank equals its number of columns. If the coefficient matrix of a system of simultaneous equations has full column rank, this is equivalent to saying that there are as many constraints as variables. Conceptually, this means that, *if consistent*, the system is precisely determined, and there is a unique solution. (Note: one could still have inconsistent due to zero rows in the coefficient matrix with a nonzero entry in the augmenting column; full column rank does not preclude this).
- A square matrix (i.e., a matrix with as many rows as columns) is said to have *full rank* if its rank equals its number of rows and also equals its number of columns.

1.2. **Playing around with the meaning of rank.** Converting a matrix to reduced row-echelon form (using Gauss-Jordan elimination) in order to be able to compute its rank seems tedious, but it is roughly the quickest way of finding the rank in general. If the goal is just to find the rank, some parts of the process can be skipped or simplified. For instance, we can use Gaussian elimination to get the row-echelon form instead of using Gauss-Jordan elimination to get the *reduced* row-echelon form. There do exist other algorithms that are better suited to finding the rank of a matrix, but now is not the time to discuss them.

The only way a matrix can have rank zero is if all the entries of the matrix are zero. The case of matrices of rank one is more interesting. Clearly, if there is only one nonzero row in the matrix, then it has rank one. Another way a matrix can have rank one is if all the nonzero rows in the matrix are scalar multiples of each other. Consider, for instance, the matrix:

$$\begin{bmatrix} 1 & 2 & 5 \\ 3 & 6 & 15 \end{bmatrix}$$

Note that each entry of the second row is three times the corresponding column entry in the first row. If the coefficient matrix of a linear system looks like this, the coefficient part of the second equation is a carbon copy of the first, with a tripling smudge. If we think in terms of row reduction, then subtracting 3 times the first equation from the second equation gives us:

$$\begin{bmatrix} 1 & 2 & 5 \\ 0 & 0 & 0 \end{bmatrix}$$

Note now that the second row is completely zero. This is because, when we got rid of the multiple of the first row, nothing survived.

This means that if we have a system of equations:

$$\begin{aligned} x + 2y + 5z &= a \\ 3x + 6y + 15z &= b \end{aligned}$$

then the system will either be redundant (this happens if $b = 3a$) or inconsistent (this happens if $b \neq 3a$). In the former case, since we effectively have only one constraint, the solution set is two-dimensional (using $3 - 1 = 2$). In the latter case, the solution set is empty.

1.3. **Row rank and its role in determining consistency.** If the coefficient matrix has *full row rank*, then the system of linear equations is consistent. This can be seen by looking at the reduced row-echelon form.

On the other hand, if the coefficient matrix does not have full row rank, this means that applying Gauss-Jordan elimination makes one or more of its rows zero. For each of the zero rows in the coefficient matrix, consider the corresponding output value (the augmented matrix entry). If that is zero, all is well, and the constraint is essentially a no-information constraint. It can be dropped. If, on the other hand, the output value is nonzero, then that equation gives a contradiction, which means that the system as a whole is inconsistent, i.e., it has no solutions.

The upshot is that, if the coefficient matrix does not have full row rank, there are two possibilities:

- After doing Gauss-Jordan elimination, the augmented matrix entries for all the zero rows in the coefficient matrix are zero. In this case, the system is consistent. We can discard all the zero rows and solve the system comprised of the remaining rows. The dimension of the solution space equals the number of non-leading variables, as usual.
- After doing Gauss-Jordan elimination, the augmented matrix entries for at least one of the zero rows is nonzero. In this case, the system is inconsistent, or equivalently, the solution space is empty.

This means that if the coefficient matrix does not have full row rank, then the consistency of the system hinges on the last column of the augmented matrix.

1.4. **Column rank and its role in determining solution space dimensions.** The relation between the rank and the number of columns affects the dimension of the solution space assuming the system is consistent. As noted above, if the system is consistent, we can discard the zero rows and get that the dimension of the solution space equals the number of non-leading variables. In other words:

Dimension of solution space = Number of non-leading variables = (Number of variables) - (Rank) = (Number of columns) - (Rank)

1.5. **Summary.** The following is a useful summary of the state-of-the-art in terms of our understanding of how the coefficient matrix and augmented matrix control the nature of the set of solutions to a linear system:

(1) If the coefficient matrix of a linear system has full row rank, then the system is consistent regardless of the output column. The dimension of the solution space is given as (number of variables) - (number of equations).
(2) If the coefficient matrix of a linear system has full column rank, then the system is either inconsistent or has a unique solution.
(3) If the coefficient matrix of a linear system with an equal number of variables and equations has full row rank and full column rank, then the system has a unique solution.
(4) If the coefficient matrix of a linear system does not have full row rank, then the system may or may not be consistent. Whether it is consistent or not depends on the nature of the output column. If the output column is such that after converting to rref all the zero rows give zero outputs, the system is consistent. Otherwise, it is inconsistent. Conditional to the system being consistent, the dimension of the solution space is (number of variables) - (rank).
(5) We can capture these by a concept of "expected dimension" of a system. Define the expected dimension of a system as (number of variables) - (number of equations). In case of full row rank, the expected dimension equals the actual dimension. In case the row rank is not full, the actual dimension is not the same as the expected dimension. Either the solution set is empty or it has

dimension (number of variables) - (rank), which is higher than the expected dimension. The expected dimension can be thought of as a kind of "averaging out" of the actual dimensions over all possible output columns, where we assign an empty solution set a dimension $-\infty$.

2. Matrices and vectors: some terminology

2.1. **The purpose of the terminology.** We know that a large part of the behavior of a linear system is controlled by its coefficient matrix. Therefore, we should ideally be able to look at the coefficient matrix, which *prima facie* is just a collection of numbers, and understand the story behind those numbers. In order to do that, it helps to develop some terminology related to matrices.

2.2. **Matrices and their entries.** A $m \times n$ matrix is a matrix with m rows and n columns. Typically, we use subscripts for the matrix entries with the first subscript denoting the row number and the second subscript denoting the column number. For instance, a matrix $A = (a_{ij})$ would mean that the entry a_{ij} describes the entry in the i^{th} row and j^{th} column. Explicitly, the numbering for a 2×3 matrix is as follows:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Note that a $m \times n$ matrix and a $n \times m$ matrix look different from each other. The roles of rows and columns, though symmetric in a cosmic sense, are *not* symmetric at the level of nitty-gritty computation. For many of the ideas that we will discuss in the near and far future, it is *very important to get clear in one's head whether we're talking, at any given instant, about the number of rows or the number of columns.*

2.3. **Vectors as row matrices, column matrices, and diagonal matrices.** A vector is a tuple of numbers (with only one dimension of arrangement, unlike matrices, that use two dimensions to arrange the numbers). A vector could be represented as a matrix in either of two standard ways. It could be represented as a *row* vector, where all the numbers are written in a row:

$$\begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

Note that a vector with n coordinates becomes a $1 \times n$ matrix when written as a row vector.

It could also be written as a *column* vector:

$$\begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_n \end{bmatrix}$$

Note that a vector with n coordinates becomes a $n \times 1$ matrix when written as a column vector.

There is a third way of writing vectors as matrices, which may strike you as a bit unusual right now, but is particularly useful for certain purposes. This is to write the vector as a *diagonal* matrix. Explicitly, a vector with coordinates $a_1, a_2, \dots, a_n$ is written as the matrix:

$$\begin{bmatrix} a_1 & 0 & 0 & \dots & 0 \\ 0 & a_2 & 0 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \dots & 0 & a_n \end{bmatrix}$$

See the below section for the definition of diagonal matrix.

2.4. **Square matrices, diagonal matrices, and upper/lower-triangular matrices.** A matrix is termed a *square* matrix if it has an equal number of rows and columns.

The following is some terminology used in the context of square matrices.

- A matrix is termed the *zero* matrix if all its entries are zero.
- The *diagonal* (also called the main diagonal or principal diagonal) of a square matrix is the set of positions (and entries) where the row number and column number are equal. The diagonal includes the top left and the bottom right entry. In symbols, the diagonal entries are the entries a_{ii}.
- A square matrix is termed a *diagonal* matrix if all the entries that are *not* on the diagonal are zero. The entries on the diagonal may be zero or nonzero. In symbols, a matrix (a_{ij}) is diagonal if $a_{ij} = 0$ whenever $i \neq j$.
- A square matrix is termed the *identity* matrix if it is a diagonal matrix and all diagonal entries are equal to 1.
- A square matrix is termed a *scalar* matrix if it is a diagonal matrix and all the diagonal entries are equal to each other.
- A square matrix is termed an *upper triangular matrix* if all the entries "below" the diagonal, i.e., entries of the form $a_{ij}, i > j$, are equal to zero. It is termed a *strictly upper triangular matrix* if the diagonal entries are also zero.
- A square matrix is termed a *lower triangular matrix* if all the entries "above" the diagonal, i.e., entries of the form $a_{ij}, i < j$, are equal to zero. It is termed a *strictly lower triangular matrix* if the diagonal entries are also zero.

Note that a diagonal matrix can be defined as a matrix that is both upper and lower triangular.

2.5. **The matrix and the equational system: diagonal and triangular matrices.** A while back, when describing equation-solving in general, we had discussed *diagonal systems* (where each equation involves only one variable, and the equations involve different variables). We also described *triangular systems* (where there is one equation involving only one variable, another equation involving that and another variable, and so on).

We can now verify that:

- A system of simultaneous linear equations is a diagonal system if and only if, when the equations and variables are arranged in an appropriate order, the coefficient matrix is a diagonal matrix.
- A system of simultaneous linear equations is a triangular system if and only if, when the equations and variables are arranged in an aprpopriate order, the coefficient matrix is an upper triangular matrix, and if and only if, for another appropriate order, the coefficient matrix is a lower triangular matrix. The *upper triangular* case means that we have to solve for the last variable first (using the last equation), then solve for the second last variable (using the second last equation) and so on. The *lower triangular* case means that we have to solve for the first variable first (using the first equation), then solve for the second variable (using the second equation), and so on.

  Note that the description we gave for Gaussian elimination is similar to the upper triangular description, in the sense that if the system has full row rank and full column rank, then Gaussian elimination (*not* the full Gauss-Jordan elimination) will yield an upper triangular matrix.

  The upper triangular and lower triangular cases are conceptually equivalent. However, to maintain clarity and consistency with the way we formulated Gaussian elimination, we will describe future results in the language of the upper triangular case.

2.6. **Facts about ranks.** The following facts about matrices are useful to know and easy to verify:

- Any diagonal matrix with all diagonal entries nonzero has full rank. More generally, for a diagonal matrix, the rank is the number of nonzero diagonal entries.
- Any nonzero scalar matrix has full rank.
- Any upper triangular or lower triangular matrix where all the diagonal entries are nonzero has full rank. More generally, for an upper triangular or lower triangular matrix, the rank is the number of nonzero diagonal entries.

3. Dot product, matrix-vector product, and matrix multiplication

3.1. **Structure follows function: creating algebraic rules that capture our most common manipulation modes.** One of our main goals is understanding functional forms that are linear (which may mean linear in the variables or linear in the parameters, depending on what our purpose is), and an intermediate and related goal is solving systems of simultaneous lineat equations that arise when trying to determine the inputs or the parameters. As we have seen, the structure of such systems is controlled by the nature of the coefficient matrix. There are typical modes of manipulation associated with understanding these systems of equations.

We want to build an algebraic structure on the collection of matrices such that the operations for that structure reflect the common types of manipulation we need to do with linear systems. Once we have defined the operations, we can study the abstract algebraic properties of these operations, and use that information to derive insight about what we're ultimately interested in: modeling and equations.

If you do *not* keep the connection closely in mind, the matrix operations feel very arbitrary, like *ad hoc* number rules. And these rules won't help you much because you won't understand *when* to do *what* operation. With that said, keep in mind that sometimes we will not be able to explain the connection immediately. So there may be a brief period in time where you really have to just store it as a formal definition, before we explore what it really means. But make sure it's a brief period, and that you do understand what the operations means.

3.2. **The dot product as linear in both sets of variables separately.** Consider variables $a_1, a_2, \dots, a_m$ and separately consider variables $b_1, b_2, \dots, b_m$. The dot product of the vectors $\langle a_1, a_2, \dots, a_m \rangle$ and $\langle b_1, b_2, \dots, b_m \rangle$ is denoted and defined as follows:

$$\langle a_1, a_2, \dots, a_m \rangle \cdot \langle b_1, b_2, \dots, b_m \rangle = \sum_{i=1}^{m} a_i b_i$$

The expression for the dot product is jointly linear in all the a_is. It is also jointly linear in all the b_is. It is, however, not linear in the a_is and b_is together (recall the earlier distinction between affine linear and affine multilinear) because each a_i interacts with the corresponding b_i through multiplication.

The dot product can be used to model the relationship between the variables and the coefficients in a homogeneous linear function (here, "homogeneous" means the intercept is zero) and more generally in the homogeneous part of a linear function.

3.3. **Row and column notation for vectors and thinking of the dot product as a sliding snake.** We can think of the dot product of two vectors as follows:

$$\begin{bmatrix} a_1 & a_2 & \dots & a_m \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{bmatrix}$$

We can think of a "sliding snake" that moves along the row for the entity being multiplied on the left and moves along the column for the entity being multiplied on the right.

We can now use this to define matrix-vector multiplication of a $n \times m$ matrix with a $m \times 1$ (i.e., a column matrix with m entries). Denote by a_{ij} the entry :

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{bmatrix}$$

The product is now no longer a number. Rather, it is a column matrix, given as:

$$\begin{bmatrix} \sum_{i=1}^m a_{1i} b_i \\ \sum_{i=1}^m a_{2i} b_i \\ \cdot \\ \cdot \\ \cdot \\ \sum_{i=1}^m a_{ni} b_i \end{bmatrix}$$

Basically, for each entry of the product (a column vector) we multiply the corresponding row in the matrix with the column vector.

NOTE: VERY IMPORTANT: The vector being multiplied is a $m \times 1$ column vector and the result of the multiplication is a $n \times 1$ column vector. Note the potential difference in the number of coordinates between the vector of b_is and the output vector. It's very important to have a sense for these numbers. Eventually, you'll get it. But the earlier the better. It's not so important which letter is m and which is n (the roles of the letters could be interchanged) but the relative significance within the context is important. Also, note that the $n \times m$ matrix starts with a $m \times 1$ input and gives a $n \times 1$ output. In other words, the matrix dimensions are (output dimension) X (input dimension). The number of rows is the output dimension, and the number of columns is the input dimension. Make sure you understand this *very clearly* at the procedural level.

3.4. **Matrix multiplication: a teaser preview.** Let A and B be matrices. Denote by a_{ij} the entry in the i^{th} row and j^{th} column of A. Denote by b_{jk} the entry in the j^{th} row and k^{th} column of B.

Suppose the number of columns in A equals the number of rows in B. In other words, suppose A is a $m \times n$ matrix and B is a $n \times p$ matrix. Then, AB is a $m \times p$ matrix, and if we denote it by C with entries c_{ik}, we have the formula:

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$$

The sliding snake visualization of the rule continues to work. For the ik^{th} entry of the product matrix C that we need to fill, we have a sliding snake that is moving horizontally along the i^{th} row of A and simultaneously moving vertically along the k^{th} column of B.

The previous cases of dot product (vector-vector multiplication) and matrix-vector multiplication can be viewed as special cases of matrix-matrix multiplication.

We will return to a more detailed treatment of matrix multiplication a week or so from now.

3.5. **Linearity of multiplication.** Matrix-vector multiplication is linear in terms of both the matrix and the vector (this is part of a more general observation that matrix multiplication is linear in terms of both the matrices involved, but that'll have to wait for later). Explicitly, the following are true for a $n \times m$ matrix A:

- For any m-dimensional vectors $\vec{x}$ and $\vec{y}$ (interpreted as column vectors), we have $A(\vec{x}+\vec{y}) = A\vec{x}+A\vec{y}$. The addition of vectors is done coordinate-wise. Note that the addition of $\vec{x}$ and $\vec{y}$ is addition of vectors both with m coordinates. The addition of $A\vec{x}$ and $A\vec{y}$ is addition of vectors with n coordinates.
- For any m-dimensional vector $\vec{x}$ (interpreted as a column vector) and any scalar λ, we have $A(\lambda\vec{x}) = \lambda(A\vec{x})$. Here, the scalar-vector product is defined as multiplying the scalar separately by each coordinate of the vector.

Again, keep in mind that the roles of m and n could be interchanged in this setting. But if you're interchanging them, *interchange them throughout*. The letters aren't sacred, but the relative roles of input and output are.

3.6. **Linear system as a statement about a matrix-vector product.** Any system of linear equations can be expressed as:

(Coefficient matrix)(Vector of unknowns) = (Vector of outputs (this would be the last column in the augmented matrix))

If the vector of unknowns is denoted $\vec{x}$, the coefficient matrix is denoted A, and the vector of outputs is denoted $\vec{b}$, we write this as:

$$A\vec{x} = \vec{b}$$

Here, both A and $\vec{b}$ are known, and we need to find $\vec{x}$.

What we would ideally like to do is write:

$$\vec{x} = \frac{\vec{b}}{A}$$

Does this make sense? Not directly. But it makes approximate sense. Stay tuned for more on this next week.

HYPOTHESIS TESTING, RANK, AND OVERDETERMINATION

MATH 196, SECTION 57 (VIPUL NAIK)

Executive summary

Words ...

(1) In order to test a hypothesis, we must conduct an experiment that could conceivably come up with an outcome that would falsify the hypothesis. This relates to Popper's notion of falsifiability.
(2) In the setting where we use a model with a functional form that is linear in the parameters, the situation where the coefficient matrix (dependent on the inputs) does *not* have full row rank is the situation where we can use consistency of the system to obtain additional confirmation of the hypothesis that the model is correct. If the coefficient matrix has full column rank, we can determine the parameters uniquely assuming consistency. The ideal situation would be that we choose inputs such that the coefficient matrix has full column rank but does not have full row rank. In this situation, we can obtain verification of the hypothesis *and* find the parameter values.
(3) In order to test a hypothesis of a function of multiple variables being affine linear, we could choose three points that are collinear in the input space and see if the outputs behave as predicted. If they do, then this is evidence in favor of linearity, but it is not conclusive evidence. If they do not, this is conclusive evidence against linearity.
(4) If the goal is to find the coefficients rather than to test the hypothesis of linearity, we should try picking independent inputs (in general, as many inputs as the number of parameters, which, for an affine linear functional form, is one more than the number of variables). Thus, the choice of inputs differ for the two types of goals. If, however, we are allowed enough inputs, then we can both find all the coefficients *and* test for linearity.

1. Testing the hypothesis of linearity: the case of functions of one variable

We begin by understanding all the ideas in the context of a function of one variable. We will then proceed to look at functions of more than one variable.

1.1. Preliminary steps.

Suppose f is a function of one variable x, defined for all reals. Recall that f is called a *linear* function (in the affine linear, rather than the homogeneous linear, sense) if there exist constants m and c such that $f(x) = mx + c$. Suppose the only thing you have access to is a black box that will take an input x and return the value $f(x)$. You have no information about the inner working of the program.

You do not know whether f is a linear function, but you believe that that may be the case.

(1) Suppose you are given the value of f at one point. This is useful information, but does not, in and of itself, shed any light on whether f is linear.
(2) Suppose you are given the values of f at two points. This would be enough to determine f uniquely under the assumption that it is linear. However, it will not be enough to provide evidence in favor of the hypothesis that f is linear. Why? Because *whatever* the values of the function, we can always fit a straight line through them.
(3) Suppose you are given the values of f at three points. This allows for a serious test of the hypothesis of linearity. There are two possibilities regarding how the three points on the graph of f look:
   - The points on the graph are non-collinear: This is *conclusive* evidence *against* the hypothesis of linearity. We can definitely *reject* the hypothesis that the function is linear.
   - The points on the graph are collinear: This is evidence *in favor of* the hypothesis of linearity, but it is not conclusive. We can imagine a function that is not linear but such that the outputs

for the three specific inputs that we chose happened to fall in a straight line. As we will discuss below, there are good reasons to treat the collinearity of points on the graph as *strong evidence* in favor of linearity. But it is not conclusive. For instance, for the function $f(x) = x^3 + x$, the points $x = -1$, $x = 0$, and $x = 1$ appear to satisfy the collinearity condition, despite the function not being linear.

(4) Suppose you are given more than three points and the values of f at all these points. This allows for an even stronger test of the hypothesis of linearity. There are two possibilities regarding how the corresponding points on the graph of the function look:
- The points on the graph are not all collinear: This is *conclusive* evidence *against* the hypothesis of linearity. We can definitely *reject* the hypothesis that the function is linear.
- The points on the graph are collinear: This is evidence *in favor of* the hypothesis of linearity, but is not conclusive. After all, we can draw curves that are not straight lines to fill in the unknown gaps between the known points on the graph. However, it does constitute strong evidence in favor of the linearity hypothesis. And the more points we have evaluated the function at, the stronger the evidence.

1.2. **Popper's concept of falsifiability.** Karl Popper, a philosopher of science, argued that scientific knowledge was built on the idea of *falsifiability.* A theory is falsifiable if we can envisage an experiment with a possible outcome that could be used to definitively show the theory to be false. Popper took the (arguably extreme) view that:

- A scientific theory *must* be falsifiable, i.e., until we come up with a possible way to falsify the theory, it isn't a scientific theory.
- A scientific theory can never be demonstrated to be true. Rather, as we construct more and more elaborate ways of trying to falsify the theory, and fail to falsify the theory for each such experiment, our confidence in the theory gradually increases. We can never reach a stage where we are *absolutely* sure of the theory. Rather, we become progressively more and more sure as the theory resists more and more challenges.

Popper's stringent criteria for what theories count as scientific have been subjected to much criticism. We do not intend to take a side in the debate. Rather, we will make the somewhat more elementary point at the heart of Popper's reasoning: *an experimental outcome can be viewed as evidence in favor of a theory only if there was some alternative outcome that would have caused one to reject (definitely or probabilistically) the theory.* Another way of putting this is that if the opposition to a theory isn't given a serious chance to make its case, the theory cannot be declared to have won the debate.[1] Further, the fairer and more elaborate the chance that is given to the opposition, the more this should boost confidence in our theory. Also, for any theory that makes assertions about infinite sets where only finitely many things can be checked at any given time, the asymmetry alluded to above exists: it may be possible to falsify the theory, but one can never be absolutely sure of the truth of the theory, though as we check more and more points and continue to fail to falsify the theory, our confidence in the theory improves.

Consider the preceding example where we are testing the hypothesis that a given function of one variable is linear. Replace the term "theory" by the term "hypothesis" in the preceding paragraph to better understand the subsequent discussion. Our hypothesis on the function f is that f is linear. The "experiments" that we can perform involve collecting the values of f at finitely many known inputs.

As per the preceding discussion:

(1) If we evaluate f at only one input, the information at hand is insufficient to come to any conclusions regarding whether f is linear. Another way of putting it is that we have had no opportunity to falsify the linearity hypothesis with just one point.

(2) The same holds if we evaluate f at only two inputs.

(3) If we evaluate f at three inputs, we have a serious opportunity to falsify the linearity hypothesis. Thus, this is the first serious test of the linearity hypothesis. Two cases:
- The points on the graph are non-collinear: This definitively falsifies the linearity hypothesis.

[1]You could argue that there are cases where the opposition doesn't exist at all, and the theory is *prima facie* true. But insofar as there is doubt about the truth of the theory, the point made here stands.

- The points on the graph are collinear: This is *consistent* with the linearity hypothesis, hence does not falsify it. It is therefore evidence in favor of the linearity hypothesis (note that it is evidence in favor because there was *potential* for the evidence to go the other way).

(4) Similar remarks apply to the evaluation of f at more than three points. Two cases:
- The points on the graph are not all collinear: In this case, the linearity hypothesis is definitively rejected, i.e., falsified.
- The points on the graph are all collinear: This is consistent with the linearity hypothesis, and can be construed as evidence in favor of the hypothesis. The greater the number of points, the stronger the evidence in favor of the linearity hypothesis.

1.3. **Interpretation of the above in terms of rank of a linear system.** When fitting a linear model $f(x) = mx + c$ using input-output pairs, we use the input-output pairs to generate a system of simultaneous linear equations in terms of the parameters m and c (these parameters become our "variables" for the purpose of solving the system). Explicitly, for each input-output pair (x_i, y_i) (with $y_i = f(x_i)$) we get a row of the following form in the augmented matrix:

$$\begin{bmatrix} 1 & x_i & \mid & y_i \end{bmatrix}$$

where the first column corresponds to the parameter c and the second column corresponds to the parameter m. Note that we choose the first column for c because using this ordering makes the process of computing the reduced row-echelon form easier. The row for the coefficient matrix reads:

$$\begin{bmatrix} 1 & x_i \end{bmatrix}$$

We can now formulate the earlier results in terms of ranks of matrices:

(1) If we evaluate f at only one input, the information at hand is insufficient to come to any conclusions regarding whether f is linear. It is also insufficient to determine f even assuming f is linear. Another way of putting it is that we have had no opportunity to falsify the linearity hypothesis with just one point. If the input-output pair is (x_1, y_1), we obtain the augmented matrix:

$$\begin{bmatrix} 1 & x_i & \mid & y_1 \end{bmatrix}$$

The rank of the coefficient matrix is 1. Note that the system has *full row rank* and is therefore consistent by definition (therefore, it cannot be used to falsify the hypothesis of linearity). The system does not have full column rank, so it does not give a unique solution even if we assume linearity.

(2) If we evaluate f at two inputs, we obtain f uniquely subject to the assumption that it is linear, but we do not obtain any verification of the linearity of f. Explicitly, if (x_1, y_1) and (x_2, y_2) are the input-output pairs, then the augmented matrix (with the first column for the variable c and the second column for the variable m; remember that we use this ordering to make the matrix easier to convert to rref) is:

$$\begin{bmatrix} 1 & x_1 & \mid & y_1 \\ 1 & x_2 & \mid & y_2 \end{bmatrix}$$

The rank of the coefficient matrix is 2. Note that the system has *full row rank* and is therefore consistent by definition. The system also has *full column rank.* This, along with consistency, implies a unique solution. The "unique solution" here refers to a unique straight line passing through the two points. Note, however, that since the coefficient matrix has full row rank, the system has no potential to be inconsistent regardless of the choice of outputs. Therefore, since there is no potential for falsification of the linearity hypothesis, we cannot use this as evidence in favor of the linearity hypothesis.

(3) If we evaluate f at three inputs, we have a serious opportunity to falsify the linearity hypothesis. Thus, this is the first serious test of the linearity hypothesis. Suppose the three input-output pairs are (x_1, y_1), (x_2, y_2), and (x_3, y_3). The augmented matrix is:

$$\begin{bmatrix} 1 & x_1 & | & y_1 \\ 1 & x_2 & | & y_2 \\ 1 & x_3 & | & y_3 \end{bmatrix}$$

The coefficient matrix has rank 2. Therefore it has full column rank but does not have full row rank. What this means is that *if* a solution exists, the solution is unique (geometrically, if a line passes through the three points, it is unique) but a solution need not exist.

There are two cases:

- The points on the graph are non-collinear. This corresponds to the case that the system of equations is inconsistent, or equivalently, when we row reduce the system, we get a row with zeros in the coefficient matrix and a nonzero augmenting entry. This definitively falsifies the linearity hypothesis.
- The points on the graph are collinear. This corresponds to the case that the system of equations is consistent. This is *consistent* with the linearity hypothesis, hence does not falsify it. There was *potential* for falsification, so this is therefore evidence in favor of the linearity hypothesis.

(4) Similar remarks apply to the evaluation of f at more than three points. The coefficient matrix now has n rows and rank 2. It therefore has full column rank (so unique solution if there exists a solution) but does not have full row rank (so that the system may be inconsistent). Explicitly, the augmented matrix is of the form:

$$\begin{bmatrix} 1 & x_1 & | & y_1 \\ 1 & x_2 & | & y_2 \\ \cdot & \cdot & | & \cdot \\ \cdot & \cdot & | & \cdot \\ \cdot & \cdot & | & \cdot \\ 1 & x_n & | & y_n \end{bmatrix}$$

Two cases:

- The points on the graph are not all collinear. This corresponds to the case that the linear system is inconsistent. In this case, the linearity hypothesis is definitively rejected, i.e., falsified.
- The points on the graph are all collinear. This corresponds to the case that the linear system is consistent. This is consistent with the linearity hypothesis, and can be construed as evidence in favor of the hypothesis. The greater the number of points, the stronger the evidence, because the greater the potential for falsification.

1.4. **Two different pictures.** It's worth noting that there are two different types of pictures we are using:

- One picture is the xy-plane in which we are trying to fit existing data points to obtain the graph $y = f(x)$ of the function f. The two axes here are the axis for x and the axis for $y = f(x)$.
- The other picture is a plane, but the coordinates for the plane are now the parameters m and c, viewed as variables. Every *point* in this plane corresponds to a particular choice of *function* $f(x) = mx + c$ and therefore a *line* in the earlier plane.

The switching back and forth between these two different geometric structures that we use can be a bit confusing. Note also that in this case, both the pictures are in two dimensions, but there is no intrinsic reason why the two kinds of pictures should involve the same number of dimensions. The number of dimensions used for the first picture is 2 because the function has one input and one output. The number of dimensions used for the second picture is 2 because that is the number of parameters. These numbers coincide because we are using an affine linear model, where the number of parameters is one more than the number of inputs. When we are using polynomial models, then the dimensions do not match. In any case, there is no intrinsic reason why the dimensions should be related.

1.5. **Short generic summary.** The following general principles relating the linear algebra setting and hypothesis testing will be useful for the :

- The coefficient matrix has full row rank: The system is always consistent (regardless of the values of the outputs), so we cannot use consistency as evidence in favor of the hypothesis.

- The coefficient matrix does not have full row rank: There is potential for inconsistency, so consistency is evidence in favor of the hypothesis.
- The coefficient matrix has full column rank: The system has a unique solution if consistent, so can be used to find the parameters assuming the truth of the hypothesis.
- The coefficient matrix does not have full column rank: We cannot find the solutions uniquely even if the system is consistent.
- The coefficient matrix has full row rank and full column rank (the situation arises when number of parameters = number of input-output pairs, and the inputs are chosen to give independent information): We can find the parameters uniquely assuming the truth of the hypothesis, but we cannot verify the truth of the hypothesis.
- The coefficient matrix has full column rank but does not have full row rank: We can find the parameters uniquely and also obtain independent confirmation of the truth of the hypothesis.

1.6. **Type I and Type II errors and p-values.** *Note: Some of this is quite confusing, so I don't really expect you to understand all the terminology if you haven't taken statistics, and perhaps even if you have.*

Let's borrow a little terminology from statistics to get a better sense of what's going on. Suppose there are two hypotheses at hand regarding whether the function f of one variable is linear. One hypothesis, called the *null hypothesis*, states that f is not linear. The other hypothesis, which is precisely the logical opposite of the null hypothesis, is that f is in fact linear.

Let's say we conduct our "hypothesis testing" by looking at n points, calculating the function values, and then looking at the points in the graph of the function and checking if they are collinear. The rule is that we reject the null hypothesis (i.e., conclude that the function is linear) if the n points are collinear. Otherwise, we do not reject the null hypothesis, i.e., we conclude that the function is not linear.

In principle, any procedure of this sort could involve two types of errors:

- *Type I error*, or "false positive" error, refers to a situation where we incorrectly reject the null hypothesis. In this case, it refers to a situation where we incorrectly conclude that the function is linear, even though it is not.
- *Type II error*, or "false negative" error, refers to a situation where we incorrectly fail to reject the null hypothesis. In ths case, it refers to a situation where we incorrectly conclude that the function is not linear, even though it is linear.

Now, in the hypothesis test we have designed here, Type II errors are impossible. In other words, if the function is linear, we will always conclude that "it is linear" if we use the above test. Type I errors, however, are possible. For $n = 1$ and $n = 2$, the test is useless, because we'll always end up with Type I errors for non-linear functions. For $n \geq 3$, Type I errors are possible in principle. But how likely are they to occur in practice? Quantifying such probabilities is very difficult. There are several different approaches one could use, including p-values and the α and β values, all of which mean different but related things. It's probably not worth it to go into these at this point in the course, but we'll get back to it later in the course in a setting more amenable to probability.

Instead of going into the details, I will explain briefly why, barring some very special information about the nature of the function, we should loosely expect Type I errors to be quite unlikely for this setup. In fact, in many models, the associated p-value for this setup is 0. Here's the intuitive reasoning: let's say we calculated the function values at two points, then fitted a line through them. Now, we want to look at the value of the function at a third point. If the function is an arbitrary function in a loose sense, the probability of its value at the third point just "happening" to be the right value for linearity seems very low: it's effectively asking for the probability that a randomly chosen real number takes a particular predetermined value. Thus, getting collinear points in the graph for three or more inputs is strong evidence in favor of linearity: it would be highly unlikely that you'd just hit upon such points by chance.

1.7. **Bayesian reasoning.** In practice, the reasoning above does not occur in a vacuum. Rather, it occurs in a *Bayesian* framework. Namely, you already have some *prior* ideas regarding the nature of the function. Data that you collect then lets you update your priors.

Below, we discuss some examples of priors under which collecting data of this sort does not provide very strong evidence of linearity. Suppose your prior belief is that the function is a *piecewise linear continuous function*: we can break the reals into contiguous intervals such that the restriction to each interval is linear, but the function as a whole is not linear. For instance, $|x|$ is a piecewise linear function of x, because it has the definition:

$$|x| = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

There are reasonable prior probability distributions on collections of piecewise linear functions where information about three points being collinear is partial, but not near-conclusive, evidence of the function overall being linear. "Piecewise linearity" offers an alternative hypothesis to global linearity that can parsimoniously explain the observed data of collinear points on the graph. Namely: perhaps all your observed points just happened to be in one piece of the piecewise linear description! Piecewise linearity is not merely a hypothetical scenario. Many functions in finance and taxation are designed to be piecewise linear. If we restrict all the inputs as being close to each other, we may be fooled into thinking the function is linear.

For useful hypothesis testing of linearity that could distinguish it from piecewise linearity, we want to pick our points spaced away as far as possible, so that it is harder to explain away linearity of the points on the graph by saying that we got lucky about all points being in the same piece.

Another type of situation where it may be hard to reject alternatives to linearity is a situation where all our observations are at integer inputs, and we believe the function may have the form:

$$f(x) := mx + A\sin(\pi x) + c$$

Note that the $A\sin(\pi x)$ is not detected by the choices of inputs, all of which are integers. Again, useful testing of this alternative hypothesis can happen only if we choose non-integer inputs.

Everything boils down to what type of prior we have, based on broad theoretical considerations, about the model of the function we are dealing with.

2. Functions of multiple variables

For functions of one variable, we can perform the entire analysis by imagining the nature of the graph of the function. For functions of two variables, graphing already becomes more tricky: the graph of the function is now a surface in three-dimensional space. For functions of three or more variables, the graph becomes too tricky to even consider.

Fortunately, we have another option: setting up a linear system.

2.1. The linear system setup: three input-output pairs for a function of two variables.

Suppose we have a function f of two variables x and y. If f is (affine) linear (i.e., we allow for a nonzero intercept), it has the form:

$$f(x, y) := ax + by + c$$

where a, b, and c are constants.

We do not, however, know for sure whether f is linear. We have a black box that outputs the values of f for various inputs.

If we provide three inputs (x_1, y_1), (x_2, y_2), and (x_3, y_3) to f, with outputs z_1, z_2, and z_3 respectively, the augmented matrix we get (with the columns corresponding to c, a, and b in that order) is:

$$\left[\begin{array}{ccc|c} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{array}\right]$$

There are two cases now:

- The inputs are collinear: In this case, the row rank of the system is 2. In other words, the system does not have full row rank. One of the rows of the coefficient matrix is redundant. Thus, the system may or may not be consistent. Which case occurs depends on the outputs. We consider both cases:
  - The system is inconsistent: This means that we have falsified the hypothesis of linearity. Pictorially, this means that in $\mathbb{R}^3$, the three points (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3) are non-collinear.

  - The system is consistent: This means that the data are consistent with the hypothesis of linearity. Pictorially, this means that in $\mathbb{R}^3$, the three points (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3) are collinear. In other words, we cannot reject the hypothesis of the function being linear. Thus, we have collected strong evidence in favor of the linearity hypothesis. Note that even though we have collected evidence in favor of linearity, the system not having full column rank means that we cannot determine a, b, and c uniquely.
- The inputs are non-collinear: In this case, the row rank of the system is 3. In other words, the system has full row rank, and also full column rank. Thus, whatever the output, there exists a unique solution, i.e., we uniquely determine a, b, and c. In fact, we can find unique values of a, b, and c even if the function is not linear. In other words, if the inputs are non-collinear, then we do not get any information that could potentially falsify the linearity hypothesis, hence we cannot conclude anything in favor of the linearity hypothesis.

In other words, if we are only allowed three inputs, then we have to choose between either attempting to test the hypothesis or attempting to find a, b and c uniquely (conditional to the truth of the hypothesis).

2.2. **More than three input-output pairs for a function of two variables.** If, however, we are allowed four or more inputs, we can have our cake and eat it too. As long as three of our inputs are non-collinear, we can use them to obtain the coefficients a, b, and c assuming linearity. Pictorially, this would be fixing the plane that is the putative graph of the function in three dimensions. Further input-output information can be used to test the theory. Pictorially, this is equivalent to testing whether the corresponding points in the graph of the function lie in the plane that they are allegedly part of.

2.3. **Functions of more than two variables.** Suppose f is a function of more than two variables that is alleged to be affine linear (i.e., we allow for an intercept), i.e., it is alleged to be of the form:

$$f(x_1, x_2, \dots, x_n) = a_1x_1 + a_2x_2 + \dots + a_nx_n + c$$

If you are skeptical of the claim of linearity, and/or want to convince somebody who is skeptical, the easiest thing to do would be to pick three points which are collinear in $\mathbb{R}^n$, thus getting a $3 \times (n + 1)$ coefficient matrix of rank 2. If the system is inconsistent, that is definite evidence against linearity. If the system is consistent, that is evidence in favor of linearity, but it is not conclusive.

If the goal is to *find* the values a_1, a_2, $\dots$, a_n, and c then we should pick $n + 1$ different points that are *affinely independent*. This is a concept that we will define later. It is beyond the current scope. However, a randomly selected bunch of points will (almost always) work.

If, however, we want to find the coefficients *and* obtain independent confirmation of the theory, then we need to use at least $n+2$ observations, with the first $n+1$ being affinely independent. The more observations we use, the more chances we have given to potential falsifiers, and therefore, if the linearity hypothesis still remains unfalsified despite all the evidence that could falsify it, that is very strong evidence in its favor.

2.4. **Bayes again!** Recall that, as mentioned earlier, how strongly we view our evidence as supporting the function being linear depends on our prior probability distribution over alternative hypotheses. Consider, for instance, the function $f(x, y) := x + y^2 - 2$. For any fixed value of y, this is linear in x. Thus, if we choose our three collinear points on a line with a fixed y-value, then this function will "fool" our linearity test, i.e., we will get a Type I error.

Therefore, as always, the original prior distribution matters.

3. Verifying a nonlinear model that is linear in parameters

Many similar remarks apply when we are attempting to verify whether a particular nonlinear model describes a function, if the model is linear in the parameters.

3.1. **Polynomial function of one variable.** Consider the case of a function of one variable that is posited to be a polynomial of degree at most n, knowing the function value at $n + 1$ distinct points allows us to determine the polynomial, but does not allow any verification of whether the function is indeed polynomial, because we can fit a polynomial of degree $\leq n$ for *any* collection of $n+1$ input-output pairs. If, however, we have $n+2$ input-output pairs and we get a polynomial of degree $\leq n$ that works for them, that is strong evidence in favor of the model being correct.

Let's make this more explicit. Consider a model with the functional form:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n$$

This is a generic polynomial of degree $\leq n$. There are $n+1$ parameters $\beta_0, \beta_1, \ldots, \beta_n$. Given m input-output pairs (x_1, y_1), (x_2, y_2), $\ldots$, (x_m, y_m), the augmented matrix that we obtain is:

$$\left[\begin{array}{ccccc|c} 1 & x_1 & x_1^2 & \ldots & x_1^n & y_1 \\ 1 & x_2 & x_2^2 & \ldots & x_2^n & y_2 \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot \\ 1 & x_m & x_m^2 & \ldots & x_m^n & y_m \end{array}\right]$$

Note that the coefficient matrix, as always, depends only on the inputs. The coefficient matrix is a $m \times (n+1)$ matrix:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{bmatrix}$$

By standard facts about rank, the rank of the coefficient matrix is at most $\min\{m, n+1\}$. It turns out that, for matrices of this type, the rank always achieves its maximum value as long as the values $x_1, x_2, \ldots, x_m$ are all distinct. This is not completely obvious, and in fact depends on some algebraic manipulation. But it should not be *surprising* per se, because we know that the values at different points are "independent" pieces of data and therefore should give independent equations to the extent feasible, so we expect the coefficient matrix to have the largest possible rank.

Essentially, this means that:

- If $m \leq n$, then we do not have enough information either to verify the hypothesis of being a polynomial of degree $\leq n$ or to find the coefficients assuming the truth of the hypothesis. Our inability to use the data to verify the hypothesis arises because the system has full row rank m, so it is *always* consistent, regardless of output. Our inability to find the solution uniquely arises from the fact that the dimension of the solution space is $n+1-m > 0$, so the solution is non-unique.
- If $m = n+1$, then we do *not* have enough information to verify the hypothesis of being a polynomial of degree $\leq n$, but we *do* have enough information to find the polynomial assuming the truth of the hypothesis. Our inability to use the data to verify the hypothesis arises because the coefficient matrix is now a square matrix of full row rank $m = n+1$, so the system is always consistent, regardless of output. On the other hand, since the coefficient matrix has full column rank, the solution, if it exists, is unique, so we can find the polnyomial uniquely.
- If $m > n+1$, then we have information that can be used both to verify the hypothesis of being polynomial of degree $\leq n$ (albeit not concislvey) and we also have enough information to find the polynomial assuming the truth of the hypothesis. Our ability to use the data to verify the hypothesis arises because the coefficient matrix no longer has full row rank (the rank is $n+1$, and is less than m) so there is potential for inconsistency, therefore, consistency provides evidence in favor of the hypothesis. We can find the polynomial uniquely assuming the truth of the hypothesis because the matrix has full column rank $n+1$, so the solution, if it exists, is unique.

3.2. **Polynomial function of more than one variable.** The ideas discussed above can be applied to the case where the functional form is polynomial of bounded degree in more than one variable. For instance, a functional form for a polynomial of total degree at most 2 in the variables x and y is:

$$f(x, y) = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2$$

We notice the following:

- In order to *find* the parameters $a_1, a_2, \dots, a_6$ uniquely, we would need to determine the outputs for a well-chosen collection of six inputs. By "well-chosen inputs" we mean that the inputs should satisfy the property that the coefficient matrix (which is a 6×6 square matrix) has full rank.

  Note that, when we were talking of an affine linear function of two variables, the condition for being well-chosen was that the inputs are not collinear as points in the xy-plane. It is possible to find a similar geometric constraint on the nature of the inputs for the coefficient matrix here to have full rank. However, finding that constraint would take us deep into realms of higher mathematics that we are not prepared to enter.
- In order to *find* the parameters and *additionally to obtain verification of the model*, we would need to determine the outputs for a well-chosen collection of more than six inputs. Here, "well-chosen" means that the coefficient matrix would still have rank 6, so it has full column rank (allowing us to find the polynomial uniquely if it exists) but does not have full row rank (creating a potential for inconsistency, and therefore allowing us to use consistency to obtain evidence in favor of the claim).

4. Measurement and modeling errors

Most of the discussion above is utopian because it ignores something very real in most practical applications: *error*, both *measurement error* and *modeling error*. Measurement error means that the inputs and outputs as measured are only approximately equal to the true values. Modeling error means that we are not claiming that the function is actually linear. Rather, we are claiming that the function is only approximately linear, even if we use the "true values" of the variables rather than the measured values.

We use the term *overdetermined* for a linear system that has more than the required minimal equations to determine the parameter values. In the absence of measurement error, overdetermined systems will be consistent assuming the hypothesis is true, i.e., the extra equations allow us to "test" the solution obtained by solving a minimal subset.

However, measurement errors can ruin this, and we need to adopt a more error-tolerant approach. The methods we use for that purpose fall broadly under the category of *linear regression*. We will discuss them later in the course.

LINEAR TRANSFORMATIONS

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 2.1.

Executive summary

(1) A $n \times m$ matrix defines a function from $\mathbb{R}^m$ to $\mathbb{R}^n$ via matrix-vector multiplication. A function arising in this manner is termed a *linear transformation*. Every linear transformation arises from a unique matrix, i.e., there is a bijection between the set of $n \times m$ matrices and the set of linear transformations from $\mathbb{R}^m$ to $\mathbb{R}^n$.

(2) A function (also called map) $f : A \to B$ of sets is termed *injective* if no two elements of A map to the same element of B. It is termed *surjective* if B equals the range of f. It is termed *bijective* if it is both injective and surjective. A bijective map has a unique inverse map.

(3) The standard basis vector $\vec{e}_i$ is the vector with a 1 in the i^{th} coordinate and 0s elsewhere. The image of $\vec{e}_i$ is precisely the i^{th} column of the matrix describing the linear tranformation.

(4) A linear transformation can alternatively be defined as a map that preserves addition and scalar multiplication. Explicitly, $T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation if and only if $T(\vec{x} + \vec{y}) = T(\vec{x}) + T(\vec{y})$ (additivity) and $T(a\vec{x}) = aT(\vec{x})$ (scalar multiples) for all $\vec{x}, \vec{y} \in \mathbb{R}^m$ and all $a \in \mathbb{R}$.

(5) A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is *injective* if the matrix of T has full column rank, which in this case means rank m, because the dimensions of the matrix are $n \times m$. Note that this in particular implies that $m \leq n$. The condition $m \leq n$ is a *necessary but not sufficient condition* for the linear transformation to be injective.

(6) A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is *surjective* if the matrix of T has full row rank, which in this case means rank n, because the dimensions of the matrix are $n \times m$. Note that this in particular implies that $n \leq m$. The condition $n \leq m$ is a *necessary but not sufficient condition* for the linear transformation to be surjective.

(7) A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is *bijective* if the matrix of T has full row rank and full column rank. Thus forces $m = n$, and forces the (now square) matrix to have full rank. As mentioned before, this is equivalent to invertibility.

(8) The inverse of a diagonal matrix with all diagonal entries nonzero is the matrix obtained by inverting each diagonal entry individually.

(9) A permutation matrix is a matrix where each row has one 1 and all other entries 0, and each column has one 1 and all other entries 0. A permutation matrix acts by permuting the standard basis vectors among themselves. It can be inverted by permuting the standard basis vectors among themselves in the reverse direction. Hence, the inverse is also a permutation matrix. If the permutation matrix just flips two basis vectors, it is self-inverse.

(10) The inverse of a shear operation is the shear operation obtained by using the negative of the shear. In particular:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

(11) Inverting a matrix requires solving a system of simultaneous linear equations with that as coefficient matrix and with the augmenting column a *generic* output vector, then using the expression for the input in terms of the output to get the matrix of the transformation. It can alternatively be done by augmenting the matrix with the identity matrix, row-reducing the matrix to the identity matrix, and then looking at what the augmented side has become.

(12) We can use linear transformations for the purposes of coding, compression, and extracting relevant information. In many of these practical applications, we are working, not over the real numbers, but over the field of two elements, which is ideally suited for dealing with the world of bits (binary digits).

1. Linear transformations from one space to another

1.1. **Matrix-vector multiplication defines a map between vector spaces.**

Recall that if A is a $n \times m$ matrix, and $\vec{x}$ is a $m \times 1$ column vector (i.e., $\vec{x}$ is a m-dimensional vector written as a column vector) then $A\vec{x}$ is a $n \times 1$ vector, i.e., $A\vec{x}$ is a n-dimensional vector written as a column vector. We can thus think of the matrix A as describing a function:

m-dimensional column vectors $\to$ n-dimensional column vectors

Further, recall that a vector can be thought of as describing the coordinates of a generic point in the space of the same number of dimensions. In particular, m-dimensional column vectors can be considered points in $\mathbb{R}^m$ and n-dimensional column vectors can be considered points in $\mathbb{R}^n$. Thus, A can be viewed as defining, via matrix-vector multiplication, a map:

$$\mathbb{R}^m \to \mathbb{R}^n$$

In other words, every $n \times m$ matrix defines a map from $\mathbb{R}^m$ to $\mathbb{R}^n$ via matrix-vector multiplication.

Please note very carefully that the dimension of the input space is the *number of columns* in the matrix and coincides with the *dimension of the row vectors in the matrix.* The dimension of the output space is the *number of rows* in the matrix and coincides with the *dimension of the column vectors in the matrix.*

Note also that the i^{th} coordinate of the output is obtained as the dot product of the i^{th} row of A with the input vector.

1.2. **First definition of linear transformation.**

We give the first definition:

Definition (Linear transformation)**.** A function $T : \mathbb{R}^m \to \mathbb{R}^n$ is termed a *linear transformation* if there exists a $n \times m$ matrix A (dependent on T) such that $T(\vec{x}) = A\vec{x}$ for all $\vec{x}$ in $\mathbb{R}^m$ (writing $\vec{x}$ as a column vector).

In other words, a transformation is linear if it can be represented via multiplication by a matrix.

We can think of a map at the *meta* level:

(Set of $n \times m$ matrices over $\mathbb{R}$) $\to$ (Linear transformations $\mathbb{R}^m \to \mathbb{R}^n$)

The map takes a matrix as input and outputs the linear transformation induced via multiplication by the matrix.

Note that the set of outputs of the map is itself a set of functions. That's why I said above that this is a map at the *meta* level.

1.3. **Injectivity, surjectivity, and bijectivity.**

The following terminology, which you may or may not have seen, will be useful in coming discussions.

Suppose $f : A \to B$ is a function (also called a *map* or a *mapping*). We say that f is:

- *injective* or *one-one* or *one-to-one* if different inputs always give different outputs, i.e., if whenever $x, y \in A$ satisfy $f(x) = f(y)$, then $x = y$. Another framing of this is that every element of B is the image of *at most* one element of A.
- *surjective* if the range of f is B, i.e., every element of B arises as the image of *at least* one element of A.
- *bijective* if it is both injective and surjective. Another framing is that every element of B arises as the image of *exactly* one element of A. If f is bijective, it establishes a "one-one correspondence" between the sets A and B.

1.4. **Bijectivity between matrices and linear transformations.** We noted a while back the existence of the map:

(Set of $n \times m$ matrices over $\mathbb{R}$) $\to$ (Linear transformations $\mathbb{R}^m \to \mathbb{R}^n$)

The map is *surjective* by the definition of linear transformation. But is it injective? That's a trickier question. At its heart is the question of whether two matrices that are not identical can give rise to the same linear transformation. If that cannot happen, the map is injective.

The map is indeed injective, but to better understand this, we need to think a little bit more about the outputs of specific vectors.

The space $\mathbb{R}^m$ has m very special vectors called the *standard basis vectors*:[1] these are vectors with a 1 in one coordinate and 0s in all the other coordinates. The basis vector e_i is the vector whose i^{th} coordinate is a 1 and remaining coordinates are all 0s. In other words, the standard basis vectors $\vec{e}_1, \vec{e}_2, \dots, \vec{e}_m$ are the vectors:

$$\vec{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \vec{e}_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \dots, \vec{e}_m = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}$$

The following will turn out to be true:

- If a matrix A describes a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$, then $T(\vec{e}_i)$ is the i^{th} column of A. Note that $T(\vec{e}_i)$ is a n-dimensional column vector.
- In particular, knowledge of the outputs $T(\vec{e}_1), T(\vec{e}_2), \dots T(\vec{e_m})$ is sufficient to reconstruct the whole matrix A.
- Thus, knowledge of T as a function (which would entail knowledge of the image of every input under T) is sufficient to reconstruct A.
- Thus, the mapping from the set of $n \times m$ matrices over $\mathbb{R}$ to linear transformations $\mathbb{R}^m \to \mathbb{R}^n$ is injective. Since we already noted that the mapping is surjective, it is in fact bijective. In other words, every matrix gives a linear transformation, and every linear transformation arises from a unique matrix.

1.5. **An alternative definition of linear transformation.** We can also define linear transformation in another, albeit related, way. This definition relies on some universal equalities that the map must satisfy. First, recall that we can add vectors (addition is coordinate-wise) and multiply a real number and a vector (the real number gets multiplied with each coordinate of the vector).

Definition (Linear transformation (alternative definition))**.** A function $T : \mathbb{R}^m \to \mathbb{R}^n$ is termed a *linear transformation* if it satisfies the following two conditions:

1. *Additivity*: $T(\vec{x} + \vec{y}) = T(\vec{x}) + T(\vec{y})$ for all vectors $\vec{x}, \vec{y} \in \mathbb{R}^m$. Note that the addition on the inside in the left side is in $\mathbb{R}^m$ and the addition on the right side is in $\mathbb{R}^n$.
2. *Scalar multiples*: $T(a\vec{x}) = aT(\vec{x})$ for all real numbers a and vectors $\vec{x} \in \mathbb{R}^m$.

To show that this definition is equivalent to the earlier definition, we need to show two things:

- Every linear transformation per the original definition (i.e., it can be represented using a matrix) is a linear transformation per the new definition: This follows from the fact that matrix-vector multiplication is linear in the vector. We noted this earlier when we defined matrix-vector multiplication.
- Every linear transformation per the new definition is a linear transformation per the original definition: The idea here is to start with the linear transformation T per the new definition, then construct the putative matrix for it using the values $T(\vec{e}_1), T(\vec{e}_2), \dots, T(\vec{e_m})$ as columns. Having constructed

[1]This is a lie in some ways, but that does not matter right now

the putative matrix, we then need to check that this matrix gives the linear transformation (per the new definition) that we started with.

2. Behavior of linear transformations

2.1. **Rank and its role.** The rank of a linear transformation plays an important role in determining whether it is injective, whether it is surjective, and whether it is bijective. Note that our earlier discussion of injective, surjective and bijective was in the context of a "meta" map from a set of matrices to a set of linear transformations. Now, we are discussing the injectivity, surjectivity, and bijectivity of a specific linear transformation.

Note that finding the pre-images for a given linear transformation is tantamount to solving a linear system where the augmenting column is the desired output. And, we have already studied the theory of what to expect with the solutions to systems of linear equations. In particular, we have that:

- A linear transformation is injective if and only if its matrix has full column rank. In other words, $T : \mathbb{R}^m \to \mathbb{R}^n$ is injective if and only if its matrix, which is a $n \times m$ matrix, has rank m. Note that this is possible only if $m \leq n$.

  This is because, as we saw earlier, the dimension of the solution space for any specific output is the difference (number of variables) - (rank). In order to get injectivity, we wish for the solution space to be zero-dimensional, i.e., we wish that whenever the system has a solution, the dimension of the solution space is zero. This means that the rank must equal the number of variables.

  The condition $m \leq n$ is *necessary but not sufficient* for the linear transformation to be injective. Intuitively, this makes sense. For a map to be injective, the target space of the map must be *at least as big* as the domain. Since the appropriate size measure of vector spaces in the context of linear transformations is the dimension, we should expect $m \leq n$. Note that $m \leq n$ is not sufficient: any linear transformation whose matrix does not have full column rank is not injective, even if $m \leq n$. An extreme example is the zero linear transformation, whose matrix is the zero matrix. This is not injective for $m > 0$.
- A linear transformation is surjective if and only if its matrix has full row rank. In other words, $T : \mathbb{R}^m \to \mathbb{R}^n$ is surjective if and only its matrix, which is a $n \times m$ matrix, has rank n. Note that this is possible only if $n \leq m$.

  This is because, as we saw earlier, if the coefficient matrix has full row rank, the system is always consistent. Hence, every output vector occurs as the image of some input vector, so the map is surjective.

  The condition $n \leq m$ is *necessary but not sufficient* for the linear trasnformation to be surjective. Intuitively, this makes sense. For a map to be surjective, the target space must be *at most as large* as the domain. The appropriate size measure of vector spaces in the context of linear transformations is dimension, so $n \leq m$ follows. Note that $n \leq m$ is not sufficient: any linear transformation whose matrix does not have full row rank is not surjective, even if $n \leq m$. An extreme example is the zero linear transformation, given by the zero matrix. This is not surjective if $n > 0$.
- A linear transformation can be bijective only if its domain and co-domain space have the same dimension, so that its matrix is a square matrix, *and* that square matrix has full rank.

Our interest for now will be in bijective linear transformations of the form $T : \mathbb{R}^n \to \mathbb{R}^n$, for which, as noted above, the rank is n, and thus, the rref is the identity matrix.

For a bijective linear transformation, we can define an *inverse* transformation that sends each element of the target space to the unique domain element mapping to it. This inverse transformation will also turn out to be linear.

Note that the inverse is unique, and the inverse of the inverse is the original linear transformation.

We will later on see the matrix algebra definition and meaning of the inverse.

3. Some important linear transformations and their inverses

3.1. **Linear transformations with diagonal matrices.** Consider a linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ whose matrix is a *diagonal* matrix. For instance, consider this linear transformation T given by the matrix:

$$A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

This means that:

$$T(\vec{e_1}) = 3\vec{e_1}, T(\vec{e_2}) = 7\vec{e_2}, T(\vec{e_3}) = 4\vec{e_3}$$

Diagonal linear transformations are special in that they send each standard basis vector to a multiple of it, i.e., they do not "mix up" the standard basis vectors with each other. To invert a diagonal linear transformation, it suffices to invert the operation for each standard basis vector, while keeping it diagonal. In other words, we invert each diagonal entry, keeping it in place.

$$A^{-1} = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/7 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}$$

Note that for a diagonal matrix where one or more of the diagonal entries is zero, the matrix does not have full rank. In fact, the corresponding standard basis vector gets killed by the linear transformation. Thus, the inverse does not exist.

3.2. **Linear transformations that permute the coordinates.** Consider the linear transformation T with matrix:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

We can describe T as follows:

$$T(\vec{e_1}) = \vec{e_2}, T(\vec{e_2}) = \vec{e_1}$$

In other words, T interchanges the two standard basis vectors $\vec{e_1}$ and $\vec{e_2}$. To invert T, we need to do T again: interchage them back! Thus, the inverse matrix is:

$$A^{-1} = A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Let's take a 3×3 matrix that cycles the vectors around:

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

We can read off the images of $\vec{e_1}$, $\vec{e_2}$, and $\vec{e_3}$ by looking at the respective columns. The first column describes $T(\vec{e_1})$, and it is $\vec{e_3}$. The second column describes $T(\vec{e_2})$, and it is $\vec{e_1}$. The third column describes $T(\vec{e_3})$, and it is $\vec{e_2}$. In other words:

$$T(\vec{e_1}) = \vec{e_3}, T(\vec{e_2}) = \vec{e_1}, T(\vec{e_3}) = \vec{e_2}$$

Thus, this linear transformation cycles the vectors around as follows:

$$\vec{e_1} \to \vec{e_3} \to \vec{e_2} \to \vec{e_1}$$

To invert this, we need to cycle the vectors in the reverse order:

$$\vec{e_1} \to \vec{e_2} \to \vec{e_3} \to \vec{e_1}$$

For $\vec{e_1} \to \vec{e_2}$, we need the first column to be $\vec{e_2}$. For $\vec{e_2} \to \vec{e_3}$, we need the second column to be $\vec{e_3}$. For $\vec{e_3} \to \vec{e_1}$, we need the third column to be $\vec{e_1}$.

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

A square matrix is termed a *permutation matrix* if it has one 1 and the remaining entries 0 in each row, and similarly it has one 1 and the remaining entries 0 in each column. There is a rich theory of permutation matrices that relies only on the theory of what are called permutation groups, which deal with finite sets. Enticing though the theory is, we will not explore it now.

3.3. **Shear operations and their inverses.** Consider the shear operation T with matrix:

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

This matrix does not keep each coordinate within itself, nor does it merely permute the coordinates. Rather, it adds one coordinate to another. Explicitly:

$$T(\vec{e}_1) = \vec{e}_1, T(\vec{e}_2) = \vec{e}_1 + \vec{e}_2$$

The operation T^{-1} should behave as follows:

$$T^{-1}(\vec{e}_1) = \vec{e}_1, T^{-1}(\vec{e}_1 + \vec{e}_2) = \vec{e}_2$$

Thus:

$$T^{-1}(\vec{e}_2) = T^{-1}(\vec{e}_1 + \vec{e}_2) - T^{-1}(\vec{e}_1)$$

We get that:

$$T^{-1}(\vec{e}_2) = \vec{e}_2 - \vec{e}_1$$

Thus, we get that:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

Note that the choice of the word "shear" is not a coincidence. We had talked earlier of operations on systems of equations. One of these operation types, called a *shear operation*, involved adding a multiple of one row to another row. You can see that this is similar in broad outline to what we are trying to do here. Is there a formalization of the relationship? You bet! It has to do with matrix multiplication, but we'll have to skip it for now.

4. Computing inverses in general

4.1. **Procedure for computing the inverse: a sneak preview.** *Note*: We will return to this topic in more detail later, after describing matrix multiplication. For the moment, think of this as a very quick sneak preview. It's a tricky collection of ideas, so multiple rounds of exposure help. At this stage, you are not expected to acquire computational fluency.

Consider a $n \times n$ matrix A. Suppose that A has full rank. Hence, it is invertible. For any vector $\vec{x}$, $A^{-1}\vec{x}$ is the unique vector $\vec{y}$ such that $A\vec{y} = \vec{x}$. To obtain this vector, we need to solve the linear system with A as the coefficient matrix and $\vec{y}$ as the augmenting column. The fact that A has full row rank guarantees consistency. The fact that A has full column rank guarantees that the solution is unique.

This is great for finding $A^{-1}\vec{x}$ for individual vectors x. But it would be ideal if we can obtain A^{-1} *qua* matrix, rather than solving the linear system separately for each $\vec{x}$.

There are two equivalent ways of thinking about this.

One is to solve the linear system for an arbitrary vector $\vec{x}$, i.e., without putting in actual numerical values, and get $A^{-1}\vec{x}$ functionally in terms of $\vec{x}$, then pattern match that to get a matrix.

A more sophisticated approach is to calculate $A^{-1}\vec{e}_1, A^{-1}\vec{e}_2, \dots A^{-1}\vec{e}_n$ separately, then combine them into a matrix. Note that all of them use the coefficient matrix A, so instead of writing each augmented matrix separately, we could just write A and augment it with multiple columns, one for each standard basis vector. That is equivalent to augmenting A with the identity matrix. Row reduce A, and keep applying the operations to the identity matrix. When A gets row reduced to the identity matrix, the identity matrix is converted to A^{-1}. We will return to this procedure later, after we have an understanding of matrix multiplication.

Yet another way of thinking of this is that each of the elementary row operations on A *undoes* A a bit, and all of them put together undo A to the identity matrix. Doing all these operations one after the other gives us the entire recipe for undoing A, and hence, applying that recipe to the identity matrix gives A^{-1}.

4.2. **Formula for the inverse of a 2×2 matrix.** Suppose we have a 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

It can be verified that:

- The matrix has full rank (or equivalently, is invertible) if and only if $ad - bc \neq 0$.
- If the matrix has full rank, then the inverse is:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

The expression $ad - bc$ controls whether or not the matrix has full rank. This expression is termed the *determinant* of the matrix. We hope to discuss determinants at a later stage in the course.

4.3. **Time complexity of computing the inverse.** The row reduction approach described above for computing the inverse has roughly the same complexity as ordinary Gauss-Jordan elimination. In particular, the arithmetic time complexity of the process is $\Theta(n^3)$. There are better approaches to matrix inversion in many situations. However, as a general rule, the time complexity cannot be brought down to a lower order using known and easy-to-implement algorithms.

5. Real-world applications

5.1. **Binary content storage.** Content on computers is mostly stored in the form of bits. A *bit* or *binary digit* can take values 0 or 1. The set $\{0, 1\}$ has the structure of a field. This is called the field of two elements. The general theory of vector spaces and linear transformations applies here. This is part of a general theory of finite fields, exploring which would take us too far afield.

A piece of content looks like a sequence of bits. For instance, a 1 KB file is a sequence of 2^{13} bits (2^{10} bytes, and each byte is 2^3 bits). We can think of the individual bits in the file as the coordinates of a vector. The dimension of the vector space for a 1 KB file is 2^{13}. Note that the number of possible 1 KB files is $2^{2^{13}}$.

Transformations that take in files of a certain size and output files of another size may well arise as linear transformations between the corresponding vector spaces. For instance, a linear transformation:

$$\{0, 1\}^{2^{33}} \to \{0, 1\}^{2^{13}}$$

represents a transformation that takes as input a 1 GB file and outputs a 1 KB file.

Note that not every transformation must be linear, but a lot of the kinds of things we wish to achieve can be achieved via linear transformations.

5.2. **Coding/encryption.** Suppose there are two people, Alice and Bob, who wish to communicate messages (which we can think of as binary files) with each other over an insecure channel, where a third party, called Carl, can eavesdrop. Alice and Bob have a little time before they start sending messages (but before they actually know what communications they would like to exchange). They can use this secure, eavesdropper-free meeting to decide on a protocol to encode and decode messages.

Alice and Bob know the size of the file they expect to need to exchange. Let us say the file is n bits long. Alice and Bob agree upon an invertible linear transformation $T : \{0, 1\}^n \to \{0, 1\}^n$. When Alice wants to send Bob a file that can be described by a vector $\vec{v}$ with n coordinates, she does not send $\vec{v}$. Instead, she sends $T\vec{v}$ over the insecure channel.

Bob, who knows T already, also knows T^{-1}. Therefore, he can apply T^{-1} to $T(\vec{v})$ and recover $\vec{v}$. Another way of thinking about this is that Bob needs to solve the linear system whose coefficient matrix is the matrix of T and whose augmenting column is the vector $T(\vec{v})$.

Carl, who overhears $T(\vec{v})$, does not know T (because this was decided by Alice and Bob earlier through a secure communication channel that Carl could not eavesdrop upon). Since Carl has no idea about what T is, it is very difficult for him to decipher $\vec{v}$.

In the process above, the linear transformation T is termed the *encoding transformation* and its inverse transformation T^{-1} is termed the *decoding transformation.* The matrix for T is the *encoding matrix* and the matrix for T^{-1} is termed the *decoding matrix.*

What kind of linear transformation should ideally be chosen for T? There are many competing considerations, which we cannot cover here. The criteria are similar to the criteria for choosing a good password.

There is a problem with this sort of coding. It is not a secure form of encryption. The main problem is that insofar as the encoding matrix needs to be communicated between the sender and receiver, people may intercept it. Once they know the encoding matrix, all they need is elementary linear algebra and they have the decoding matrix. Now, if they intercept any communication, they will be able to decode it. The key problem with this method is that the decoding matrix is easy to deduce from the encoding matrix, because the algorithm for matrix inversion can run in polynomial time, since it essentially relies on Gauss-Jordan elimination.

There are more secure cryptographic methods, including RSA and the Diffie-Hellman key exchange. The key ideas are described below:

(1) The key principle of RSA (public key cryptography) is to use the fact that factoring a number is much harder than finding large primes to construct a pair of (encoding, decoding) such that, even if the encoding procedure is publicly known, the decoding procedure cannot be deduced quickly from that. With RSA, Bob has an (encoding, decoding) pair. He publishes the encoding information (called the *public key*) to the public. This includes Alice and Carl. Alice then sends Bob a message encoded using the public key. Bob then uses his decoding procedure (his *private key*) to decode the message. Alice herself cannot decode her own message after she has encoded it, even though she knows the original message.

    Carl, who is overhearing, knows the encoding procedure and knows the result of the encoding, but he still cannot deduce what the original message was.

    The reason that linear transformations are not amenable to this approach is that with linear transformations, the inverse of a linear transformation can be computed quickly (in time $\Theta(n^3)$ where n is the dimension of the spaces being transformed). So, publishing the encoding procedure is tantamout to publishing the decoding procedure. Note that when working over the field of two elements, as we are here, we do not need to distinguish between arithmetic complexity and bit complexity, because the numbers are always 0 or 1. They don't get arbitrarily complicated.

(2) Diffie-Hellman key exchange involves two people using a possibly insecure channel to exchange information that allows both of them to agree upon an encoding and decoding procedure whereas other people listening in on the same channel cannot figure out what procedure has been decided upon. The idea is that each person contributes part of the procedure in a way that third parties have no way of figuring out what's going on.

In both cases (RSA and Diffie-Hellman) the security of the protocols rests on the computational difficulty of certain mathematical problems related to prime numbers and modular arithmetic. This computational difficulty has not been mathematically proved, so it is possible that there exist fast algorithms to break these allegedly secure protocols. However, theoretical computer scientists and cryptographers believe it is highly unlikely that such fast algorithms exist.

The upshot of that is that the reason linear encoding is insecure is precisely that linear algebra is *computationally too easy!* Even if you don't think of it that way right now.

5.3. **Redundancy.**

Return to the use of linear transformations for encoding. One of the problems with using an invertible (i.e., bijective) linear transformation is that it leaves no room for redundancy. If any bit gets corrupted, the original message can appear very different when decoded from what it was intended to look like. One way of handling with is to build redundancy by using a linear transformation that is injective but not surjective. Explicitly, use an injective linear transformation:

$$T : \{0,1\}^n \to \{0,1\}^{n+p}$$

for the purpose of encoding. Here, $p > 0$ describes the extent of redundancy in the message.

Injectivity still guarantees that the original message can be recovered uniquely from the encoded message. What we need to do is solve a linear system with a $(n+p) \times n$ coefficient matrix and full column rank. Assuming the message is not corrupted, the solution is unique.

What happens if the message gets corrupted? Note that since the system has rank n but has $n+p$ rows (equations), there are p redundant conditions. These p redundant conditions means that with a corrupted message, it is quite likely that the system will become inconsistent and we will not be able to decode the message, as we rightly should not, since it is corrupt. The probability that a corrupted message will still appear like a genuine message (i.e., it will give a consistent system to solve) is $1/2^p$. By choosing p reasonably large, we can make this probability quite small. Note that over the reals, the probability is effectively zero, but we are working over the field of two elements.

If you are interested more in this, look up *error-detecting codes* and *error-correcting codes.*

5.4. **Compression and more on redundancy.** In some cases, linear transformations can be used for purposes such as hashing, checksums, and various forms of compression. The idea is to take the original vector, which may live in a very huge dimension, and use a linear transformation to map it to a smaller dimension. The mapping is not injective, hence it is theoretically conceivable for two different vectors to give the same image. Nonetheless, the probability that two randomly chosen vectors give the same output is very small.

Suppose there are two files, $\vec{v}$ and $\vec{w}$, stored on different nodes of a network, both of size 1 GB. This means that both files can be viewed as vectors with 2^{33} coordinates. We want to check if $\vec{v}$ and $\vec{w}$ are equal. It would be very difficult to send the entire vectors across the network. One approach to checking equality probabilistically is to check if, say, the first 2^{10} coordinates of $\vec{v}$ are the same as the first 2^{15} coordinates of $\vec{w}$ (this is basically the first 4 KB of the files). This method may be misleading because there may have been copy errors made near the end which will not be caught by just looking at a subset of the coordinates.

A better way is to choose a linear transformation $T : \{0,1\}^{2^{33}} \to \{0,1\}^{2^{15}}$ that uses all the coordinates in a nontrivial fashion, apply T to $\vec{v}$ and $\vec{w}$ separately, then look at the outputs $T(\vec{v})$ and $T(\vec{w})$ and check whether they are equal (by communicating them across the network). The files that need to be compared are now just 4 KB long, and this comparison can be made relatively easily.

A couple of other remarks:

- If we wish, we could pick a *random* linear transformation T. A random linear transformation will most likely have rank equal to the output dimension, which is in this case 2^{15}. The probability of full rank is hard to calculate precisely, but it is almost 1.
- The probability of a collision, i.e., the probability that two different vectors $\vec{v}$ and $\vec{w}$ give rise to the same output, is quite low. Explicitly, if T has full rank, the probability is $1/2^{2^{15}}$. The intuitive explanation is that the probability that each bit happens to agree is $1/2$, and we multiply these probabilities. The independence of the probabilities requires full rank.

5.5. **Getting relevant information: the case of nutrition.** Suppose there are m different foodstuffs, and n different types of nutrients, and each food contains a fixed amount of each given nutrient per unit quantity of the good. The "foodstuffs vector" of your diet is a vector whose coordinates describe how much of each foodstuf you consume. The "nutrient vector" of your diet describes how much of each nutrient you obtain. There is a matrix describing the nutritional content of the food that defines a linear transformation from your food vector to your nutrition vector. The matrix has rows corresponding to nutrients and columns corresponding to foodstuffs, with the entry in each cell describing the amount of the row nutrient in the column food. This matrix defines a linear transformation from food vectors to nutrient vectors, i.e., whatever your foodstuff vector is, multiplying it by the matrix gives your nutrient vector.

If you wish for a balanced, healthy, and nutritious diet, this typically involves some constraints on the nutrient vector. Specifically, for some nutrients, there are both minimum and maximum values. For some nutrients, there may be only minimum values. The goal is to find a foodstuff vector whose image under the linear transformation satisfies all the constraints. This requires solving a system of linear *inequalities*. Whether a solution exists or not depends on the nature of the matrix describing the nutritional contents of the foods. For instance, if the only foodstuffs available to you are Coca Cola and Doritos, you are unlikely to be able to find a food vector that gives you a nutrient vector satisfying the constraints.

Note that the linearity assumption is a reasonable approximation for most nutrients, but there may be cases where this linear model fails. This occurs if there are complicated interactions between the nutrients or between various types of foods, and/or diminishing returns in nutrient content as you consume more of a certain food. For instance, it may be the case that consuming food A and food B does not just give you the sum of the nutritional content you would get from consuming foods A and B separately. The linear assumption is probably a reasonable approximation that works for most nutrients and hence one worth using insofar as it keeps the problem tractable.

5.6. **Many different dot products with a fixed vector.** One way of thinking of matrix-vector multiplication is in terms of there being many different dot products we want to compute where one of the vectors in the dot product is fixed, and the other one takes a finite list of values. The varying list is put as rows in a matrix, and the vector that is fixed is put as the column vector to be multiplied.

MATRIX MULTIPLICATION AND INVERSION

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Sections 2.3 and 2.4.

Executive summary

Note: The summary does not include some material from the lecture notes that is not important for present purposes, or that was intended only for the sake of illustration.

(1) *Recall*: A $n \times m$ matrix A encodes a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ given by $T(\vec{x}) = A\vec{x}$.
(2) We can add together two $n \times m$ matrices entry-wise. Matrix addition corresponds to addition of the associated linear transformations.
(3) We can multiply a scalar with a $n \times m$ matrix. Scalar multiplication of matrices corresponds to scalar multiplication of linear transformations.
(4) If $A = (a_{ij})$ is a $m \times n$ matrix and $B = (b_{jk})$ is a $n \times p$ matrix, then AB is defined and is a $m \times p$ matrix. The $(ik)^{th}$ entry of AB is the sum $\sum_{j=1}^{n} a_{ij}b_{jk}$. Equivalently, it is the dot product of the i^{th} row of A and the k^{th} column of B.
(5) Matrix multiplication corresponds to composition of the associated linear transformations. Explicitly, with notation as above, $T_{AB} = T_A \circ T_B$. Note that $T_B : \mathbb{R}^p \to \mathbb{R}^n$, $T_A : \mathbb{R}^n \to \mathbb{R}^m$, and $T_{AB} : \mathbb{R}^p \to \mathbb{R}^m$.
(6) Matrix multiplication makes sense *only if* the number of columns of the matrix on the left equals the number of rows of the matrix on the right. This comports with its interpretation in terms of composing linear transformations.
(7) Matrix multiplication is associative. This follows from the interpretation of matrix multiplication in terms of composing linear transformations and the fact that function composition is associative. It can also be verified directly in terms of the algebraic definition of matrix multiplication.
(8) Some special cases of matrix multiplication: multiplying a row with a column (the inner product or dot product), multiplying a column with a row (the outer product or Hadamard product), and multiplying two $n \times n$ diagonal matrices.
(9) The $n \times n$ identity matrix is an identity (both left and right) for matrix multiplication wherever matrix multiplication makes sense.
(10) Suppose n and r are positive integers. For a $n \times n$ matrix A, we can define A^r as the matrix obtained by multiplying A with itself repeatedly, with A appearing a total of r times.
(11) For a $n \times n$ matrix A, we define A^{-1} as the unique matrix such that $AA^{-1} = I_n$. It also follows that $A^{-1}A = I_n$.
(12) For a $n \times n$ invertible matrix A, we can define A^r for all integers r (positive, zero, or negative). A^0 is the identity matrix. $A^{-r} = (A^{-1})^r = (A^r)^{-1}$.
(13) Suppose A and B are matrices. The question of whether $AB = BA$ (i.e., of whether A and B commute) makes sense only if A and B are both square matrices of the same size, i.e., they are both $n \times n$ matrices for some n. However, $n \times n$ matrices need not always commute. An example of a situation where matrices commute is when both matrices are powers of a given matrix. Also, diagonal matrices commute with each other, and scalar matrices commute with all matrices.
(14) Consider a system of simultaneous linear equations with m variables and n equations. Let A be the coefficient matrix. Then, A is a $n \times m$ matrix. If $\vec{y}$ is the output (i.e., the augmenting column) we can think of this as solving the vector equation $A\vec{x} = \vec{y}$ for $\vec{x}$. If $m = n$ and A is invertible, we can write this as $\vec{x} = A^{-1}\vec{y}$.

(15) There are a number of algebraic identities relating matrix multiplication, addtion, and inversion. These include distributivity (relating multiplication and addition) and the involutive nature or reversal law (namely, $(AB)^{-1} = B^{-1}A^{-1}$). See the "Algebraic rules governing matrix multiplication and inversion" section in the lecture notes for more information.

Computational techniques-related ...

(1) The arithmetic complexity of matrix addition for two $n \times m$ matrices is $\Theta(mn)$. More precisely, we need to do mn additions.
(2) Matrix addition can be completely parallelized, since all the entry computations are independent. With such parallelization, the arithmetic complexity becomes $\Theta(1)$.
(3) The arithmetic complexity for multiplying a generic $m \times n$ matrix and a generic $n \times p$ matrix (to output a $m \times p$ matrix) using naive matrix multiplication is $\Theta(mnp)$. Explicitly, the operation requires mnp multiplications and $m(n-1)p$ additions. More explicitly, computing each entry as a dot product requires n multiplications and $(n-1)$ additions, and there is a total of mp entries.
(4) Matrix multiplication can be massively but not completely parallelized. All the entries of the product matrix can be computed separately, already reducing the arithmetic complexity to $\Theta(n)$. However, we can parallelized further the computation of the dot product by parallelizing addition. This can bring the arithmetic complexity (in the sense of the depth of the computational tree) down to $\Theta(\log_2 n)$.
(5) We can compute powers of a matrix quickly by using repeated squaring. Using repeated squaring, computing A^r for a positive integer r requires $\Theta(\log_2 r)$ matrix multiplications. An explicit description of the minimum number of matrix multiplications needed relies on writing r in base 2 and counting the number of 1s that appear.
(6) To assess the invertibility and compute the inverse of a matrix, augment with the identity matrix, then row reduce the matrix to the identity matrix (note that if its rref is not the identity matrix, it is not invertible). Now, see what the augmented side has turned to. This takes time (in the arithmetic complexity sense) $\Theta(n^3)$ because that's the time taken by Gauss-Jordan elimination (about n^2 row operations and each row operation requires $O(n)$ arithmetic operations).
(7) We can think of pre-processing the row reduction for solving a system of simultaneous linear equations as being equivalent to computing the inverse matrix first.

Material that you can read in the lecture notes, but not covered in the summary.

(1) Real-world example(s) to illustrate matrix multiplication and its associativity (Sections 3.4 and 6.3).
(2) The idea of fast matrix multiplication (Section 4.2).
(3) One-sided invertibility (Section 8).
(4) Noncommuting matrix examples and finite state automata (Section 10).

1. The goal of what we are trying to do

Recall that a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ can be encoded using a $n \times m$ matrix. Note that the dimension of the output space equals the number of rows and the dimension of the input space equals the number of columns. If A is the matrix for T, then $T(\vec{x}) = A\vec{x}$ for any vector $\vec{x} \in \mathbb{R}^m$.

Recall also that A is uniquely determined by T, because we know that the j^{th} column of A is the image in $\mathbb{R}^n$ of the standard basis vector $\vec{e_j} \in \mathbb{R}^m$. In other words, knowing T as a function determines A as a matrix.

Linear transformations are *functions*. This means that we can perform a number of operations on these in the context of operations on functions. Specifically, we can do the following:

- Pointwise addition: For $T_1, T_2 : \mathbb{R}^m \to \mathbb{R}^n$, we can define $T_1 + T_2$ as the operation $\vec{v} \mapsto T_1(\vec{v}) + T_2(\vec{v})$. It is easy to verify from the definition that if T_1 and T_2 are both linear transformations, then $T_1 + T_2$ is also a linear transformation.
- Scalar multiplication by a scalar constant: For $T : \mathbb{R}^m \to \mathbb{R}^n$ and a scalar a, define $aT : \mathbb{R}^m \to \mathbb{R}^n$ as the operation $\vec{v} \mapsto aT(\vec{v})$. It is easy to verify from the definition that if T is a linear transformation and a is a real number, then aT is also a linear transformation.

- Composition: If the co-domain of one linear transformation is the domain of another, it makes sense to *compose* the linear transformations. As you have seen in homework problems, a composite of linear transformations is a linear transformation.
- For a bijective linear transformation, we can define the *inverse*, which (as you have seen in a homework) also turns out to be a linear transformation.

In each case, we would like a description of the matrix for the new linear transformation in terms of matrices for the original linear transformation(s). The description should be purely algebraic, and for specific choices of the matrices, should involve pure arithmetic. Once we have done this, we have converted a tricky conceptual idea involving huge, hard-to-visualize spaces into easily coded numbers with easily coded operations.

We will see that:

- Addition of linear transformations corresponds to an operation that we will call *matrix addition.*
- Scalar multiplication of linear transformations corresponds to *scalar multiplication* of matrices.
- Composition of linear transformations corresponds to *matrix multiplication.*
- Computing the inverse of a linear transformation corresponds to computing the *matrix inverse.*

Let's get to it now!

2. Matrix addition and scalar multiplication

2.1. **Formal definition of matrix addition.** For matrices A and B, we define the sum $A+B$ only if A and B have an equal number of rows and also have an equal number of columns. Suppose A and B are both $n \times m$ matrices. Then $A+B$ is also a $n \times m$ matrix, and it is defined as the matrix whose $(ij)^{th}$ entry is the sum of the $(ij)^{th}$ entries of A and B. Explicitly, if $A = (a_{ij})$ and $B = (b_{ij})$, with $C = A + B = (c_{ij})$, then:

$$c_{ij} = a_{ij} + b_{ij}$$

2.2. **Matrix addition captures pointwise addition of linear transformations.** Suppose $T_1, T_2 : \mathbb{R}^m \to \mathbb{R}^n$ are linear transformations. Then, $T_1 + T_2$ is also a linear transformation from $\mathbb{R}^m$ to $\mathbb{R}^n$, defined by *pointwise addition*, similar to how we define addition for functions:

$$(T_1 + T_2)(\vec{v}) = T_1(\vec{v}) + T_2(\vec{v})$$

Note that the pointwise definition of addition uses only the vector space structure of the target space.

The claim is that the matrix for the pointwise sum of two linear transformations is the sum of the corresponding matrices. This is fairly easy to see. Explicitly, it follows from the distributivity of matrix-vector multiplication.

2.3. **Multiplying a matrix by a scalar.** Suppose A is a $n \times m$ matrix and λ is a real number. We define λA as the matrix obtained by multiplying each entry of A by the scalar λ.

2.4. **Matrices form a vector space.** We can think of $n \times m$ matrices as mn-dimensional vectors, just written in a matrix format. The matrix addition is the same as vector addition, and scalar multiplication of matrices mimics scalar-vector multiplication. In other words, the structure we have so far does not really make use of the two-dimensional storage format of the matrix. The two-dimensional storage format was, however, crucial to thinking about matrices as encoding linear transformations: the columns corresponding to the input coordinates, and the rows corresponded to the output coordinates. We will see that our definition of matrix multiplication is different and not just entry-wise: it involves using the row-column structure.

2.5. **Arithmetic complexity of matrix addition and scalar multiplication.** To add two $n \times m$ matrices, we need to do nm additions, one for each entry of the sum. Thus, the arithmetic complexity of matrix addition is $\Theta(nm)$. Similarly, to multiply a scalar with a $n \times m$ matrix, we need to do nm multiplication, so the arithmetic complexity of scalar multiplication is $\Theta(nm)$.

Both these operations can be massively parallelized if all our parallel processors have access to the matrices. Explicitly, we can make each processor compute a different entry of the sum. Thus, the parallelized arithmetic complexity is $\Theta(1)$. This ignores communication complexity issues.

3. Matrix multiplication: preliminaries

3.1. **Formal definition.** Let A and B be matrices. Denote by a_{ij} the entry in the i^{th} row and j^{th} column of A. Denote by b_{jk} the entry in the j^{th} row and k^{th} column of B.

Suppose the number of columns in A equals the number of rows in B. In other words, suppose A is a $m \times n$ matrix and B is a $n \times p$ matrix. Then, AB is a $m \times p$ matrix, and if we denote it by C with entries c_{ik}, we have the formula:

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$$

The previous cases of dot product (vector-vector multiplication) and matrix-vector multiplication can be viewed as special cases of matrix-matrix multiplication. Explicitly, matrix-matrix multiplication can be thought of in three ways:

- It is a bunch of matrix-vector multiplications: The k^{th} column of the product matrix is the matrix-vector product of the matrix A with the k^{th} column of B.
- It is a bunch of vector-matrix multiplications: The i^{th} row of the product matrix is the vector-matrix product of the i^{th} row of A with the entire matrix B.
- Each entry is a dot product: The $(ik)^{th}$ entry of the product matrix is the dot product of the i^{th} row of A with the k^{th} column of B.

3.2. **It captures composition of linear transformations.** Suppose A is a $m \times n$ matrix and B is a $n \times p$ matrix. Suppose T_A is the linear transformation whose matrix is A and T_B is the linear transformation whose matrix is B. Note that T_B is a transformation $\mathbb{R}^p \to \mathbb{R}^n$ and T_A is a transformation $\mathbb{R}^n \to \mathbb{R}^m$. Then, $T_A \circ T_B$ is the linear transformation $\mathbb{R}^p \to \mathbb{R}^m$ that corresponds to the matrix product AB. In other words, the matrix for the composite of two linear transformations is the product of the matrices.

Note also that the matrix product makes sense in precisely the same circumstance (in terms of number of columns equaling number of rows) as the circumstance where the composite of linear transformations makes sense (the dimension of the output space to the operation done first, namely the one on the right, equals the dimension of the input space to the operation done second, namely the one on the left).

Let us now understand a little more in depth *why* this happens. The idea is to think about where exactly the standard basis vectors for $\mathbb{R}^p$ go under the composite.

Suppose $\vec{e_k}$ is the k^{th} standard basis vector in $\mathbb{R}^p$. We know that, under the transformation T_B, $\vec{e_k}$ gets mapped to the k^{th} column of B. In particular, the j^{th} coordinate of the image $T_B(\vec{e_k})$ is the matrix entry b_{jk}. Explicitly:

$$T_B(\vec{e_k}) = \sum_{j=1}^{n} b_{jk} \vec{e_j}$$

We now want to apply T_A to both sides. We get:

$$T_A(T_B(\vec{e_k})) = T_A\left(\sum_{j=1}^{n} b_{jk} \vec{e_j}\right)$$

We know that T_A is linear, so this can be rewritten as:

$$T_A(T_B(\vec{e_k})) = \sum_{j=1}^{n} (b_{jk} T_A(\vec{e_j}))$$

$T_A(\vec{e_j})$ is just the j^{th} column of the matrix A, so we get:

$$T_A(T_B(\vec{e_k})) = \sum_{j=1}^{n} b_{jk} \left(\sum_{i=1}^{m} a_{ij} \vec{e_i}\right)$$

The i^{th} coordinate in this is thus:

$$\sum_{j=1}^{n} b_{jk} a_{ij}$$

Thus, we get that:

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$$

3.3. **Chaining and multiple intermediate paths.** The intuitive way of thinking of matrix multiplication is as follows. The value c_{ik} roughly describes the strength of the total pathway from $\vec{e_k}$ in $\mathbb{R}^p$ to $\vec{e}_i$ in $\mathbb{R}^m$. This "total pathway" is a sum of pathways via intermediate routes, namely the basis vectors for $\mathbb{R}^n$. The strength of the pathway via an intermediate vector $\vec{e_j}$ is $a_{ij} b_{jk}$. The total strength is the sum of the individual strengths.

This is very similar (and closely related) to the relationship between the chain rule for differentiation for a composite of functions of one variable and the chain rule for differentiation for a composite of functions of multiple variables.

3.4. **A real-world example.** Suppose there is a bunch of m people, a bunch of n foodstuffs that each person could consume, and a bunch of p nutrients. Let $A = (a_{ij})$ be the "person-foodstuff" matrix. This is the matrix whose rows are indexed by people and columns are indexed by foodstuffs, and where the entry in a particular row and particular column is the amount of the column foodstuff that the row person consumes. A is a $m \times n$ matrix. Note that we can set units in advance, but the units do not need to be uniform across the matrix entries, i.e., we could choose different units for different foodstuffs. It does make sense to use the same units for a given foodstuff across multiple persons, for reasons that will become clear soon.

Let $B = b_{jk}$ be the "foodstuff-nutrient" matrix. This is the matrix whose rows are indexed by foodstuffs and columns are indexed by nutrients, and where the entry in a particular row and particular column is the amount of the column nutrient contained per unit of the row foodstuff. B is a $n \times p$ matrix. Note that we want the units used for measuring the foodstuff to be the same in the matrices A and B.

Now, let us say we want to construct a "person-nutrient" matrix $C = (c_{ik})$. The rows are indexed by persons. The columns are indexed by nutrients. The entry in the i^{th} row and k^{th} column represents the amount of the k^{th} nutrient consumed by the i^{th} person. C is a $m \times p$ matrix. We now proceed to explain why $C = AB$, and in the process, provide a concrete illustration that justifies our definition of matrix multiplication.

The $(ik)^{th}$ entry of C is the amount of the k^{th} nutrient consumed by the i^{th} person. Now, the i^{th} person might get his fix of the k^{th} nutrient from a combination of multiple foodstuffs. In a sense, each foodstuff gives the i^{th} person some amount of the k^{th} nutrient (though that amount may well be zero). What is the contribution of each foodstuff?

Consider the j^{th} foodstuff, with $1 \le j \le n$. The i^{th} person consumes a_{ij} units of the j^{th} foodstuff. Each unit of the j^{th} foodstuff contains b_{jk} units of the k^{th} nutrient. The total amount of the k^{th} nutrient consumed by the i^{th} person via the j^{th} foodstuff is the product $a_{ij} b_{jk}$.

We now need to sum this up over all the foodstuffs. We thus get:

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$$

This agrees with our definition of matrix multiplication, vindicating its utility.

3.5. **Matrix multiplication requires dimensions to match.** Matrix multiplication *only* makes sense when the number of columns in the first matrix equals the number of rows in the second matrix. Otherwise, it *just doesn't make sense.*

For instance, if m, n, and p are all different, A is a $m \times n$ matrix, and B is a $n \times p$ matrix, then AB makes sense whereas BA does not make sense. On the other hand, if $m = p$, then both AB and BA make sense. However, whereas AB is a $m \times (m = p)$ square matrix, BA is a $n \times n$ square matrix.

3.6. **Multiplication by the identity matrix.** We denote by I_n or Id_n the $n \times n$ identity matrix. The following are true:

- $I_n A = A$ for any $n \times p$ matrix A.
- $A I_n = A$ for any $m \times n$ matrix A.
- $I_n A = A I_n = A$ for any $n \times n$ matrix A.

3.7. **Reinterpreting the inverse in terms of matrix multiplication.** We had earlier defined a notion of *inverse* of a bijective linear transformation. We can interpret inverse easily in terms of matrix multiplication. The inverse of a $n \times n$ matrix A, denoted A^{-1}, satisfies the condition that both AA^{-1} and $A^{-1}A$ are equal to the $n \times n$ identity matrix. The $n \times n$ identity matrix is written as I_n or Id_n.

4. Computation of matrix products

4.1. **Time and space complexity of naive matrix multiplication.** The naive matrix multiplication procedure is to compute each cell entry by computing all the products involved and then summing them up. If we are considering the product of a $m \times n$ matrix with a $n \times p$ matrix, the product is a $m \times p$ matrix, with each matrix entry a sum of n products. This means that each matrix entry, naively, requires n multiplications and $(n-1)$ additions. The total number of multiplications is thus mnp and the total number of additions is $mp(n-1)$.

In the special case that we are multiplying two $n \times n$ square matrices, we require n^3 multiplications and $n^2(n-1)$ additions. The arithmetic complexity of naive matrix multiplication for $n \times n$ square matrices is thus $\Theta(n^3)$.

4.2. **Fast matrix multiplication: the idea.** *I may not get time to cover this in class.*

One key thing to note is that to multiply matrices, the only things we finally care about are the sums of products. We do not care to know the individual products at all. If there is some way of calculating the sums of products without calculating the individual products, that might be better.

Consider a simpler but related example. The expression here is not the expression that appears in a matrix product, but it offers a simple illustration of an idea whose more complicated version appears in fast matrix multiplication algorithms.

Consider the following function of four real numbers a, b, c, and d:

$$(a, b, c, d) \mapsto ac + bd + ad + bc$$

The naive approach to computing this function is to calculate separately the four products ac, bd, ad, and bc, and then add them up. This requires 4 multiplications and 3 additions.

An alternative approach is to note that this is equivalent to:

$$(a, b, c, d) \mapsto (a + b)(c + d)$$

Calculating the function in this form is considerably easier. It requires two additions and one multiplication. When we use the new method of calculation, we end up not getting to know the values of the individual products ac, bd, ad, and bc. But we were not interested in these anyway. Our interest was in the sum, which can be computed through this alternate method.

Some algorithms for fast matrix multiplication rely on the underlying idea here. Unfortunately, the idea needs a lot of tweaking to effectively be used for fast matrix multiplication. If you are interested, look up the *Strassen algorithm* for fast matrix multiplication. The Strassen algorithm combines a divide-and-conquer strategy with an approach that does a 2×2 matrix using only 7 multiplications instead of the 8 multiplications used in the naive algorithm. The saving accumulates when combined with the divide-and-conquer strategy, and we end up with an arithmetic complexity of $\Theta(n^{\log_2 7})$. The number $\log_2 7$ is approximately 2.8, which is somewhat of a saving over the 3 seen in naive matrix multiplication.

4.3. **Parallelizability.** Naive matrix multiplication is massively parallelizable as long as it is easy for multiple processors to access specific entries from the matrix. Explicitly, each entry of the product matrix can be computed by a different processor, since the computations of the different entries are independent under naive matrix multiplication. To compute the product of a $m \times n$ matrix and a $n \times p$ matrix, whereby the output is a $m \times p$ matrix, we can calculate each cell entry of the output using a separate processor which computes a dot product. Each processor does n multiplications and $n - 1$ additions, so the arithmetic complexity is $2n - 1$ operations, or $\Theta(n)$.

However, if we have access to more processors, we can do even better. Let's take a short detour into how to parallelize addition.

4.4. **Parallelizing addition.** Suppose we want to add n numbers using a (replicable) black box that can add two numbers at a time. The naive approach uses $n - 1$ steps: add the first two numbers, then add the third number to that, then add the fourth number to that, and so on. Without access to parallel computing, this is the best we can do. A parallel algorithm can do better using a *divide and conquer* strategy.

The one-step strategy would be to divide the list into two roughly equal halves, have two separate processors sum them up, and then add up their respective totals. This makes sense in the real world. Note that the time seems to have halved, but there is an extra step of adding up the two halves.

The smart approach is to then recurse: divide each half again into two halves, and so on. Ultimately, we will be left with finding sums of pairs of elements, then adding up those pairs, and so on. Effectively, our computation tree is a binary tree where the leaves are the values to be added. The depth of this tree describes the time complexity of the algorithm, and it is now $\Theta(\log_2 n)$.

4.5. **Extreme parallel processing for matrix multiplication.** The above idea for parallelizing addition allows us to compute each entry of the product matrix in time $\Theta(\log_2 n)$ (in terms of the depth of the arithmetic computation tree). Since all the entries are being computed in parallel, the arithmetic complexity of this massively parallelized algorithm is $\Theta(\log_2 n)$. Note how this is massively different from ordinary naive matrix multiplication, which takes times $\Theta(n^3)$, and in fact is better than any possible non-parallel algorithm, since any such algorithm must take time $\Omega(n^2)$ just to fill in all the entries.

5. Some particular cases of matrix multiplication

5.1. **The inner product or dot product.** A matrix product AB where A is a single row matrix and B is a single column matrix, and where the number of columns of A is the number of rows of B, becomes a dot product. Explicitly, the matrix product of a $1 \times n$ matrix and a $n \times 1$ matrix is simply their dot product as vectors, written as a 1×1 matrix.

5.2. **The outer product or Hadamard product.** A matrix product AB where A is a single column matrix and B is a single row matrix gives a rectangular matrix with as many rows as in A and as many columns as in B. Explicitly, the matrix product of a $m \times 1$ matrix and a $1 \times n$ matrix is a $m \times n$ matrix. The entries of the product can be thought of as being constructed from a *multiplication table*. The entry in the i^{th} row and j^{th} column is the product of the i^{th} entry of A and the j^{th} entry of B.

This particular type of product is called the *outer product* or *Hadamard product*.

5.3. **Multiplication of diagonal matrices.** Suppose A and B are both diagonal $n \times n$ matrices. Then, the matrix product AB is also a diagonal matrix, and each diagonal entry is obtained as a product of the corresponding diagonal entry of A and of B. Explicitly, if $C = AB$, then:

$$c_{ii} = a_{ii}b_{ii}$$

and

$$c_{ij} = 0 \text{ for } i \neq j$$

In terms of composition of linear transformations, each diagonal matrix involves scaling the basis vectors by (possibly) different factors. Composing just means composing the operations separately on each basis vector. In a diagonal transformation, the basis vectors do not get mixed up with each other. So, we do not have to worry about adding up multiple pathways.

6. Products of more than two matrices

6.1. **Associativity of matrix multiplication.** Suppose A is a $m \times n$ matrix, B is a $n \times p$ matrix, and C is a $p \times q$ matrix. Then, AB is a $m \times p$ matrix and BC is a $n \times q$ matrix. Thus, both $(AB)C$ and $A(BC)$ make sense, and they are both $m \times q$ matrices. The associativity law for matrix multiplication states that in fact, they are equal as matrices.

Intuitively, this makes sense, because it is just adding up weights over a lot of different pathways. Explicitly, the $(il)^{th}$ entry of the product is:

$$\sum_{j,k} a_{ij} b_{jk} c_{kl}$$

6.2. **Explaining associativity in terms of interpretation as linear transformations.** With the setup as above, we have that:

- The $m \times n$ matrix A represents a linear transformation $T_A : \mathbb{R}^n \to \mathbb{R}^m$.
- The $n \times p$ matrix B represents a linear transformation $T_B : \mathbb{R}^p \to \mathbb{R}^n$.
- The $p \times q$ matix C represents a linear transformation $T_C : \mathbb{R}^q \to \mathbb{R}^p$.

Then, we can say that:

- The $m \times p$ matrix AB represents the linear transformation $T_{AB} = T_A \circ T_B : \mathbb{R}^p \to \mathbb{R}^m$.
- The $n \times q$ matrix BC represents the linear transformation $T_{BC} = T_B \circ T_C : \mathbb{R}^q \to \mathbb{R}^n$.

Finally, we obtain that:

- The $m \times q$ matrix $(AB)C$ represents the linear transformation $T_{(AB)C} = (T_A \circ T_B) \circ T_C : \mathbb{R}^q \to \mathbb{R}^m$.
- The $m \times q$ matrix $A(BC)$ represents the linear transformation $T_{A(BC)} = T_A \circ (T_B \circ T_C) : \mathbb{R}^q \to \mathbb{R}^m$.

We know that function composition is associative. Therefore, the linear transformations $(T_A \circ T_B) \circ T_C$ and $T_A \circ (T_B \circ T_C)$ are equal. Hence, their corresponding matrices $(AB)C$ and $A(BC)$ are also equal. This gives another proof of the associativity of matrix multiplication.

6.3. **Associativity of matrix multiplication in a real-world context.** Consider the following four types of entities:

- A *community* is a set of people living in a geographical area. Suppose there are m communities.
- A *diet* is a particular specification ofhow much of each foodstuff an individual should consume daily. For instance, there may be a "standard paleo diet" or a "South Beach diet" or an "Atkins diet." Suppose there are N diets.
- A *foodstuff* is a type of food. Suppose there are p foodstuffs.
- A *nutrient* is something that a person needs for survival or human flourishing in his or her diet. Suppose there are q nutrients.

We can now construct matrices:

- A $m \times n$ matrix A called the "community-diet matrix" whose rows are indexed by communities and columns by diets. The entry in the i^{th} row and j^{th} column specifies the fraction of people in the i^{th} community that follows the j^{th} diet.
- A $n \times p$ matrix B called the "diet-foodstuff matrix" whose rows are indexed by diets and columns by foodstuffs. The entry in the j^{th} row and k^{th} column specifies the amount of the k^{th} food that is to be consumed daily in the j^{th} diet.
- A $p \times q$ matrix C called the "foodstuff-nutrient matrix" whose rows are indexed by foodstuffs and columns by nutrients. The entry in the k^{th} row and l^{th} column specifies the amount of the l^{th} nutrient per unit of the k^{th} foodstuff.

The matrix products are as follows:

- The $m \times p$ matrix AB is the "community-foodstuff matrix" whose rows are indexed by communities and columns by foodstuffs. The entry in the i^{th} row and k^{th} column specifies the average amount of the k^{th} foodstuff consumed in the i^{th} community.
- The $n \times q$ matrix BC is the "diet-nutrient matrix" whose rows are indexed by diets and columns by nutrients. The entry in the j^{th} row and l^{th} column specifies the total amount of the l^{th} nutrient in the j^{th} diet.

- The $m \times q$ matrix $A(BC) = (ABC)C$ is the "community-nutrient matrix" whose rows are indexed by communities and columns by nutrients. The entry in the i^{th} row and l^{th} column specifies the total amount of the l^{th} nutrient in the i^{th} community.

6.4. **Powers of a matrix.** Note that in order to multiply a matrix by itself, its number of columns must equal its number of rows, i.e., it must be a square matrix. If A is a $n \times n$ square matrix, we can make sense of $A^2 = AA$. Since matrix multiplication is associative, we can uniquely interpret higher powers of A as well, so A^r is the product of A with itself r times. In particular, $A^3 = A^2A = AA^2$. We also define A^0 to be the identity matrix.

This will make sense after you read the section on inverses: If A is invertible, then we can also make sense of *negative* powers of A. We define A^{-n} (for n a natural number) as $(A^{-1})^n$, and it is also equal to $(A^n)^{-1}$.

6.5. **Computing powers of a matrix.** We will discuss later how to invert a matrix. For now, let us consider how to find the positive powers of a matrix.

The *naive* approach to computing A^r where A is a $n \times n$ matrix is to just keep multiplying by A as many times as necessary. Recall that each naive matrix multiplication takes $n^2(2n-1)$ arithmetic operations (n^3 multiplications and $n^2(n-1)$ additions). If we naively multiply, we need $(r-1)$ matrix multiplications, so the total number of operations is $n^2(2n-1)(r-1)$. In order terms, it is $\Theta(n^3 r)$.

There is a trick to speeding it up at both ends. First, we can speed up individual matrix multiplication using fast matrix multiplication. Second, we can use repeated squaring. For instance, to compute A^8, we can square A three times. In general, the number of matrix multiplications that we need to do if chosen smartly is $\Theta(\log_2 r)$.

Note that the minimum number of multiplications that we need to carry out through repeated squaring to compute A^r is *not* an increasing, or even a non-decreasing, function of r. When r is a power of 2, the process of using repeated squaring is particularly efficient: we just square $\log_2 r$ times. If r is *not* a power of 2, we need to perform some repeated squarings, but we *also* need to perform other operations that multiply things together. For instance, to compute A^5, we do $(A^2)^2A$.

The general strategy is to compute A^{2^l} where 2^l is the largest power of 2 that is less than or equal to r, and in the process store all the intermediate A^{2^k}, $0 \le k \le l$. Now, A^r is a product of some of these matrices. Just multiply them all. The "worst case" scenario is when r is one less than a power of 2. In this case, $r = 2^{l+1} - 1 = 1 + 2 + \dots + 2^l$ so we need to perform a total of $2l$ multiplications (l repeated squarings and then l "piecing together" multiplications). For instance, for A^7, we first compute A^2 (one multiplication), then compute $(A^2)^2 = A^4$ (second multiplication), then multiply A^4A^2A (two more operations).

The explicit formula for the minimum number of matrix multiplications needed is as follows: write r in base 2 notation (i.e., binary notation). The number of multiplications needed is:

(Total number of digits) + (Number of digits that are equal to 1) - 2

The reason is as follows: we need to do as many repeated squarings as (total number of digits) - 1, and then do as many multiplications as (number of digits that are equal to 1) - 1.

The first few examples are in the table below:

r	r in base 2	total number of digits	number of 1s	number of matrix multiplications
1	1	1	1	0
2	10	2	1	1
3	11	2	2	2
4	100	3	1	2
5	101	3	2	3
6	110	3	2	3
7	111	3	3	4
8	1000	4	1	3
9	1001	4	2	4

7. Inverting a matrix: the meaning and the method

7.1. How to invert a matrix: augment with the identity matrix.

7.1. How to invert a matrix: augment with the identity matrix. The following is a procedure for inverting a $n \times n$ matrix A:

- Write down the matrix A, and augment it with the $n \times n$ identity matrix.
- Complete the conversion of the matrix to reduced row-echelon form, and perform the same operations on the augmenting matrix.
- If the reduced row-echelon form of the original matrix is *not* the identity matrix, the matrix is non-invertible.
- If the reduced row-echelon form of the matrix is the identity matrix, then the matrix on the augmenting side is precisely the inverse matrix we are looking for.

There are two ways of explaining why this works.

First, think of the matrix A as the matrix of a linear transformation T. The matrix A^{-1} has columns equal to the vectors $T^{-1}(\vec{e}_1)$, $T^{-1}(\vec{e}_2)$, $\dots$, $T^{-1}(\vec{e}_n)$. To find $T^{-1}(\vec{e}_1)$ is tantamount to solving $A\vec{x} = \vec{e}_1$, which can be obtained by augmenting A with $\vec{e}_1$ and solving. Similarly, to find $T^{-1}(\vec{e}_2)$ is tantamount to solving $A\vec{x} = \vec{e}_2$, which can be obtained by augmeting A with $\vec{e}_2$ and solving. In order to do all these computations together, we need to augment A with all of the columns $\vec{e}_1$, $\vec{e}_2$, and so on. This means augmenting A with the identity matrix. After A is converted to rref, the respective columns on the augmenting side are $T^{-1}(\vec{e}_1)$, $T^{-1}(\vec{e}_2)$, $\dots$, $T^{-1}(\vec{e}_n)$. That's exactly what we want of A^{-1}.

The second approach is more subtle and involves thinking of the elementary row operations as matrices that are being done and undone. This is a separate approach that we will consider later.

7.2. Arithmetic complexity of matrix inversion. The arithmetic complexity of matrix inversion is the same as the arithmetic complexity of Gauss-Jordan elimination (actually, we need to do about twice as many operations, since we have many more augmenting columns, but the order remains the same). Explicitly, for a $n \times n$ matrix:

- The worst-case arithmetic complexity (in terms of time) of inverting the matrix is $\Theta(n^3)$ and the space requirement is $\Theta(n^2)$.
- A parallelized version of the inversion algorithm can proceed in $\Theta(n)$ time with access to enough processors.

7.3. Particular cases of matrix inversion. The following is information on inverting matrices that have a particular format:

- The inverse of a $n \times n$ diagonal matrix with all entries nonzero is a $n \times n$ diagonal matrix where each diagonal entry is inverted in place. Note that if any diagonal entry is zero, the matrix does not have full rank and is therefore non-invertible.
- The inverse of a $n \times n$ scalar matrix with scalar value λ is a scalar matrix with scalar value $1/\lambda$.
- The inverse of a $n \times n$ upper-triangular matrix where all the diagonal entries are nonzero is also an upper-triangular matrix where all the diagonal entries are nonzero, and in fact the diagonal entries of the inverse matrix are obtained by inverting the original diagonal entries in place (we need to use the augmented identity method to compute the other entries of the inverse matrix). An analogous statement is true for lower-triangular matrices.
- *This will make sense after you've seen permutation matrices*: The inverse of a $n \times n$ permutation matrix is also a $n \times n$ permutation matrix corresponding to the inverted permutation.

7.4. Using the matrix inverse to solve a linear system. Suppose A is an invertible $n \times n$ matrix. Solving a linear system with coefficient matrix A and augmenting column (i.e., output vector) $\vec{y}$ means solving the following vector equation for $\vec{x}$:

$$A\vec{x} = \vec{y}$$

We earlier saw that this can be done using Gauss-Jordan elimination for the augmented matrix $[A \mid \vec{y}]$. We can now think of it more conceptually. Multiply both sides of this vector equation on the left by the matrix A^{-1}:

$$A^{-1}(A\vec{x}) = A^{-1}\vec{y}$$

Solving, we get:

$$\vec{x} = A^{-1}\vec{y}$$

Suppose now that A is known in advance, so that we can pre-compute A^{-1}. Then, in order to solve the linear system

$$A\vec{x} = \vec{y}$$

we simply need to execute the vector-matrix multiplication

$$A^{-1}\vec{y}$$

This is multiplication of a $n \times n$ matrix and a $n \times 1$ vector, so by the naive matrix multiplication algorithm, the time taken for this is $\Theta(n^2)$.

Did we already know this? Yes, by a different name. We had not earlier conceptualized A^{-1} as a transformation. Rather, if A was known in advance, we carried out Gauss-Jordan elimination on A, then "stored the steps" used, so that then on being told the output vector $\vec{y}$, we could apply the steps one after another to $\vec{y}$ and obtain back the input vector $\vec{x}$. Now, instead of applying the steps sequentially, we just multiply by another matrix, A^{-1}. The matrix A^{-1} stores the information somewhat differently.

One downside is that we can use A^{-1} *only* in the situation where the coefficient matrix is a square matrix and is invertible. On the other hand, Gauss-Jordan elimination as a process works in general. There are generalizations of the inverse to other cases that we will hint at shortly.

One upside of using A^{-1} is that matrix-vector multiplication can be massively parallelized down to $\Theta(\log_2 n)$ arithmetic complexity (that is the complexity of the computation tree). Storing the sequence of row operations that need to be applied does work, but it is harder to parallelize, because the *sequence* in which the operations are applied is extremely important.

8. One-sided invertibility

We will not cover this section in class.

Suppose A is a $m \times n$ matrix and B is a $n \times m$ matrix such that AB is the $m \times m$ identity matrix, but $m \neq n$. In that case, the following are true:

- We must have that $m < n$.
- A has full row rank (rank m), i.e., the corresponding linear transformation is surjective.
- B has full column rank (rank m), i.e., the corresponding linear transformation is injective.

Interestingly, the converse results are also true, namely:

- If a $m \times n$ matrix A has full row rank m, there exists a $n \times m$ matrix B such that AB is the $m \times m$ identity matrix.
- If a $n \times m$ matrix B has full column rank m, there exists a $m \times n$ matrix A such that AB is the $m \times m$ identity matrix.

Both of these can be thought in terms of solving systems of simultaneous linear equations. We will see the proofs of these statements later. (If you are interested, look up the *Moore-Penrose inverse*).

9. Algebraic rules governing matrix multiplication and inversion

9.1. **Distributivity of multiplication.** Matrix multiplication is both left and right distributive. Explicitly:

$$\begin{aligned} A(B+C) &= AB + AC \\ (A+B)C &= AC + BC \end{aligned}$$

The equality is conditional both ways: if the left side in any one equality of this sort makes sense, so does the right side, and they are equal. In particular, the first law makes sense if B and C have the same row and column counts as each other, and the column count of A equals the row count of B and C. The second law makes sense if A and B have the same row count and the same column count and both column counts equal the row count of C.

9.2. **The interplay between multiplication and inversion.** The basic algebra relating matrix multiplication and inversion follows from the algebraic manipulation rules related to an abstract structure type called a *group*. In other words, all the algebraic rules we discuss here hold in arbitrary groups. Nonetheless, we will not think of it in that generality.

Matrix inversion is *involutive* with respect to multiplication in the following sense. These two laws hold:

- $(A^{-1})^{-1} = A$ for any invertible $n \times n$ matrix A.
- $(AB)^{-1} = B^{-1}A^{-1}$ for any two invertible $n \times n$ matrices A and B (note that A and B could be equal)
- $(A_1A_2 \dots A_r)^{-1} = A_r^{-1}A_{r-1}^{-1} \dots A_1^{-1}$ for any invertible $n \times n$ matrices A_1, A_2, $\dots$, A_r.

9.3. **Commutativity and the lack thereof.** First off, there are situations where we have matrices A and B such that AB makes sense but BA does not make any sense. For instance, if A is a $m \times n$ matrix and B is a $n \times p$ matrix with $m \neq p$, this is exactly what happens. In particular, $AB = BA$, far from being true, makes no sense.

Second, there can be situations where both AB and BA are defined but their row counts and/or column counts don't match. For instance, if A is a $m \times n$ matrix and B is a $n \times m$ matrix, then AB is a $m \times m$ matrix and BA is a $n \times n$ matrix. In particular, $AB = BA$ again makes no sense.

The only situation where the question of whether $AB = BA$ can be legitimately asked is the case where A and B are both square matrices of the same size, say both are $n \times n$ matrices. In such a situation, $AB = BA$ may or may not be true. It depends on the choice of matrices. In the case that $AB = BA$, we say that the matrices A and B *commute*.

Recall that matrix multiplication corresponds to composing the associated linear transformations. $AB = BA$ corresponds to the statement that the order of *composition* for the corresponding linear transformations does not matter. We will have a lot more to say about the conditions under which two linear transformations commute, but briefly:

- Any matrix commutes with itself, its inverse (if the inverse exists) and with powers of itself (positive, and also negative if it is invertible).
- Any two $n \times n$ diagonal matrices commute with each other.
- A $n \times n$ scalar matrix commutes with every possible $n \times n$ matrix.
- The set of matrices that commute with any matrix A is closed under addition and multiplication.

10. Noncommuting matrix examples and finite state automata

The explanation given here is not adequate for understanding the material. So it is strongly recommended that you pay attention in class.

Suppose $f : \{0, 1, 2, \dots, n\} \to \{0, 1, 2, \dots, n\}$ is a function with $f(0) = 0$. Consider a linear transformation T_f given by $T_f(\vec{e}_i) = \vec{e}_{f(i)}$ if $f(i) \neq 0$ and $T(\vec{e}_i) = \vec{0}$ if $f(i) = 0$.. The matrix for T has at most one nonzero entry in each column, and if there is a nonzero entry, then that entry is 1.

If the range of f is $\{0, 1, 2, \dots, n\}$ then the matrix for T is a permutation matrix: exactly one nonzero entry, with value 1, in each row and exactly one nonzero entry, with value 1, in each column. Permutation matrices are quite important for a number of purposes. Our interest right now, though, is in the other kinds of matrices, because they help us construct simple examples of noncommuting matrices.

Pictorially, these kinds of linear transformations can be described using what are called *finite state automata*. The finite state automaton is a directed graph with one edge out of every node. The nodes here correspond to the standard basis vectors and the zero vector. An edge from one node to another means that the latter node is the image of the former node under the linear transformation. There is a loop at the zero node.

Composing the linear transformations is equivalent to composing the corresponding functions. In symbols:

$$T_{f \circ g} = T_f \circ T_g$$

If we denote by M_f the matrix for T_f, then we obtain:

$$M_{f \circ g} = M_f M_g$$

What this means is that to compose two linear transformations of this sort, we can compose the corresponding functions by "following the arrows" for their finite state automaton diagrams.

To find a power of a linear transformation of this sort, we simply keep doing multiple steps along the automaton. By various finiteness considerations, the powers of the matrix will eventually start repeating. Further, once there is a repetition, the pattern after that will involve repetition. For instance, suppose $A^{13} = A^{17}$ is the first repetition. Then the sequence $A^{13}, A^{14}, A^{15}, A^{16}$, will be repeated *ad infinitum*: $A^{17} = A^{13}, A^{18} = A^{14}$, and so on.

Some special cases are worth mentioning:

- If $A = A^2$, we say that A is *idempotent*, and the corresponding linear transformation is a *projection*. Certain kinds of projections are termed *orthogonal projections*. We will explore the terminology later.
- If $A^r = 0$ for some r, we say that A is *nilpotent*.

The guarantee of a repetition is specific to linear transformations of this sort, and is not true for linear transformations in general.

Consider the case $n = 2$. We have the following examples. We are constructing the matrix in each example as follows. For the i^{th} column for the matrix, enter the vector that is $T_f(\vec{e}_i) = \vec{e}_{f(i)}$. For instance, if $f(1) = 2$, the first column is the vector $\vec{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

$f(1)$ and $f(2)$ (in order)	Matrix A	Smallest $0 \le k < l$ s.t. $A^k = A^l$	Type of matrix
0 and 0	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	1 and 2	zero matrix
0 and 1	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	2 and 3	nilpotent (square is zero)
0 and 2	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	1 and 2	idempotent (orthogonal projection)
1 and 0	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	1 and 2	idempotent (orthogonal projection)
1 and 1	$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$	1 and 2	idempotent (non-orthogonal projection)
1 and 2	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	0 and 1	identity matrix
2 and 0	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$	2 and 3	nilpotent (square is zero)
2 and 1	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	0 and 2	permutation matrix (square is identity)
2 and 2	$\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$	1 and 2	idempotent (non-orthogonal projection)

These automata can be used to construct examples of noncommuting pairs of matrices. For instance, suppose we want matrices A and B such that $AB = 0$ but $BA \neq 0$. We try to construct functions $f, g : \{0, 1, 2\} \to \{0, 1, 2\}$ with $f(0) = g(0) = 0$ such that $f \circ g$ maps everything to 0 but $g \circ f$ does not. A little thought reveals that we can take:

$$f(0) = 0, f(1) = 0, f(2) = 1$$

$$g(0) = 0, g(1) = 1, g(2) = 0$$

To write the matrices explicitly, we look at the above and write in the i^{th} column the vector $\vec{e}_{f(i)}$. We get:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

The matrix products are:

$$AB = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

and:

$$BA = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

We will discuss more about matrix algebra and examples and counterexamples later.

GEOMETRY OF LINEAR TRANSFORMATIONS

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 2.2.

Executive summary

(1) There is a concept of *isomorphism* as something that preserves essential structure or feature, where the concept of isomorphism depends on what feature is being preserved.
(2) There is a concept of *automorphism* as an isomorphism from a structure to itself. We can think of automrohpisms of a structure as *symmetries* of that structure.
(3) Linear transformations have already been defined. An *affine linear transformation* is something that preserves lines and ratios of lengths within lines. Any affine linear transformation is of the form $\vec{x} \mapsto A\vec{x} + \vec{b}$. For the transformation to be linear, we need $\vec{b}$ to be the zero vector, i.e., the transformation must send the origin to the origin. If A is the identity matrix, then the affine linear transformation is termed a *translation*.
(4) A linear *isomorphism* is an invertible linear transformation. For a linear isomorphism to exist from $\mathbb{R}^m$ to $\mathbb{R}^n$, we must have $m = n$. An affine linear isomorphism is an invertible affine linear transformation.
(5) A linear automorphism is a linear isomorphism from $\mathbb{R}^n$ to itself. An affine linear automorphism is an affine linear isomorphism from $\mathbb{R}^n$ to itself.
(6) A self-isometry of $\mathbb{R}^n$ is an invertible function from $\mathbb{R}^n$ to itself that preserves Euclidean distance. Any self-isometry of $\mathbb{R}^n$ must be an affine linear automorphism of $\mathbb{R}^n$.
(7) A self-homothety of $\mathbb{R}^n$ is an invertible function from $\mathbb{R}^n$ to itself that scales all Euclidean distances by a factor of λ, where λ is the factor of homothety. We can think of self-isometries precisely as the self-homotheties by a factor of 1. Any self-homothety of $\mathbb{R}^n$ must be an affine linear automorphism of $\mathbb{R}^n$.
(8) Each of these forms a group: the affine linear automorphisms of $\mathbb{R}^n$, the linear automorphisms of $\mathbb{R}^n$, the self-isometries of $\mathbb{R}^n$, the self-homotheties of $\mathbb{R}^n$.
(9) For a linear transformation, we can consider something called the *determinant*. For a 2×2 linear transformation with matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the determinant is $ad - bc$.

We can also consider the *trace*, defined as $a + d$ (the sum of the diagonal entries).
(10) The trace generalizes to $n \times n$ matrices: it is the sum of the diagonal entries. The determinant also generalizes, but the formula becomes more complicated.
(11) The determinant for an affine linear automorphism can be defined as the determinant for its linear part (the matrix).
(12) The sign of the determinant being positive means the transformation is orientation-preserving. The sign of the determinant being negative means the transformation is orientation-reversing.
(13) The magnitude of the determinant gives the factor by which volumes are scaled. In the case $n = 2$, it is the factor by which areas are scaled.
(14) The determinant of a self-homothety with factor of homothety λ is $\pm\lambda^n$, with the sign depending on whether it is orientation-preserving or orientation-reversing.
(15) Any self-isometry is volume-preserving, so it has determinant ± 1, with the sign depending on whether it is orientation-preserving or orientation-reversing.

(16) For $n = 2$, the orientation-preserving self-isometries are precisely the translations and rotations. The ones fixing the origin are precisely rotations centered at the origin. These form groups.
(17) For $n = 2$, the orientation-reversing self-isometries are precisely the reflections and glide reflections. The ones fixing the origin are precisely reflections about lines passing through the origin.
(18) For $n = 3$, the orientation-preserving self-isometries fixing the origin are precisely the rotations about axes through the origin. The overall classification is more complicated.

1. Geometry of linear transformations

1.1. Geometry is secondary, but helps build institutions.

Geometry is secondary, but helps build institutions. Our focus on linear transformations so far has been *information-centric*: they help reframe existing pieces of information in new ways, allowing us to extract things that are valuable to us. This information-centric approach undergirds the applicability of linear algebra in the social sciences.

We will now turn to a geometric perspective on linear transformations, restricting largely to the case $n = 2$ for many aspects of our discussion. A lot of this geometry is more parochial than the information-centric approach, and is not too important for the application of linear algebra to the social sciences. The main reason the geometry is valuable to us is that it can help build intuition regarding what is going on with the algebraic operations and hence offer another way of "sanity checking" our algebraic intuitions. As a general rule, always try to come up with reasons that are algebraic and information-centric, but in cases where we can perform sanity checks using geometric intuitions, use them.

1.2. **Composition of transformations.** If $f : A \to B$ is a function and $g : B \to C$ is a function, we can make sense of the composite $g \circ f$. The key feature needed to make sense of the composite is that the co-domain (the target space) of the function applied first (which we write on the right) must equal the domain (the starting space) of the function applied next (which we write later).

This means that if we have a collection of maps all from a space to *itself*, it makes sense to compose any two maps in the collection. We can even compose more than two maps if necessary.

The geometry of linear transformations that we discuss here is in the context of transformations from $\mathbb{R}^n$ to $\mathbb{R}^n$. Here, we can compose, and if the transformations are bijective, also invert.

A small note is important here. We often see maps from $\mathbb{R}^n$ to $\mathbb{R}^n$ where, even though both the domain and co-domain space have the same dimension, they do not represent the "same space" conceptually. For instance, one side may be measuring masses, while the other side may be measuring prices. In this case, composing multiple such maps does not make sense because even though the domain and co-domain are mathematically $\mathbb{R}^n$, they are conceptually different.

2. Particular kinds of transformations in linear algebra

2.1. **The concept of isomorphism and automorphism.** We will briefly describe two central mathematical concepts called *isomorphism* and *automorphism*. You are not expected to understand these concepts, but they help demystify some of the following discussion.

Understanding the concept of *isomorphism* is central to abstraction and to human intelligence, even though the word is not too well-known outside of mathematics and philosophy. When looking at different structures, we may be interested in whether they have *a certain feature in common*. For instance, when looking at sets, we may be interested in judging them by the number of elements in them. If we care only about size, then in our tunnel vision, all other aspects of the set are irrelevant. For our purposes, then, a set of three rabbits is effectively the same as a set of three lions, or a set of three stars. As another related example, suppose the only thing you care about potential mates in the dating market is their bank balance. In that case, two potential mates with the same bank balance are effectively the same, i.e., they are isomorphic.

This kind of abstraction is crucial to humans being able to understand numbers in the first place. If you crossed that hurdle back at the age of 3, 4, 5, or 6 (or whenever you understood the idea of counting), it's high time you went a step further.

One way of capturing the idea that two sets have the same size is as follows. We say that two sets have the same size if it is possible to construct a bijective function from one set to the other. In fact, roughly speaking, this is the *only* way to define the concept of "same size". In other words, we can *define* the size (technical term: cardinality) of a set as that attribute that is preserved by bijections and is different for sets if there is no bijection between them.

Another way of framing this is to christen bijective functions as *isomorphisms of sets*. In other words, a function from a set to a (possibly same, possibly different) set is termed an isomorphism of sets if it is a bijective function of sets. "Iso+morph" stands for "same shape" and signifies that the bijective function preserves the shape.

Why is this important? Instead of caring only about size, we can ratchet up our caring levels to care about more structural aspects of the sets we are dealing with. If we do so, our definition of "isomorphism" will become correspondingly more stringent, since it will require preserving more of the structure.

A related notion to isomorphism is that of *automorphism*. An automorphism is an isomorphism (in whatever sense the term is being used) from a set to itself.

Whatever our definition of isomorphism and automorphism, the identity map from a set to itself is always an automorphism.

Non-identity automorphisms from a set to itself signify *symmetries* of the set. This will become clearer as we proceed.

2.2. **Linear isomorphisms and automorphisms.** A *linear isomorphism* is defined as a bijective linear transformation. We can think of a linear isomorphism as a map that preserves precisely all the "linear structure" of the set.

Note that for a linear transformation to be bijective, the dimensions of the start and end space are the same. Explicitly, if $T : \mathbb{R}^m \to \mathbb{R}^n$ is a bijective linear transformation, then $m = n$. Also, the matrix for T has full rank. It is also a square matrix, since $m = n$.

In other words, the dimension of a vector space is invariant under linear isomorphisms. In fact, the relationship between linear isomorphisms and dimension is similar to the relationship between set isomorphisms and cardinality (set size).

A *linear automorphism* is defined as a linear isomorphism from a vector space $\mathbb{R}^n$ to itself. Based on the above discussion, you might believe that every linear isomorphism must be a linear automorphism. This is not quite true. The main caveat at the moment (there will be more later) is that $\mathbb{R}^n$ and $\mathbb{R}^n$ could refer to different spaces depending on what kinds of things we are storing using the real numbers (for instance, one $\mathbb{R}^n$ might be masses, the other $\mathbb{R}^n$ might be prices, and so on). A linear isomorphism between a $\mathbb{R}^n$ of one sort and a $\mathbb{R}^n$ of another sort should not rightly be considered a linear automorphism.

2.3. **Affine isomorphisms.** An *affine linear transformation* is a function that preserves collinearity and ratios within lines, $T : \mathbb{R}^m \to \mathbb{R}^n$ is affine if the image of any line (not necessarily through the origin) in $\mathbb{R}^m$ is a line (not necessarily through the origin) in $\mathbb{R}^n$, and moreover, it preserves the ratios of lengths within each line. So in particular, if $\vec{x}, \vec{y} \in \mathbb{R}^m$, then $T(a\vec{x} + (1-a)\vec{y}) = aT(\vec{x}) + (1-a)T(\vec{y})$. In particular, it preserves midpoints.

Note that any linear transformation is affine linear. The main thing we get by allowing "affine" is that the origin need not go to the origin. An important class of affine linear transformations that are not linear is the class of *translations*. Explicitly, for a nonzero vector $\vec{v} \in \mathbb{R}^n$, the function $T : \mathbb{R}^n \to \mathbb{R}^n$ given by $\vec{x} \mapsto \vec{x} + \vec{v}$ is affine linear but not linear. In fact, all affine linear transformations can be obtained by composing translations with linear transformations.

An *affine linear isomorphism* (or *affine isomorphism* for short) is a bijective affine linear transformation.

An *affine linear automorphism* is an affine linear isomorphism from a vector space to itself.

Every linear automorphism is an affine linear automorphism. Nonzero translations give examples of affine linear automorphisms that are not linear. In an important sense, these cover all the affine linear automorphisms: every affine linear automorphism can be expressed as a composite of a translation and a linear automorphism.

2.4. **Linear and affine linear: the special case of one dimension.** In the case of one dimension, a linear isomorphism is a function of the form $x \mapsto mx, m \neq 0$. The matrix of this, viewed as a linear transformation, is $[m]$. In other words, it is a linear function with zero intercept.

An affine linear isomorphism is a function of the form $x \mapsto mx + c$, $m \neq 0$. In other words, it is a linear function with the intercept allowed to be nonzero.

2.5. **The general description of affine linear transformations.** A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ can be written in the form:

$$\vec{x} \mapsto A\vec{x} + \vec{b}$$

where A is a $n \times m$ matrix and $\vec{v}$ is a vector in $\mathbb{R}^n$. Explicitly, the matrix A describes the linear transformation part and the vector $\vec{b}$ describes the translation part. Notice how this general description parallels and generalizes the description in one dimension.

2.6. **Understanding how transformations and automorphisms behave intuitively.** In order to understand transformations and automorphisms using our geometric intuitions, it helps to start off with some geometric picture in the plane or Euclidean space that we are transforming, then apply the transformation to it, and see what we get. It is preferable to not take something *too* symmetric, because the less the symmetry, the more easily we can discern what features the transformation preserves and what features it destroys. Human stick features with asymmetric faces may be a good starting point for intuitive understanding, though more mundane figures like triangles might also be reasonable.

3. Euclidean geometry

3.1. **Euclidean distance and self-isometries.** The geometry of $\mathbb{R}^n$ is largely determined by how we define distance between points. The standard definition of distance is *Euclidean distance.* Explicitly, if $\vec{x}$ and $\vec{y}$ are in $\mathbb{R}^n$, then the Euclidean distance between $\vec{x}$ and $\vec{y}$ is:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

If a transformation preserves Euclidean distance, then it preserves all the geometry that we care about. In particular, it preserves shapes, sizes, angles, and other geometric features.

A bijective function from $\mathbb{R}^n$ to itself that preserves Euclidean distance is termed a *self-isometry* of $\mathbb{R}^n$.

We now proceed to an explanation of why self-isometries of $\mathbb{R}^n$ must necessarily be affine linear automorphisms of $\mathbb{R}^n$. The claim is that if something preserves distance, it must preserve linearity.

One way of characterizing the fact that three points $A, B, C \in \mathbb{R}^n$ are collinear, with B *between* A and C, is that we get the equality case for the triangle inequality. Explicitly:

$$(\text{the distance between } A \text{ and } B) + (\text{the distance between } B \text{ and } C) = (\text{the distance between } A \text{ and } C)$$

Suppose T is a self-isometry of $\mathbb{R}^n$. Then, we have:

$$\begin{array}{rcl} \text{The distance between } T(A) \text{ and } T(B) & = & \text{The distance between } A \text{ and } B \\ \text{The distance between } T(B) \text{ and } T(C) & = & \text{The distance between } B \text{ and } C \\ \text{The distance between } T(A) \text{ and } T(C) & = & \text{The distance between } A \text{ and } C \end{array}$$

Combining all these, we get that:

$$(\text{the distance between } T(A) \text{ and } T(B)) + (\text{the distance between } T(B) \text{ and } T(C)) = (\text{the distance between } T(A) \text{ and } T(C))$$

The conclusion is that $T(A)$, $T(B)$, and $T(C)$ are collinear with $T(B)$ between $T(A)$ and $T(C)$. In other words, collinear triples of points get mapped to collinear triples of points, so T preserves collinearity. Further, it obviously preserves ratios of lengths within lines. Thus, T is an affine linear automorphism of $\mathbb{R}^n$. The upshot: every self-isometry is an affine linear automorphism.

Self-isometries preserve not just collinearity, but *all* the geometric structure. What they are allowed to change is the location, angling, and orientation. They do not affect the shape and size of figures. In particular, they send triangles to congruent triangles.

3.2. **Self-homotheties.** A *self-homothety* or *similarity transformation* or *similitude transformation* is a bijective map $T : \mathbb{R}^n \to \mathbb{R}^n$ that scales all distances by a fixed nonzero factor called the *ratio of similitude* or *factor of similitude* of T. For instance, a self-homothety by a factor of $1/2$ will have the property that the distance between $T(A)$ and $T(B)$ is half the distance between A and B.

Self-isometries can be described as self-homotheties by a factor of 1.

Self-homotheties are affine linear automorphisms for roughly the same reason that self-isometries are.

A special kind of self-homothety is a *dilation* about a point. A dilation about the origin, for instance, would simply mean multiplying all the position vectors of points by a fixed nonzero scalar. The absolute value of that scalar will turn out to be the factor of similitude. The matrix for such a dilation is a scalar matrix. For instance, the matrix:

$$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

represents a dilation by a factor of 5 about the origin.

Self-homotheties send triangles to similar triangles.

3.3. **Other types of transformations.** One important type of linear transformation that is not a self-homothety is a transformation with diagonal matrix where the diagonal entries are not all the same. For instance, consider the linear transformation with matrix the following diagonal matrix:

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

This sends $\vec{e_1}$ to itself and sends $\vec{e_2}$ to $2\vec{e_2}$. Pictorially, it keeps the x-axis as is and stretches the y-axis by a factor of 2. It distorts shapes, but preserves linearity. It is not a self-homothety because of the different scaling factors used for the axes.

3.4. **Group structure.** A collection of bijective functions from $\mathbb{R}^n$ to $\mathbb{R}^n$ is said to form a *group* if it satisfies these three conditions:

- The composite of two functions in the collection is in the collection.
- The identity function is in the collection.
- The inverse to any function in the collection is in the collection.

The set of all automorphisms of any structure forms a group. Here is the stylized argument:

- Automorphisms preserve some particular structural feature. Composing two automorphisms will therefore also preserve the structural feature.
- The identity map preserves *everything.* Therefore, it must be an automorphism.
- Since an automorphism preserves a specific structural feature, doing it backwards must also preserve the structural feature.

All the examples we have seen above give groups of linear transformations. Explicitly:

- The set of all affine linear automorphisms of $\mathbb{R}^n$ is a group, because these are precisely the invertible functions that preserve the collinearity and ratios-within-lines structure.
- The set of all linear automorphisms of $\mathbb{R}^n$ is a group, because these are precisely the invertible functions that preserve the linear structure.
- The set of all self-isometries of $\mathbb{R}^n$ is a group, because these are precisely the invertible functions that preserve the Euclidean distance.
- The set of all self-homotheties of $\mathbb{R}^n$ of a group, because these are precisely the invertible functions that preserve the "ratios of Euclidean distances" structure.

Further, these groups have containment relations:

Group of all self-isometries of $\mathbb{R}^n \subseteq$ Group of all self-homotheties of $\mathbb{R}^n \subseteq$ Group of all affine linear automorphisms of $\mathbb{R}^n$

And also, separately:

Group of all linear automorphisms of $\mathbb{R}^n \subseteq$ Group of all affine linear automorphisms of $\mathbb{R}^n$

4. The case of two dimensions

If $n = 2$, we can obtain a relatively thorough understanding of the various types of linear transformations. These are discussed in more detail below.

4.1. **Trace and determinant.**

There are two interesting invariants of 2×2 matrices, called respectively the *trace* and *determinant.* The trace of a 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is defined as the quantity $a + d$. It is the sum of the diagonal entries. The significance of the trace is not clear right now, but will become so later.

The other important invariant for 2×2 matrices is the *determinant.* The determinant of a 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is defined as the quantity $ad - bc$. As we noted in a homework exercise, the determinant is nonzero if and only if the matrix is invertible.

Both the trace and the determinant generalize to $n \times n$ matrices. The trace of a $n \times n$ matrix is defined as the sum of all the diagonal entries of the matrix. The determinant is defined in a more complicated fashion.

The trace and determinant of a linear transformation are defined respectively as the trace and determinant of the matrix for the linear transformation.

For an affine linear transformation, the trace and determinant are defines respectively as the trace and determinant of the linear part of the transformation.

The role of the determinant is somewhat hard to describe, but it can be split into two aspects:

- The absolute value of the determinant is the factor by which volumes multiply. In the case $n = 2$, it is the factor by which areas multiply. In particular, the determinant of a 2×2 matrix is ± 1 if and only if the corresponding linear transformation is area-preserving.
- The sign of the determinant describes whether the linear transformation is orientation-preserving or orientation-reversing. A positive sign means orientation-preserving, whereas a negative sign means orientation-reversing. Here, *orientation-preserving* means that left-handed remains left-handed while right-handed remains right-handed. In contrast, *orientation-reversing* means that left-handed becomes right-handed while right-handed becomes left-handed. Note that any transformation that can be accomplished through rigid motions, i.e., through a continuous deformation of the identity transformation, must be orientation-preserving. The reason is that continuous change cannot suddenly change the orientation status.

4.2. **Justifying statements about the determinant using diagonal matrices.**

Consider a linear transformation with diagonal matrix:

$$\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$$

The trace of this matrix is $a + d$ and the determinant is ad. Here is the justification for both observations made above:

- The absolute value of the determinant is the factor by which volumes multiply: Think of a rectangle with sides parallel to the axes. The x-dimension gets multiplied by a factor of $|a|$ and the y-dimension gets multiplied by a factor of $|d|$. The area therefore gets multiplied by a factor of $|a||d|$ which is $|ad|$, the absolute value of the determinant.
- The sign of the determinant describes whether the linear transformation is orientation-preserving or orientation-reversing. The sign of a determines whether the x-direction gets flipped. The sign of d determines whether the y-direction gets flipped. The sign of the product determines whether the overall orientation stays the same or gets reversed.

4.3. **Abstract considerations: determinants of self-homotheties.** Consider a linear transformation that is a self-homothety with factor of similitude λ. This linear transformation scales all lengths by a factor of λ. If $n = 2$ (i.e., we are in two dimensions) then it scales all areas by a factor of λ^2. In particular:

- If it is orientation-preserving, then the determinant is λ^2.
- If it is orientation-reversing, then the determinant is $-\lambda^2$.

In particular, for a self-*isometry*:

- If it is orientation-preserving, then the determinant is 1.
- If it is orientation-reversing, then the determinant is -1.

4.4. **Rotations.** A particular kind of transformation of interest in the two-dimensional case is a *rotation.* A rotation is specified by two pieces of information: a point (called the *center of rotation*) and an angle (called the *angle of rotation*). The angle is defined only up to additive multiples of 2π, i.e., if two rotations have the same center and their angles differ by a multiple of 2π, then they are actually the same rotation.

Note that we set a convention in advance that we will interpret rotations in the counter-clockwise sense.

For a rotation whose angle of rotation is not zero (or more precisely, is not a multiple of 2π), the center of rotation is uniquely determined by the rotation and is the only fixed point of the rotation.

The rotation by an angle of π about a point is termed a *half turn* about the point and can alternatively by thought of as *reflecting* relative to the *point.* This is not to be confused with reflections about lines in $\mathbb{R}^2$.

All rotations are self-isometries of $\mathbb{R}^2$. Thus, they are affine linear automorphisms. They are also area-preserving. They are also orientation-preserving, since they can be obtained through continuous rigid motions. However, unless the center of rotation is the origin, the rotation is not a linear automorphism. Rotations centered at the origin *are* linear transformations. We will now proceed to describe rotations centered at the origin in matrix terms.

To describe a rotation centered at the origin, we need to describe the images of the two standard basis vectors $\vec{e_1}$ and $\vec{e_2}$. These images form the first and second column respectively of the matrix describing the rotation as a linear transformation.

Suppose the rotation is by an angle θ. Then $\vec{e_1}$ goes to the vector with coordinates $(\cos\theta, \sin\theta)$. $\vec{e_2}$ goes to the vector with coordinates $(-\sin\theta, \cos\theta)$. The matrix for the rotation is thus:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Note that the inverse to rotation about the origin by θ is rotation by $-\theta$. Using the fact that cos is an even function and sin is an odd function ,the matrix for the inverse operation is:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Some specific rotations of interest are listed below:

Angle	Matrix of rotation
0 (no change)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$\pi/4$	$\begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$
$\pi/2$ (right angle)	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
π (half turn)	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
$3\pi/2$ (right angle clockwise)	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

Let us try predicting the determinant of a rotation matrix theoretically, then proceed to verify it computationally.

Theoretically, we know that rotations are both area-preserving (on account of being self-isometries) and orientation-preserving (on account of being realized through rigid motions). The area-preserving nature tells us that the magnitude of the determinant is 1, i.e., the determinant is ± 1. The orientation-preserving nature tells us that the determinant is positive. Combining these, we get that the determinant must be 1. Let's check this. The determinant of

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

is $\cos^2\theta + \sin^2\theta$, which is 1.

4.5. **Reflections and orientation-reversal.** A reflection about a line does exactly what it is supposed to do: it sends each point to another point such that the line of reflection is the perpendicular bisector of the line segment joining them.

Every reflection is an affine linear automorphism. A reflection is a linear automorphism if and only if the line of reflection passes through the origin. If that is the case, we can write the matrix of the linear transformation. Let's consider the case of a reflection about the x-axis.

This reflection fixes all points on the x-axis, and sends all points on the y-axis to their mirror images about the origin. In particular, it sends $\vec{e_1}$ to $\vec{e_1}$ and sends $\vec{e_2}$ to $-\vec{e_2}$.

The matrix of the linear transformation is:

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

More generally, consider a reflection about a line through the origin that makes an angle of $\theta/2$ counter-clockwise from the x-axis. The reflection sends $\vec{e_1}$ to a vector making an angle θ counter-clockwise from the horizontal, and sends $\vec{e_2}$ to the vector making an angle of $\theta - (\pi/2)$ counter-clockwise from the horizontal. The matrix is thus:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

This reflection matrix has trace zero. For the determinant, let us first predict it theoretically, then verify it computationally. Theoretically, we know that reflections are self-isometries, hence they are area-preserving. So, the absolute value of the determinant is 1, and the determinant is ± 1. Reflections are also orientation-reversing, so the determinant is negative. Thus, the determinant must be -1. Let's check this. The determinant of:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

is $-\cos^2\theta - \sin^2\theta = -1$.

4.6. **Shear operations.** A shear operation is an operation where one axis is kept fixed, and the other axis is "sheared" by having stuff from the fixed axis added to it.

For instance, suppose the x-axis is the fixed axis and we add the standard basis vector for the x-axis to the y-axis. Explicitly, $\vec{e_1}$ stays where it is, but $\vec{e_2}$ gets sent to $\vec{e_1} + \vec{e_2}$. The matrix of this looks like:

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

Note that unlike translations, rotations, and reflections, shear operations are not self-isometries.

More generally, we can have a shear of the form:

$$\begin{bmatrix} 1 & \lambda \\ 0 & 1 \end{bmatrix}$$

This sends $\vec{e_1}$ to $\vec{e_1}$ and sends $\vec{e_2}$ to $\lambda\vec{e_1} + \vec{e_2}$.

We could also shear in the other direction:

$$\begin{bmatrix} 1 & 0 \\ \lambda & 1 \end{bmatrix}$$

Here, $\vec{e_2}$ is fixed, and $\vec{e_1}$ gets sends to $\vec{e_1} + \lambda\vec{e_2}$.

The trace of a shear operation is 2, and the determinant is 1. Thus, shear operations are both area-preserving and orientation-preserving. This can be verified pictorially.

4.7. **Composites of various types, including glide reflections.** We have considered translations, rotations, and reflections. All of these are self-isometries. Self-isometries form a group, so composing things of these types should also give a self-isometry. Two interesting questions:

- Is every self-isometry a translation, rotation, or reflection?
- Can every self-isometry be obtained by composing translations, rotations, and reflections?

It turns out that the answers are respectively *no* and *yes*. Let's look at various sorts of composites:

(1) *Composite of translations*: Translations form a subgroup of the group of all self-isometries of $\mathbb{R}^2$. In other words, the composite of two translations is a translation, the identity map is a translation (namely, by the zero vector) and the inverse of a translation is a translation.
(2) *Composite of rotations centered at the same point*: The rotations centered at a particular point form a subgroup of the group of all self-isometries of $\mathbb{R}^2$. In other words, the composite of two rotations centered at the same point is a rotation, the identity map is a rotation with any point as center (namely, with zero angle of rotation), and the inverse of a rotation is a rotation with the same center of rotation. However, the set of *all* rotations is not a subgroup, as is clear from the next point.
(3) *Composite of rotations centered at different points*: If two rotations centered at different points are composed, the composite is typically a rotation about yet a third point, with the angle of rotation the sum of the angles. The exception is when the angles add up to a multiple of 2π. In that case, the composite is a translation. It is easy to convince yourself by using human stick figures that the angles of rotation add up.
(4) *Composite of rotation and translation*: The composite is again a rotation with the same angle of rotation, but about a different center of rotation.
(5) *Composite of two reflections*: If the lines of reflection are parallel, the composite is a translation by a vector perpendicular to both. If the lines of reflection intersect, then the composite is a rotation by twice the angle of intersection between the lines.
(6) *Composite of reflection and translation*: This gives rise to what is called a **glide reflection**, which is a new type of self-isometry of $\mathbb{R}^2$.
(7) *Composite of reflection and rotation*: This is trickier. It could be a reflection or a glide reflection, depending on whether the center of rotation lies on the line of reflection.

The upshot is that:

- The orientation-*preserving* self-isometries of $\mathbb{R}^2$ are precisely the translations and rotations. Note that these form a group.
- The orientation-*reversing* self-isometries of $\mathbb{R}^2$ are precisely the reflections and glide reflections. Note that these do not form a group, but *together* with translations and rotations, they form the group of all self-isometries.

4.8. **Self-isometries that are linear.** Let's consider self-isometries that are linear, i.e., they fix the origin. These are subgroups of the group of all self-isometries. Explicitly:

- The orientation-*preserving* linear self-isometries of $\mathbb{R}^2$ are precisely the rotations about the origin, specified by the angle of rotation (determined up to additive multiples of 2π). Composing two such rotations involves adding the corresponding angles. These form a group. This group is denoted $SO(2, \mathbb{R})$ (you don't need to know this!) and is called the *special orthogonal group* of degree two over the reals.
- The orientation-*reversing* linear self-isometries of $\mathbb{R}^2$ are precisely the reflections about lines through the origin. These do not form a group, but *together* with rotations about the origin, they form a group. The whole group is denoted $O(2, \mathbb{R})$ (you don't need to know this!) and is called the *orthogonal group* of degree two over the reals.

In general, an affine linear automorphism is a self-isometry if and only if its linear automorphism part is a self-isometry. In other words:

$$\vec{x} \mapsto A\vec{x} + \vec{b}$$

is a self-isometry if and only if $\vec{x} \mapsto A\vec{x}$ is a self-isometry.

5. The case of three dimensions

The case of three dimensions is somewhat trickier than two dimensions, but we can still come somewhat close to a classification.

5.1. Rotations about axes.

The simplest type of orientation-preserving self-isometry is a rotation about an axis of rotation. Euler proved a theorem (called *Euler's rotation theorem*) that every orientation-preserving self-isometry that fixes a point must be a rotation about an axis through that point. In particular, all the orientation-preserving self-isometries of $\mathbb{R}^3$ that are *linear* (in the sense of fixing the origin) are rotations about axes through the origin.

5.2. Rotations composed with translations.

We know that translations are orientation-preserving self-isometries of $\mathbb{R}^n$ for any n. So are rotations. We also know that the self-isometries form a group. Thus, composing a rotation about an axis with a translation should yield a self-isometry. For $n = 2$, any such self-isometry would already be a rotation. For $n = 3$, this is no longer the case. It is possible to have orientation-preserving self-isometries that are expressible as composites of translations and rotations but are not translations or rotations themselves. For instance, a rotation about the z-axis followed by a translation parallel to the z-axis works.

5.3. Reflections about planes and their composites.

A reflection about a plane in $\mathbb{R}^3$ is an orientation-reversing self-isometry of $\mathbb{R}^3$. For instance, the isometry:

$$(x, y, z) \mapsto (-x, y, z)$$

is a reflection about the yz-plane. It is orientation-reversing, and its matrix is:

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

We have not yet seen how to compute the determinant of a general 3×3 matrix. However, the determinant of a diagonal matrix is simply the product of the diagonal entries. In this case, the determinant is -1, as it should be, since the reflection is orientation-reversing but, on account of being a self-isometry, is *volume*-preserving.

A composite of two reflections about different planes is an orientation-preserving self-isometry. If the planes are not parallel, this is a rotation about the axis of intersection by twice the angle of intersection between the planes. For instance, the transformation:

$$(x, y, z) \mapsto (-x, -y, z)$$

corresponds to the diagonal matrix:

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

This has determinant 1. It is both orientation-preserving and area-preserving. We can also think of it as a rotation by an angle of π about the z-axis, i.e., a half-turn about the z-axis. The angle is π because the individual planes of reflection are mutually perpendicular (angle $\pi/2$).

Finally, consider a composite of *three* reflections. Consider the simple case where the reflections are about three mutually perpendicular planes. An example is:

$$(x, y, z) \mapsto (-x, -y, -z)$$

This corresponds to the diagonal matrix:

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

The linear transformation here is orientation-reversing on account of being a composite of an odd number of reflections. It is a self-isometry, so the determinant should be -1, and indeed, the determinant is -1.

More generally, we could have a map of the form:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

This map reflects the x-coordinate and performs a rotation by θ on the yz-plane.

6. Where we're hoping to go with this

In the future, we will build on what we have learned so far in the following ways:

- We will understand the procedure for composing linear transformations (or more generally affine linear transformations) purely algebraically, i.e., in terms of their description using matrices and vectors.
- We will understand criteria for looking at a linear transformation algebraically to determine whether it is a self-isometry, self-homothety, orientation-preserving, and/or area-preserving.
- We will understand more about the group structure of various kinds of groups of linear transformations and affine linear transformations.

IMAGE AND KERNEL OF A LINEAR TRANSFORMATION

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 3.1.

Executive summary

(1) For a function $f : A \to B$, we call A the domain, B the co-domain, $f(A)$ the range, and $f^{-1}(b)$, for any $b \in B$, the fiber (or inverse image or pre-image) of b. For a subset S of B, $f^{-1}(B) = \bigcup_{b \in S} f^{-1}(b)$.
(2) The sizes of fibers can be used to characterize injectivity (each fiber has size at most one), surjectivity (each fiber is non-empty), and bijectivity (each fiber has size exactly one).
(3) Composition rules: composite of injective is injective, composite of surjective is surjective, composite of bijective is bijective.
(4) If $g \circ f$ is injective, then f must be injective.
(5) If $g \circ f$ is surjective, then g must be surjective.
(6) If $g \circ f$ is bijective, then f must be injective and g must be surjective.
(7) Finding the fibers for a function of one variable can be interpreted geometrically (intersect graph with a horizontal line) or algebraically (solve an equation).
(8) For continuous functions of one variable defined on all of $\mathbb{R}$, being injective is equivalent to being increasing throughout or decreasing throughout. More in the lecture notes, sections 2.2-2.5.
(9) A vector $\vec{v}$ is termed a *linear combination* of the vectors $\vec{v_1}, \vec{v_2}, \dots, \vec{v_r}$ if there exist real numbers $a_1, a_2, \dots, a_r \in \mathbb{R}$ such that $\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + \dots + a_r\vec{v_r}$. We use the term *nontrivial* if the coefficients are not all zero.
(10) A subspace of $\mathbb{R}^n$ is a subset that contains the zero vector and is closed under addition and scalar multiplication.
(11) The span of a set of vectors is defined as the set of all vectors that can be written as linear combinations of vectors in that set. The span of any set of vectors is a subspace.
(12) A spanning set for a subspace is defined as a subset of the subspace whose span is the subspace.
(13) Adding more vectors either preserves or increases the span. If the new vectors are in the span of the previous vectors, it preserves the span, otherwise, it increases it.
(14) The kernel and image (i.e., range) of a linear transformation are respectively subspaces of the domain and co-domain. The kernel is defined as the inverse image of the zero vector.
(15) The column vectors of the matrix of a linear transformation form a spanning set for the image of that linear transformation.
(16) To find a spanning set for the kernel, we convert to rref, then find the solutions parametrically (with zero as the augmenting column) then determine the vectors whose linear combinations are being discussed. The parameters serve as the coefficients for the linear combination. There is a shortening of this method. (See the lecture notes, Section 4.4, for a simple example done the long way and the short way).
(17) The fibers for a linear transformation are translates of the kernel. Explicitly, the inverse image of a vector is either empty or is of the form (particular vector) + (arbitrary element of the kernel).
(18) The dimension of a subspace of $\mathbb{R}^n$ is defined as the minimum possible size of a spanning set for that subspace.
(19) For a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ with $n \times m$ matrix having rank r, the dimension of the kernel is $m - r$ and the dimension of the image is r. Full row rank $r = n$ means surjective (image is all of $\mathbb{R}^n$) and full column rank $r = m$ means injective (kernel is zero subspace).
(20) We can define the intersection and sum of subspaces of $\mathbb{R}^n$.

(21) The kernel of $T_1 + T_2$ contains the intersection of the kernels of T_1 and T_2. More is true (see the lecture notes).
(22) The image of $T_1 + T_2$ is contained in the sum of the images of T_1 and T_2. More is true (see the lecture notes).
(23) The dimension of the inverse image $T^{-1}(X)$ of any subspace X of $\mathbb{R}^n$ under a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ satisfies:

$$\dim(\text{Ker}(T)) \leq \dim(T^{-1}(X)) \leq \dim(\text{Ker}(T)) + \dim(X)$$

The upper bound holds if X lies inside the image of T.
(24) Please read through the lecture notes thoroughly, since the summary here is very brief and inadequate.

1. Image (range) and inverse images (fibers) for a function

1.1. **Domain, range (image), and co-domain.** Suppose $f : A \to B$ is a function (of any sort). We call A the *domain* of f and we call B the *co-domain* of f. Note that it is not necessary that every element of B occur as the image of an element of A under f. The subset of B that comprises the elements of the form $f(x), x \in A$ is termed the *range* of f or the *image* of f. Note that if the range is all of B, we say that f is *surjective*.

1.2. **Fibers of a function.** As before, $f : A \to B$ is a function. For any $b \in B$, define $f^{-1}(b)$ as the set $\{a \in A \mid f(a) = b\}$. We call $f^{-1}(b)$ the *fiber* of f corresponding to b. Other words used are *pre-image* and *inverse image*. Note the following:

- $f^{-1}(b)$ is non-empty if and only if b is in the range of f. In particular, $f^{-1}(b)$ is non-empty for *every* $b \in B$ if and only if f is surjective.
- $f^{-1}(b)$ has size *at most one* for every $b \in B$ if and only if f is injective.
- $f^{-1}(b)$ has size *exactly one* for every $b \in B$ if and only if f is bijective, i.e., invertible.

We sometimes say that a set map is *uniform* (not a standard term) if all its non-empty fibers have the same size. Note that injective set maps are uniform. However, there may exist uniform non-injective set maps. For instance, a set map where all the fibers have size two is uniform. The idea of uniformity based on cardinality works well for maps of finite sets. For infinite sets, we may use structurally dependent concepts of uniformity since cardinality (set size) is too crude a size measure in the infinite context. We will revisit this idea in the context of linear transformations.

1.3. **Composition, injectivity, surjectivity, and one-sided inverses.** We've discussed in the past that a function has a two-sided inverse (i.e., is invertible) if and only if it is bijective. We now discuss some material that will help us understand one-sided inverses.

First, a slight extension of the terminology introduced previously. Consider a function $f : A \to B$ For a subset S of B, define $f^{-1}(S) = \{a \in A \mid f(a) \in S\}$. It is clear that $f^{-1}(S) = \bigcup_{b \in S} f^{-1}(b)$. Pictorially, it is the union of the fibers over all the points in S.

Now, consider a situation where we are composing two functions:

$$f : A \to B, g : B \to C, \text{composite } g \circ f : A \to C$$

Given $c \in C$, let's try to figure out what $(g \circ f)^{-1}(c)$ would look like. We must find all $a \in A$ such that $g(f(a)) = c$. In order to do that, we first find the $b \in B$ such that $g(b) = c$. In other words, we first find $g^{-1}(c)$. Then, we find, for each $b \in g^{-1}(c)$, the $a \in A$ such that $f(a) = b$. The upshot is that we are computing $f^{-1}(g^{-1}(c))$, where the inner computation yields a subset to which we apply the outer inverse function.

Pictorially, we first take the fiber over c for g. This is a subset of B. For each element here, we take the fiber in A. Then, we take the union of the fibers to get $(g \circ f)^{-1}(a)$.

We note the following facts regarding composition, injectivity and surjectivity:

(1) Suppose $f : A \to B$ and $g : B \to C$ are both injective functions. Then, the composite function $g \circ f : A \to C$ is also an injective function. Here's the proof:
Want to show: If $a_1, a_2 \in A$ satisfy $g(f(a_1)) = g(f(a_2))$, then $a_1 = a_2$.

Proof: We "peel off the layers" one by one. Explicitly, we begin with:

$$g(f(a_1)) = g(f(a_2))$$

We use that g is injective as a function from B to C to conclude that we can cancel the g, so we obtain that:

$$f(a_1) = f(a_2)$$

We now use the injectivity of f as a function from A to B and obtain that:

$$a_1 = a_2$$

as desired.

(2) Suppose $f : A \to B$ and $g : B \to C$ are both surjective functions. Then, the composite function $g \circ f : A \to C$ is also a surjective function. The proof follows:

Want to show: For any $c \in C$, there exists $a \in A$ such that $g(f(a)) = c$.

Proof:
- Since g is surjective, the set $g^{-1}(c)$ is non-empty, i.e., there exists $b \in B$ such that $g(b) = c$.
- Since f is surjective, the set $f^{-1}(b)$ is non-empty, i.e., there exists $a \in A$ such that $f(a) = b$.
- The element a obtained this way is the one we desire: $g(f(a)) = g(b) = c$.

In both the injectivity and surjectivity cases, we use the information about the individual functions one step at a time.

(3) Suppose $f : A \to B$ and $g : B \to C$ are both bijective functions. Then, the composite function $g \circ f : A \to C$ is also a bijective function.

This follows by combining the corresponding statements for injectivity and surjectivity, i.e., points (1) and (2) above.

(4) Suppose $f : A \to B$ and $g : B \to C$ are functions such that the composite $g \circ f : A \to C$ is injective. Then, f must be injective. This is easiest to prove by contradiction: we can show that if f is not injective, then $g \circ f$ is not injective. Here's how: if f has a collision, i.e., two different elements $a_1, a_2 \in A$ satisfying $f(a_1) = f(a_2)$, then $g \circ f$ also has a collision for the same pair of inputs, i.e., we also have $g(f(a_1)) = g(f(a_2))$.

In fact, the proof shows something stronger: any collision for f remains a collision for $g \circ f$.

(5) Suppose $f : A \to B$ and $g : B \to C$ are functions such that the composite $g \circ f : A \to C$ is surjective. Then, g is surjective.

To see this, note that if everything in C is hit by the composite, it must be hit by g. Explicitly, if every $c \in C$ is of the form $g(f(a))$ for $a \in A$, then in particular it is $g(b)$ where $b = f(a)$.

In fact, the proof shows something slightly stronger: the range of $g \circ f$ is contained within the range of g.

(6) Suppose $f : A \to B$ and $g : B \to C$ are functions such that the composite function $g \circ f : A \to C$ is bijective. Then, g is surjective and f is injective. This follows from the preceding two observations ((4) and (5) in the list).

(7) Suppose $f : A \to B$ and $g : B \to A$ are functions such that the composite function $g \circ f : A \to A$ is the identity map on A. In other words, g is a left inverse to f, and f is a right inverse to g. Then, f is injective and g is surjective. This is a direct consequence of (6), along with the observation that the identity map from any set to itself is bijective.

There was a lot of confusion about points (4) and (5) of the list in class, suggesting that it is pretty hard to "get" these by yourself. So please review the reasoning carefully and convince yourself of it.

Point (7) raises two interesting converse questions:

- Suppose $f : A \to B$ is injective. Does there exist $g : B \to A$ such that $g \circ f$ is the identity map on A? The answer turns out to be *yes* for set maps. The idea is to construct g as essentially the inverse function to f on the range of f, and define it whatever way we please on the rest of B. Note that if f is injective but not surjective (so that the range of f is not all of B) then this one-sided inverse to f is non-unique, because we have flexibility regarding where we send all the elements that are in B but outside $f(A)$.

- Suppose $g : B \to A$ is surjective. Does there exist $f : A \to B$ such that $g \circ f$ is the identity map on A? The answer turns out to be *yes* for set maps. The idea is to construct f as follows: for every element $a \in A$, let $f(a)$ be any element b satisfying $g(b) = a$. We always *can* pick such an element, because g is surjective.

2. Some warm-up examples involving functions of one variable

2.1. Real-valued functions of one variable. Before we proceed to apply the ideas above to linear transformations, let us consider what the ideas tell us in the context of continuous functions from $\mathbb{R}$ to $\mathbb{R}$. Since you are quite familiar with single-variable precalculus and calculus, this will be helpful.

For a function f with domain a subset of $\mathbb{R}$ and co-domain $\mathbb{R}$ (so f is a real-valued function of one variable), we can find the fiber $f^{-1}(y_0)$ for a particular $y_0 \in \mathbb{R}$ in either of these ways:

- *Graphically*: We draw the line $y = y_0$, find all its points of intersection with the graph of f, and then look at the x-coordinates of those points of intersection. The set of all the x-coordinates is the inverse image (also called the pre-image or fiber) of y_0.
- *Algebraically*: We set up the equation $f(x) = y_0$ and try to solve for x.

The following turn out to be true:

- A function is injective if and only if every horizontal line intersects its graph at at most one point.
- The range of a function is the set of y-values for which the corresponding horizontal line intersects the graph.
- A function is surjective to $\mathbb{R}$ if and only if every horizontal line intersects its graph at at least one point.
- A function is bijective to $\mathbb{R}$ if and only if every horizontal line intersects its graph at *exactly* one point.

2.2. The continuous case. Suppose $f : \mathbb{R} \to \mathbb{R}$ is a continuous function. Note that some of the remarks we make have analogues for functions defined on intervals in $\mathbb{R}$ instead of on all of $\mathbb{R}$, but we will for simplicity consider only the case of f defined on all of $\mathbb{R}$. The following are true:

(1) f is injective if and only if it is increasing throughout $\mathbb{R}$ or decreasing throughout $\mathbb{R}$.
(2) f is surjective if and only if one of these cases is satisfied (*Note: this point was edited to include the oscillatory cases*):
  (a) $\lim_{x\to-\infty} f(x) = -\infty$ and $\lim_{x\to\infty} f(x) = \infty$. For instance, f could be any polynomial with odd degree and positive leading coefficient.
  (b) $\lim_{x\to-\infty} f(x) = \infty$ and $\lim_{x\to\infty} f(x) = -\infty$. For instance, f could be any polynomial with odd degree and negative leading coefficient.
  (c) As $x \to \infty$, $f(x)$ is oscillatory between $-\infty$ and ∞. For instance, $f(x) := e^x \sin x$.
  (d) As $x \to -\infty$, $f(x)$ is oscillatory between $-\infty$ and ∞. For instance, $f(x) := e^{-x} \sin x$.
  (e) As $x \to \infty$, $f(x)$ is oscillatory with the upper end of oscillation going to ∞, and as $x \to -\infty$, $f(x)$ is oscillatory with the lower end of oscillation going to $-\infty$. For instance, $f(x) := x \sin^2 x$.
  (f) As $x \to \infty$, $f(x)$ is oscillatory with the lower end of oscillation going to $-\infty$, and as $x \to -\infty$, $f(x)$ is oscillatory with the upper end of oscillation going to $+\infty$. For instance, $f(x) := -x \sin^2 x$.

  Note that the cases (c)-(f) are not mutually exclusive. For instance, $f(x) := x \sin x$ satisfies the conditions for cases (c), (d), (e), and (f). Note also that all the cases (c)-(f) are incompatible with injectivity, and also that these cases do not occur for polynomial functions.
(3) f is bijective if and only if one of these cases is satisfied:
  (a) f is increasing, $\lim_{x\to-\infty} f(x) = -\infty$, and $\lim_{x\to\infty} f(x) = \infty$
  (b) f is decreasing, $\lim_{x\to-\infty} f(x) = \infty$, and $\lim_{x\to\infty} f(x) = -\infty$

2.3. Addition, injectivity, and surjectivity. We can see from these facts that the following is true:

- The sum of two continuous injective (respectively, surjective or bijective) functions need not be injective (respectively, surjective or bijective). The problem arises if the two functions are of opposite subtypes, for instance, one is increasing and the other is decreasing. If both functions are of a similar type, then the sum is also of that type. Note that for surjective functions, the types (c)-(f) are in general problematic.

- If we take three continuous injective (respectively, bijective) functions, then at least one pair of them has the same subtype, so that sum is also of the same subtype, hence is continuous injective (respectively, bijective). In general, we cannot say more. Note that we cannot make any assertion for surjectivity per se, because of the weird types (c)-(f) for continuous surjective functions. However, if we restrict attention to continuous surjective functions of the subtypes (a) and (b), then adding two functions of the same subtype gives a function of the same subtype.

2.4. **Fibers for polynomial functions.** If $f : \mathbb{R} \to \mathbb{R}$ is a polynomial function of degree $n \geq 1$, finding the inverse image under f of any particular real number amounts to solving a particular polynomial equation of degree n. Explicitly, for an output y_0, finding $f^{-1}(y_0)$ is tantamount to solving the polynomial equation $f(x) - y_0 = 0$ which is a degree n equation in x. The polynomial equations for different points in the co-domain differ only in their constant term. The following are true:

- In the case $n = 1$ (i.e., f is a linear polynomial), every fiber has size exactly 1, since f is bijective. In fact, the inverse function to f is also a linear polynomial.
- In the case $n = 2$ (i.e., f is a quadratic polynomial), every fiber has size 0, 1, or 2. The maximum of the fiber sizes is always 2. If the leading coefficient of f is positive, then f has a unique absolute minimum value. The fiber size for that is 1, the fiber size for bigger values is 2, and the fiber size for smaller values is 0. If the leading coefficient of f is negative, then f has a unique absolute maximum value. The fiber size for that is 1, the fiber size for bigger values is 0, and the fiber size for smaller values is 2.
- In the case $n \geq 3$, the maximum of the fiber sizes is at most n. However, it is *not* necessary that there exist fibers of size n for a particular f (by Rolle's theorem, we know that for this to happen it must be the case that f' has $n - 1$ distinct roots, and this does not happen for all f). An extreme case where we have small fiber sizes is x^n. For n odd, the fiber sizes are all 1 (the function is bijective). For n even, the fiber sizes are all 0, 1, or 2.

2.5. **Fibers for periodic functions, specifically trigonometric functions.** For a periodic function with period h, the inverse image of any point is invariant under translation by h and therefore also by all integer multiples of h. In particular, the inverse image of any point, if non-empty, must be infinite.

For instance, consider the sin function from $\mathbb{R}$ to $\mathbb{R}$. The range is $[-1, 1]$. For any point in this set, the inverse image is invariant under translation by 2π and therefore also by all integer multiples of 2π.

To determine the inverse image for such a function, it makes sense to first determine the inverse image over an interval of length equal to the period, and then note that the inverse image over $\mathbb{R}$ is obtained from this set by translating by multiples of 2π. For instance, to find the inverse image of $1/2$ under sin, we note that on $[0, 2\pi]$, the only solutions to $\sin x = 1/2$ are $x = \pi/6$ and $x = 5\pi/6$. Thus, the fiber is the set:

$$\{2n\pi + \pi/6 \mid n \in \mathbb{Z}\} \cup \{2n\pi + 5\pi/6 \mid n \in \mathbb{Z}\}$$

3. Linear combinations and spans

3.1. **Linear combination.** There is a very important concept that undergirds linear algebra: *linear combination.*

Suppose $\vec{v_1}, \vec{v_2}, \dots, \vec{v_r}$ are vectors in $\mathbb{R}^n$. A vector $\vec{v}$ is termed a *linear combination* of the vectors $\vec{v_1}$, $\vec{v_2}$, $\dots$, $\vec{v_r}$ if there exist real numbers a_1, a_2, $\dots$, a_r such that:

$$\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + \cdots + a_r\vec{v_r}$$

The real numbers $a_1, a_2, \dots, a_r$ are allowed to be zero and repetitions in values are allowed.

The term "linear combination" indicates the idea of "combining" the vectors in a *linear* fashion. Recall that linearity has to do with addition and scalar multiplication. The values $a_1, a_2, \dots, a_r$ are termed the *coefficients* used in the linear combination.

Geometrically, linear combinations include scaling and addition. If we think of vectors as physical translation, then addition is how we compose the translation operations. Thus, the linear combinations of vectors are the vectors we can obtain by moving along the directions of the individual vectors one after the other. The coefficients $a_1, a_2, \dots, a_r$ describe how much we move along the direction of each vector.

We use the following terms:

- The *trivial* linear combination refers to the situation where all the coefficients are zero. The only vector you can write as a trivial linear combination is the zero vector. Thus, given any collection of vectors, the zero vector is the trivial linear combination of these.
- Any other linear combination is termed a *nontrivial* linear combination. However, a nontrivial linear combination may still have *some* coefficients equal to zero. We may use informal jargon such as "uses a vector nontrivially" to indicate that the linear combination has a nonzero coefficient for that particular vector.

Here are a couple of extreme cases:

- For the empty collection of vectors, we can still define the "trivial linear combination" to be the zero vector. This seems somewhat silly, but there's a deep logic.
- For a set of vectors of size one, the "linear combinations" possible are precisely the scalar multiples of that vector.

Here are a few easy observations:

- If a vector is a linear combination of a certain set of vectors, the given vector is also a linear combination of any bigger set of vectors: just set the coefficients on all the other vectors to be 0. For instance, if $\vec{v} = 2\vec{v_1} + 5\vec{v_2}$, then we also have $\vec{v} = 2\vec{v_1} + 5\vec{v_2} + 0\vec{v_3}$. We don't end up "using" the extra vectors, but it is still a legitimate linear combination.
- For a (possibly infinite) set S of vectors in $\mathbb{R}^n$, writing a vector $\vec{v}$ as a linear combination of S means writing $\vec{v}$ as a linear combination of a finite subset of S. We can think of that linear combination as attaching a coefficient of zero to all but finitely many vectors in $\mathbb{R}^n$.

A priori, there may be more than one way that a given vector can be expressed as a linear combination of a particular collection of vectors. We will later on discuss a concept called *linear independence* to describe a situation where no vector can be written as a linear combination in multiple ways.

3.2. **Definition of subspace.** We will study the concept of subspace more later, but a very brief definition is worthwhile right now. Consider a subset V of $\mathbb{R}^n$. We call V a *subspace* of $\mathbb{R}^n$ if it saitsfies the following three conditions:

- The zero vector is in V.
- V is closed under addition, i.e., for any vectors $\vec{v}, \vec{w}$ (possibly equal, possibly distinct) in V, the sum $\vec{v} + \vec{w}$ is also a vector in V. Note that the choice of leters for the vectors is not important. What matters is that regardless of the vectors chosen, the sum remains in the subspace.
- V is closed under scalar multiplication, i.e., if $\vec{v}$ is a vector in V and a is a real number, then $a\vec{v}$ is also a vector in V.

Two extreme examples of subspaces are:

- The zero subspace, i.e., the subspace comprising only the zero vector.
- The whole space, in this case $\mathbb{R}^n$.

Pictorially, a subspace is something flat, closed both under scaling and addition. Lines through the origin are subspaces. Planes through the origin are subspaces. There are other higher-dimensional concepts of subspaces. I'd love to say more, but that would be jumping the gun.

3.3. **Spanning sets and spans.** Consider a subset S of $\mathbb{R}^n$. For convenience, you can imagine S to be finite, though this is not necessary for the definition. The *span* of S is defined as the set of all vectors that arise as linear combinations of vectors in S. The span of S is a *subspace* of $\mathbb{R}^n$. Explicitly:

- The zero vector is in the span of any set, on account of being the trivial linear combination.
- *Addition*: If $\vec{v}$ and $\vec{w}$ are vectors in the span of S, then $\vec{v} + \vec{w}$ is also in the span of S: To see this, note that we can add up the coefficients for each vector. For instance, if we are looking at the span of three vectors $\vec{u_1}$, $\vec{u_2}$, $\vec{u_3}$, and we have:

$$\begin{aligned}\vec{v} &= 4\vec{u_1} - 7\vec{u_2} + 6\vec{u_3}\\ \vec{w} &= 11\vec{u_1} + 9\vec{u_2} + 3\vec{u_3}\end{aligned}$$

Then we get:

$$\vec{v} + \vec{w} = (4+11)\vec{u_1} + (-7+9)\vec{u_2} + (6+3)\vec{u_3}$$

The same idea applies in general.

Note that when we add, some of the coefficients may become zero, so the sum may end up "not using" all the vectors used in the descriptions of the individual vectors. However, *a priori*, we do not know whether any vectors will become unused after adding.

- *Sclaar multiplication*: If $\vec{v}$ is in the span of S, and $a \in \mathbb{R}$, then $a\vec{v}$ is also in the span of S: Whatever linear combination we use to describe $\vec{v}$, we can multiply each of the coefficients there with a to describe $a\vec{v}$ as a linear combination of the same set.

Suppose W is a sub*space* of $\mathbb{R}^n$, i.e., it is a nonempty subset of $\mathbb{R}^n$ closed under addition and scalar multiplication. Then, a subset S of W is termed a *spanning set* for W if W is the span of S. Note that if $S_1 \subseteq S_2 \subseteq W$ with W a subspace, and S_1 is a spanning set for W, then S_2 is also a spanning set for W.

3.4. **Why do we care about spanning sets and spans?** Vector spaces such as $\mathbb{R}^n$, and their subspaces, are *infinite* (with the exception of the zero space, which has only one element, namely the zero vector). Describing a subspace of $\mathbb{R}^n$ by listing out all the elements will take infinite time. Thus, it is useful to have a way to describe subspaces of $\mathbb{R}^n$ using finite data. Finite spanning sets offer a convenient method.

Note, however, that there can be many different candidates for a spanning set for a vector space. We will later on consider a concept of *basis*, which can be defined as a *minimal* spanning set. Even if we use basis instead of spanning set, however, we do not have any form of uniqueness. While this is a drawback, we will later see ways to figure out whether two sets have the same span.

4. Image and kernel

4.1. **Image of a linear transformation is a subspace.** Suppose $T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation and A is the matrix of T. The *image* of T (also called the *range* of T) is the subset of $\mathbb{R}^n$ comprising those vectors that can be written in the form $T(\vec{v})$ where $\vec{v} \in \mathbb{R}^m$. The question is: *what does the image of T look like*?

First, it is easy to see from abstract considerations that the image of T is a subspace of $\mathbb{R}^n$. We show this in three parts:

- The zero vector is in the image of any linear transformation, since it is the image of the zero vector.
- *Addition*: Suppose $\vec{u_1}$ and $\vec{u_2}$ are vectors in the image (range) of T. Our goal is to show that $\vec{u_1} + \vec{u_2}$ is in the image of T.

  Proof sketch: By the definition of what it means to be in the image, there exist vectors $\vec{v_1}, \vec{v_2} \in \mathbb{R}^m$ such that $\vec{u_1} = T(\vec{v_1})$ and $\vec{u_2} = T(\vec{v_2})$. Thus, $\vec{u_1} + \vec{u_2} = T(\vec{v_1}) + T(\vec{v_2}) = T(\vec{v_1} + \vec{v_2})$ (using the linearity of T). Thus, $\vec{u_1} + \vec{u_2}$ is in the image of T. Hence, the image of T is closed under addition.
- *Scalar multiplication*: Suppose $\vec{u}$ is a vector in the image of T and a is a real number. Our goal is to show that the vector $a\vec{u}$ is in the image of T.

  Proof sketch: By the definition of what it means to be in the image, there exists a vector $\vec{v} \in \mathbb{R}^m$ such that $T(\vec{v}) = \vec{u}$. Then, $a\vec{u} = aT(\vec{v}) = T(a\vec{v})$. In particular, $a\vec{u}$ is in the image of T.

Note that both these proofs rely on the *abstract* conception of linearity. In particular, the proofs above also work for "infinite-dimensional vector spaces" – something we shall return to after a while.

4.2. **Column vectors span the image.** The goal now is to *explicitly describe* the image as a vector space. The explicit description is as the span of a finite set of vectors, namely, the column vectors of the matrix. This explicit description crucially relies on the concrete description of the linear transformation.

Recall that the matrix A for $T : \mathbb{R}^m \to \mathbb{R}^n$ is a $n \times m$ matrix. The columns of A describe the values $T(\vec{e_1})$, $T(\vec{e_2})$, and so on, right till $T(\vec{e_m})$ (the last column). Thus, each column of A, viewed as a column vector, is in the image of T.

There are other vectors in the image of T, but these are essentially described as *linear combinations* of these column vectors. By *linear combination*, I mean a sum of scalar multiples of the given vectors (as described a little while back). Explicitly, consider a generic vector of the domain $\mathbb{R}^m$:

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_m \end{bmatrix} = x_1\vec{e_1} + x_2\vec{e_2} + \dots + x_m\vec{e_m}$$

The image of this is:

$$T\left(\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_m \end{bmatrix}\right) = T(x_1\vec{e_1} + x_2\vec{e_2} + \dots + x_m\vec{e_m}) = x_1T(\vec{e_1}) + x_2T(\vec{e_2}) + \dots + x_mT(\vec{e_m})$$

In particular, the image is a linear combination of $T(\vec{e_1})$, $T(\vec{e_2})$, $\dots$, $T(\vec{e_m})$. Conversely, every linear combination of these vectors is in the image. In other words, the image of the linear transformation T is *precisely* the set of linear combinations of $T(\vec{e_1})$, $T(\vec{e_2})$, $\dots$, $T(\vec{e_m})$.

As per the terminology introduced in a previous section, a set of vectors such that every vector in the image can be expressed as a linear combination of vectors in that set is said to be a *spanning set* for the image or is said to *span* the image. The upshot is that the image of a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is a subspace of $\mathbb{R}^n$ and that the column vectors of the matrix form a spanning set for the image. Note that its being a subspace follows both directly and from the fact that the span of any set of vectors is a subspace.

For instance, consider the linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^3$ with matrix:

$$\begin{bmatrix} 3 & 4 \\ 2 & 6 \\ 1 & 5 \end{bmatrix}$$

Note that this is a 3×2 matrix, as expected, since the linear transformation goes from $\mathbb{R}^2$ to $\mathbb{R}^3$. The image of $\vec{e_1}$ under the linear transformation is the column vector $\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ and the image of $\vec{e_2}$ under the linear transformation is the column vector $\begin{bmatrix} 4 \\ 6 \\ 5 \end{bmatrix}$.

The image is thus the set of all vectors that can be expressed as linear combinations of these two vectors. Explicitly, it is the set of all vectors of the form (secretly, the input vector is $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$):

$$x_1 \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} 4 \\ 6 \\ 5 \end{bmatrix}, x_1, x_2 \in \mathbb{R}$$

The generic vector in the image could alternatively be written in the form:

$$\begin{bmatrix} 3x_1 + 4x_2 \\ 2x_1 + 6x_2 \\ x_1 + 5x_2 \end{bmatrix}, x_1, x_2 \in \mathbb{R}$$

We can therefore use the two column vectors as our spanning set:

$$\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 6 \\ 5 \end{bmatrix}$$

4.3. **Kernel of a linear transformation is a subspace.** The kernel of a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is the set of those vectors in the domain that get mapped to the zero vector. In other words, the kernel is the fiber (inverse image) corresponding to the zero vector.

It turns out that the kernel of a linear transformation is also a subspace:

- The zero vector is in the kernel of any linear transformation, because its image is the zero vector.
- *Addition*: We want to show that if $\vec{v}$ and $\vec{w}$ are both vectors in the kernel of T, then $\vec{v} + \vec{w}$ is also in the kernel of T. This follows from the additivity of T:

$$T(\vec{v} + \vec{w}) = T(\vec{v}) + T(\vec{w}) = 0 + 0 = 0$$

- *Scalar multiplication*: If $\vec{v}$ is in the kernel of T and a is a real number, then $a\vec{v}$ is also in the kernel of T. This follows from the scalar multiples aspect of T:

$$T(a\vec{v}) = aT(\vec{v}) = a(0) = 0$$

Thus, the kernel is a subspace. Note that the argument used relied only on the *abstract* definition of linearity and therefore applies to vector spaces in the abstract setting, and also to infinte-dimensional spaces.

4.4. **Finding a spanning set for the kernel.** Finding the kernel is effectively the same as solving the system with the matrix as coefficient matrix and the augmenting column the zero column. We solve this the usual way: first, convert the matrix to reduced row echelon form. We can now describe the kernel the way we have done in the past: parametrically, as a solution set. Alternatively, we can try to describe the kernel using a spanning set. To describe it in the latter way, do the following: for each non-leading variable, take the case where that variable takes the value 1 and all the other non-leading variables take the value 0. Find the values of the leading variables, and construct a vector from these. Thus, for each non-leading variable, we get a corresponding vector. This set of vectors spans the kernel.

We will illustrate both the longer and shorter approach using a worked example in the next section.

4.4.1. *A worked-out example, the long way.* Suppose that the reduced row-echelon form of a matrix turns out to be:

$$\begin{bmatrix} 1 & 3 & 8 & 0 & 10 \\ 0 & 0 & 0 & 1 & 7 \end{bmatrix}$$

There are three non-leading variables: the second, third, and fifth variable. Let's first use the parametric solution description for the kernel. Suppose t is the fifth variable, u is the third variable, and v is the second variable. Then, the general solution can be written as:

$$\begin{bmatrix} -3v - 8u - 10t \\ v \\ u \\ -7t \\ t \end{bmatrix}$$

Let us separate out the multiples of t, u, and v here. We can write this as:

$$\begin{bmatrix} -3v \\ v \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -8u \\ 0 \\ u \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -10t \\ 0 \\ 0 \\ -7t \\ t \end{bmatrix}$$

If we now "pull out" the variables from their respective expressions, we obtain:

$$v \begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + u \begin{bmatrix} -8 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + t \begin{bmatrix} -10 \\ 0 \\ 0 \\ -7 \\ 1 \end{bmatrix}$$

Note that t, u, v all vary freely over all of $\mathbb{R}$. Thus, the kernel is spanned by the vectors:

$$\begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -8 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -10 \\ 0 \\ 0 \\ -7 \\ 1 \end{bmatrix}$$

Setting up names for the parameters, writing the general expression, and then pulling the variables out is a somewhat time-consuming process. Let's now consider a shorter way of doing the same example.

4.4.2. *The same worked example, a shorter way.*

- Our spanning set will include one vector corresponding to each non-leading variable, i.e., each column without a pivotal 1.
- The vector corresponding to a particular non-leading variable is obtained as follows: set the value of that variable to equal 1, and set the values of all the other non-leading variables to equal 0. Now, calculate the values of the *leading* variables based on the linear system. The value of a particular leading variable is simply the negative of the entry for the pivotal row of the leading variable in the column of the non-leading variable.

Let's first take the case where the second variable is 1 while the third and fifth variable are 0. In this case, the first variable is -3 and the fourth variable is 0, so we get the vector:

$$\begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The -3 out there comes from the fact that the entry in the first row and second column is 3. The fourth variable is 0 because it comes after the second variable.

The vector corresponding to the third variable being 1 while the second and fifth are 0 is:

$$\begin{bmatrix} -8 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The -8 out there comes from the fact that the entry in the first row and third column is 8. The fourth variable is 0 because it comes after the third variable.

The vector corresponding to the fifth variable being 1 while the second and third are 0 is:

$$\begin{bmatrix} -10 \\ 0 \\ 0 \\ -7 \\ 1 \end{bmatrix}$$

The first variable is -10 because the entry in the first row and fifth column is 10. The fourth variable is -7 because the entry in the *second* row (the row where the pivotal variable is the fourth variable) and fifth column is 7.

If you compare this shorter approach with the longer approach that involves writing the parameters explicitly, you should after a while be able to see just how the shorter approach is a shortening of the longer approach.

5. Fibers of a linear transformation

5.1. **Fibers are translates of the kernel.** The fibers of a linear transformation are the inverse images of specific points in the co-domain. To find the fiber (inverse image) for a particular point, we need to solve the linear system with that as the augmenting column. The solution set is either empty or is described parametrically.

Thus, the fibers over points not in the image are empty. For points that are in the image, the corresponding fiber looks like a translate of the kernel. Explicitly, any nonempty fiber looks like:

(particular solution vector) + (arbitrary element of the kernel)

This is extremely crucial to understand, so let's go over it explicitly.

Suppose $T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation. We have a vector $\vec{v} \in \mathbb{R}^n$. We will show the following two things:

- If $\vec{u}$ is a vector in $\mathbb{R}^m$ such that $T(\vec{u}) = \vec{v}$, and $\vec{w}$ is in the kernel of T, then $T(\vec{u} + \vec{w}) = \vec{v}$. In other words, adding any element of the kernel to an element in the fiber keeps one within the fiber.

  Proof: We have that:

$$T(\vec{u} + \vec{w}) = T(\vec{u}) + T(\vec{w}) = \vec{v} + 0 = \vec{v}$$

- If there are two elements in the same fiber, they differ by an element in the kernel. Explicitly, if $\vec{u_1}, \vec{u_2} \in \mathbb{R}^m$ are such that $T(\vec{u_1}) = T(\vec{u_2}) = \vec{v}$, then $\vec{u_1} - \vec{u_2}$ is in the kernel of T. In other words, each fiber must involve only a *single* translate of the kernel.

  Proof: We have:

$$T(\vec{u_1} - \vec{u_2}) = T(\vec{u_1}) - T(\vec{u_2}) = \vec{v} - \vec{v} = 0$$

In other words, each non-empty fiber is a *single* translate of the kernel. For instance, imagine the linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$ with matrix:

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

The kernel of this linear transformation is the subspace generated by the vector $\vec{e_2}$. The image of this linear transformation is the space generated by the vector $\vec{e_2}$.

If we use x to denote the first coordinate and y to denote the second coordinate, the map can be written as:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ x \end{bmatrix}$$

The kernel is the y-axis and the image is also the y-axis.

Now, consider a vector, say:

$$\vec{v} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$

We know that the vector is in the image of the linear transformation. What's an example of a vector that maps to it? Clearly, since this transformation sends $\vec{e_1}$ to $\vec{e_2}$, the following vector $\vec{u}$ gets mapped to $\vec{v}$ under the transformation:

$$\vec{u} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

Now, however, note that since the kernel comprises the scalar multiples of the vector $\vec{e_2}$, we can add any scalar multiple of $\vec{e_2}$ to the vector $\vec{u}$ and get a new vector that maps to $\vec{v}$. Explicitly, the fiber over $\vec{v}$ is the set:

$$\left\{ \begin{bmatrix} 5 \\ y \end{bmatrix} \mid y \in \mathbb{R} \right\}$$

Thus, the fiber over the point $(0,5)$ in the image is the line $x = 5$. Notice that this is a line parallel to the kernel of the linear transformation. In fact, all the fibers are lines parallel to the kernel of the linear transformation. In this case, they are all vertical lines.

Let's look at another, somewhat more complicated example:

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

Explicitly, the map is:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} x+y \\ 0 \end{bmatrix}$$

The map sends both $\vec{e_1}$ and $\vec{e_2}$ to $\vec{e_1}$. Hence, the matrix is idempotent.

The kernel is the line $y = -x$, and it is spanned by the vector:

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

The image is the line $y = 0$, i.e., the x-axis, so it is spanned by the vector:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The inverse image of the vector:

$$\vec{v} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

is the empty set, because this vector is not in the image of the linear transformation. On the other hand, the inverse image of the vector:

$$\vec{w} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

is the set of all vectors on the line $x + y = 2$. Note that this is a parallel line to the kernel $x + y = 0$. Also note that the linear transformation is idempotent, and this explains the fact that every point in the image is in its own fiber.

5.2. **In terms of solving linear systems.** Let's think a bit more about this in terms of solving linear systems. If the linear system has full row rank, then row reducing gives a consistent system and we can read off the solutions after row reducing. Thus, *the linear transformation is surjective.* Each fiber looks like a translate of the kernel.

If the system does not have full row rank, there are rows that, after converting to rref, become zero in the coefficient matrix. Start with an augmenting column describing a vector in the co-domain. If the corresponding augmenting entry after doing the row operations is nonzero, the system is inconsistent, so the augmenting column is not in the image, or equivalently, the fiber is empty. If, however, after row reduction, all zero rows in the coefficient matrix have zero as the augmenting entry, then the system is consistent. We can take the particular solution vector as a vector where each leading variable is the corresponding augmenting column value and the non-leading variables are all zero. This is clearly a solution. We can add anything from the kernel to it and still get a solution.

6. Dimension computation

6.1. **Dimension of subspace.** The *dimension* of a subspace is defined as the minimum possible size of a spanning set. The dimension of the zero subspace is zero, because by convention we consider the span of the empty set to be zero (this may be unclear right now but will hopefully become clearer as we proceed). The dimension of $\mathbb{R}^p$ itself is p.

6.2. **Dimension of kernel and image.** Suppose $T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation. The matrix A of T is a $n \times m$ matrix. We have seen the following in the past:

- T is injective if and only if A has full column rank, i.e., rank m. This implies $m \leq n$.
  Explanation: For T to be injective, we need that the linear system $A\vec{x} = \vec{y}$ always has at most one solution. This requires that there be no non-leading variables, i.e., the rref of A should have all its columns as pivotal columns. This forces full column rank.
- T is surjective if and only if A has full row rank, i.e., rank n. This implies $n \leq m$.
  Explanation: For T to be surjective, we need that the linear system $A\vec{x} = \vec{y}$ always has at least one solution, regardless of $\vec{y}$. In particular, this means that we cannot have any zero rows in the rref of A, because a zero row would open up the possibility that the corresponding augmenting entry is nonzero and the system could therefore be inconsistent for some $\vec{y}$. This possibility needs to be avoided, so we need that all rows of A be nonzero, i.e., all of them contain pivotal 1s. This forces A to have full row rank.

Also note that:

- *Injectivity* is the same as the kernel being the zero subspace: This was noted at the end of our discussion on fibers.
- *Surjectivity* is the same as the image being the whole target space $\mathbb{R}^n$.

Thus, we get that:

- The kernel of T is the zero subspace if and only if T has full column rank, i.e., rank m. This implies $m \leq n$.
- The image of T is all of $\mathbb{R}^n$ if and only if T has full row rank, i.e., rank n. This implies $n \leq m$.

We can generalize these to give formulas for the dimension of the kernel and the image:

- The dimension of the kernel is (number of columns) - (rank). In terms of systems of linear equations, it is the number of non-leading variables. Intuitively, this is because the non-leading variables are the parameters in an equational description. Our explicit construction of a spanning set for the kernel gave one vector for each non-leading variable. In fact, a spanning set constructed this way is minimal.
- The dimension of the image is the rank: Although our original choice of spanning set for the image was the set of *all* column vectors, we may not need all column vectors. If the map is not injective, then some of the column vectors are *redundant* (we will discusss this term later). It will turn out that it suffices to use the column vectors (in the original matrix) corresponding to the leading variables. Note that determining which variables are leading and which variables are non-leading requires us to convert the matrix to rref, but we need to use the columns of the original matrix, *not* those of the rref. We will discuss this more after introducing the concept of basis.

6.3. **Facts about dimension we will prove later.** We will show later that if A is a vector subspace of a vector space B, then the dimension of A is less than or equal to the dimension of B. In fact, we will show that any minimal spanning set of A (also called a *basis* for A) can be extended (non-uniquely) to a minimal spanning set of B.

7. Kernel and image for some easy and useful types of linear transformations

7.1. **Definition of projection.** A linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ is termed a *projection* if $T^2 = T$. Here, T^2 denotes the composite $T \circ T$ of T with itself. Another term used to describe a projection is *idempotent*, and the term *idempotent* is typically used for the matrix corresponding to a projection (the term "idempotent" means "equal to its own square" and is used in all algebraic structures, whereas projection is a term specific to thinking of transformations).

If T is a projection, then the intersection of the kernel of T and the image of T comprises only the zero vector. However, this is not a sufficient condition to describe projections.

Here is an example of a projection (discussed earlier in the notes, when talking about fibers):

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

This linear transformation sends $\vec{e_1}$ to itself and sends the vector $\vec{e_2}$ to the vector $\vec{e_1}$. Doing it twice is the same as doing it once.

Note that the matrix is already in rref, so we can read off the kernel as well as the image:

- The kernel is the one-dimensional space spanned by the vector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. To see this, note that the second variable is non-leading, and setting that to 1 gives the first variable as -1.
- The image is the one-dimensional space spanned by $\vec{e_1}$. We can see this from the fact that the image is spanned by the column vectors, and that in this case, both the column vectors are the same.

Pictorially, both the kernel and the image are lines. If we call the coordinate directions x and y, then the kernel is the line $y = -x$ and the image is the line $y = 0$ (i.e., the x-axis). These lines intersect at a unique point (the origin).

Note that it is possible to have a linear transformation whose kernel and image intersect only in the zero vector, but such that the linear transformation is not a projection. In fact, any linear automorphism other than the identity automorphism is of this form: the kernel is the zero subspace, and the image is the whole space, so the intersection is the zero subspace, so the kernel and image intersect in the zero subspace, but the map is not a projection. As a concrete example, consider the linear transformation with matrix:

$$\begin{bmatrix} 2 \end{bmatrix}$$

The kernel of this is the zero subspace, and the image is all of $\mathbb{R}$. It is not idempotent. To see this, note that $2^2 = 4$ which differs from 2.

7.2. **Orthogonal projection.** We define $T : \mathbb{R}^n \to \mathbb{R}^n$ to be an *orthogonal projection* if it is a projection and every vector in the kernel of T is orthogonal to every vector in the image of T. Note that the projection described above, namely:

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

is *not* an orthogonal projection, i.e., it does *not* have orthogonal kernel and image.

Suppose an orthogonal projection $T : \mathbb{R}^n \to \mathbb{R}^n$ has rank m. That means that the image is m-dimensional and the kernel is $(n - m)$-dimensional. The kernel and image are orthogonal subspaces.

For now, we consider a special case of orthogonal projection: *the orthogonal projection to the subspace of the first few coordinates*. Explicitly, consider a vector space $\mathbb{R}^n$ and consider the projection map that preserves the first m coordinates, with $0 < m < n$, and sends the remaining coordinates to zero. The matrix for such a linear transformation has the following block form:

$$\begin{bmatrix} I_m & 0 \\ 0 & 0 \end{bmatrix}$$

where I_m is the $m \times m$ identity matrix. Explicitly, the first m diagonal entries are 1. The remaining $n - m$ entries on the diagonal are 0. All the entries that are not on the diagonal are also zero.

In the case $n = 2$, $m = 1$, the projection is given by the matrix:

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

Physically, this is the orthogonal projection onto the x-axis.

If P is an orthogonal projection matrix, then $P^2 = P$. This makes sense, because projecting something and then projecting again should have the same effect as doing one projection.

7.3. **Linear transformations that project, permute, and pad.** A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ could be constructed that combines aspects from many different types of linear transformations:

- It could look at only r of the m coordinates and forget the remaining $m - r$.
- It could then map these r coordinates into separate coordinates of $\mathbb{R}^n$.
- It could then fill in zeros for the remaining coordinates.

The matrix for this linear transformation has rank r. There are r columns with one 1 and the other $(n-1)$ entries 0. The remaining $m-r$ columns are all zeros. Similarly, there are r rows with one 1 and the other $(m-1)$ entries zero. The remaining $n-r$ rows are all zeros.

Here's what we can say about the kernel and image:

- The kernel comprises those vectors that have zeros in all the r coordinates that matter, and can have arbitrary entries in the other coordinates. It thus has dimension $m-r$, and has a spanning set comprising those $m-r$ standard basis vectors.
- The image has dimension r, and has a spanning set comprising the standard basis vectors for the coordinates that get hit.

Here's an example of a linear transformation of this sort from $\mathbb{R}^3$ to $\mathbb{R}^4$:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \mapsto \begin{bmatrix} x_3 \\ 0 \\ x_2 \\ 0 \end{bmatrix}$$

The kernel here is spanned by $\vec{e_1}$ in $\mathbb{R}^3$ and the image is spanned by $\vec{e_1}$ and $\vec{e_3}$ in $\mathbb{R}^4$. The kernel is one-dimensional and the image is two-dimensional.

8. Intersection and sum of vector spaces

8.1. **Intersection of vector subspaces.** Suppose U and V are vector subspaces of $\mathbb{R}^n$. The intersection $U \cap V$ of U and V as sets is then also a vector subspace of $\mathbb{R}^n$. This is fairly easy to see: if both U and V individually contain the zero vector, so does the intersection. If both U and V are closed under addition, so is the intersection. Finally, if both U and V are closed under scalar multiplication, so is the intersection.

8.2. **Sum of vector subspaces.** Suppose U and V are vector subspaces of $\mathbb{R}^n$. We define $U+V$ as the subset of $\mathbb{R}^n$ comprising vectors expressible as the sum of a vector in U and a vector in V. The sum $U+V$ is a vector space: clearly it contains the 0 vector (since $0 = 0+0$). In addition, it is closed under addition and scalar multiplication:

- *Addition*: Suppose $\vec{w_1}$, $\vec{w_2}$ are vectors in $U+V$. Our goal is to show that $\vec{w_1}+\vec{w_2}$ is also in $U+V$. We know that for $\vec{w_1}$, we can find vectors $\vec{u_1} \in U$ and $\vec{v_1} \in V$ such that $\vec{w_1} = \vec{u_1} + \vec{v_1}$. Similarly, for $\vec{w_2}$, we can find vectors $\vec{u_2} \in U$ and $\vec{v_2} \in V$ such that $\vec{w_2} = \vec{u_2} + \vec{v_2}$. Thus, we have that:

$$\vec{w_1} + \vec{w_2} = (\vec{u_1} + \vec{v_1}) + (\vec{u_2} + \vec{v_2}) = (\vec{u_1} + \vec{u_2}) + (\vec{v_1} + \vec{v_2})$$

  Since U is a vector space, we know that $\vec{u_1} + \vec{u_2} \in U$. And since V is a vector space, we know that $\vec{v_1} + \vec{v_2} \in V$. Thus, the vector $\vec{w_1} + \vec{w_2}$ is in $U+V = W$.
- *Scalar multiplication*: Suppose $\vec{w} \in W$ and $a \in \mathbb{R}$. We want to show that $a\vec{w} \in W$. Note that by the definition of W as $U+V$, there exists a vector $\vec{u} \in U$ and $vecv \in V$ such that $\vec{w} = \vec{u} + \vec{v}$. Then:

$$a\vec{w} = a(\vec{u} + \vec{v}) = a\vec{u} + a\vec{v}$$

9. Sum of linear transformations, kernel, and image

In general, adding two linear transformations can play havoc with the structure, particularly if the linear transformations look very different from each other. We explore the kernel and image separately.

9.1. **Effect of adding linear transformations on the kernel.** Suppose $T_1, T_2 : \mathbb{R}^m \to \mathbb{R}^n$ are linear transformations. Note that although for convenience we describe these as maps from $\mathbb{R}^m$ to $\mathbb{R}^n$, whatever we say generalizes to maps between arbitrary vector spaces, including infinite-dimensional ones (with the caveat that statements about dimension would have to accommodate the possibility of the dimension being infinite). The sum $T_1 + T_2$ is also a linear transformation. What is the relation between the kernels of T_1, T_2, and $T_1 + T_2$? The kernels satisfy a "flower arrangement" which means that the following are true:

- If a vector $\vec{u} \in \mathbb{R}^m$ is in the kernels of both T_1 and T_2, then $\vec{u}$ is also in the kernel of $T_1 + T_2$. To see this, note that:

$$(T_1 + T_2)(\vec{u}) = T_1(\vec{u}) + T_2(\vec{u}) = 0 + 0 = 0$$

- If a vector $\vec{u} \in \mathbb{R}^m$ is in the kernels of both T_1 and $T_1 + T_2$, then $\vec{u}$ is also in the kernel of T_2. To see this, note that:

$$T_2(\vec{u}) = (T_1 + T_2)(\vec{u}) - T_1(\vec{u}) = 0 - 0 = 0$$

- If a vector $\vec{u} \in \mathbb{R}^m$ is in the kernels of both T_2 and $T_1 + T_2$, then $\vec{u}$ is also in the kernel of T_1. To see this, note that:

$$T_1(\vec{u}) = (T_1 + T_2)(\vec{u}) - T_2(\vec{u}) = 0 - 0 = 0$$

- In particular, this means that the intersection of the kernels of T_1, T_2, and $T_1 + T_2$ equals the intersection of any two of the three.

Note that the kernel of $T_1 + T_2$ could be a lot *bigger* than the kernels of T_1 and T_2. For instance, if T_1 is the identity transformation and $T_2 = -T_1$, then both T_1 and T_2 have zero kernel but the kernel of $T_1 + T_2$ is the entire domain.

9.2. **Effect of adding linear transformations on the image.** Using the same notation as for the preceding section, the following results hold regarding the relationship between the image of T_1, the image of T_2, and the image of $T_1 + T_2$:

- The image of $T_1 + T_2$ is contained in the sum of the image of T_1 and the image of T_2.
- The image of T_1 is contained in the sum of the image of T_2 and the image of $T_1 + T_2$.
- The image of T_2 is contained in the sum of the image of T_1 and the image of $T_1 + T_2$.
- Combining, we get that the sum of the images of all three of T_1, T_2, and $T_1 + T_2$ equals the sum of the images of any two of the three.

10. Composition, kernel, and image

10.1. **Dimensions of images of subspaces.** Suppose $T : U \to V$ is a linear transformation and W is a subspace of U. Then, $T(W)$ is a subspace of V, and the dimension of $T(W)$ is less than or equal to the dimension of W. It's fairly easy to see this: if S is a spanning set for W, then $T(S)$ is a spanning set for $T(W) = \mathrm{Im}(T)$.

In fact, there is a stronger result, which we will return to later:

$$\dim(W) = \dim(\mathrm{Im}(T)) + \dim(\ker(T))$$

10.2. **Dimensions of inverse images of subspaces.** Suppose $T : U \to V$ is a linear transformation and X is a linear subspace of V. Then, $T^{-1}(X)$, defined as the set of vectors in U whose image is in X, is a subspace of U. We have the following lower and upper bounds for the dimension of $T^{-1}(X)$:

$$\dim(\mathrm{Ker}(T)) \le \dim(T^{-1}(X)) \le \dim(\mathrm{Ker}(T)) + \dim(X)$$

The second inequality becomes an equality if and only if $X \subseteq \mathrm{Im}(T)$.

The first inequality is easy to justify: since X is a vector space, the zero vector is in X. Thus, everything in $\mathrm{Ker}(T)$ maps inside X, so that:

$$\mathrm{Ker}(T) \subseteq T^{-1}(X)$$

Based on the observation made above about dimensions, we thus get the first inequality:

$$\dim(\mathrm{Ker}(T)) \le \dim(T^{-1}(X))$$

The second inequality is a little harder to show directly. Suppose $Y = T^{-1}(X)$. Now, you may be tempted to say that $X = T(Y)$. That is not quite true. What is true is that $T(Y) = X \cap \mathrm{Im}(T)$. Based on the result of the preceding section:

$$\dim(Y) = \dim(\mathrm{Ker}(T)) + \dim(T(Y))$$

Using the fact that $T(Y) \subseteq X$ we get the second inequality.

These generic statements are easiest to justify using the case of projections.

10.3. **Composition, injectivity, and surjectivity in the abstract linear context.** The goal here is to adapt the statements we made earlier about injectivity, surjectivity, and composition to the context of linear transformations.

Suppose $T_1 : U \to V$ and $T_2 : V \to W$ are linear transformations. The composite $T_2 \circ T_1 : U \to W$ is also a linear transformation. The following are true:

(1) If T_1 and T_2 are both injective, then $T_2 \circ T_1$ is also injective. This follows from the corresponding statement for arbitrary functions.
 In linear transformation terms, if the kernel of T_1 is the zero subspace of U and the kernel of T_2 is the zero subspace of V, then the kernel of $T_2 \circ T_1$ is the zero subspace of U.
(2) If T_1 and T_2 are both surjective, then $T_2 \circ T_1$ is also surjective. This follows from the corresponding statement for arbitrary functions.
 In linear transformation terms, if the image of T_1 is all of V and the image of T_2 is all of W, then the image of $T_2 \circ T_1$ is all of W.
(3) If T_1 and T_2 are both bijective, then $T_2 \circ T_1$ is also bijective. This follows from the corresponding statement for arbitrary functions. Bijective linear transformations between vector spaces are called *linear isomorphisms*, a topic we shall return to shortly.
(4) If $T_2 \circ T_1$ is injective, then T_1 is injective. This follows from the corresponding statement for functions.
 In linear transformation terms, if the kernel of $T_2 \circ T_1$ is the zero subspace of U, the kernel of T_1 is also the zero subspace of U.
(5) If $T_2 \circ T_1$ is surjective, then T_2 is surjective. This follows from the corresponding statement for functions.
 In linear transformation terms, if $T_2 \circ T_1$ has image equal to all of W, then the image of T_2 is also all of W.

10.4. **Composition, kernel and image: more granular formulations.** The above statements about injectivity and surjectivity can be modified somewhat to keep better track of the *lack* of injectivity and surjectivity by using the dimension of the kernel and image. Explicitly, suppose $T_1 : U \to V$ and $T_2 : V \to W$ are linear transformations, so that the composite $T_2 \circ T_1 : U \to W$ is also a linear transformation. We then have:

(1) The dimension of the kernel of the composite $T_2 \circ T_1$ is at *least* equal to the dimension of the kernel of T_1 and at *most* equal to the sum of the dimensions of the kernels of T_1 and T_2 respectively. This is because the kernel of $T_2 \circ T_1$ can be expressed as $T_1^{-1}(T_2^{-1}(0))$, and we can apply the upper and lower bounds discussed earlier. Note that the lower bound is realized through an actual subspace containment: the kernel of T_1 is contained in the kernel of $T_2 \circ T_1$.
(2) The dimension of the image of the composite $T_2 \circ T_1$ is at *most* equal to the minimum of the dimensions of the images of T_1 and T_2. The bound in terms of the image of T_2 is realized interms of an explicit subspace containment: the image of $T_2 \circ T_1$ is a subspace of the image of T_2. The bound in terms of the dimension of T_1 is more subtle: the image of $T_2 \circ T_1$ is an image under T_2 of the image of T_1, therefore its dimension is at most equal to the dimension of the image of T_1.

10.5. **Consequent facts about matrix multiplication and rank.** Suppose A is a $m \times n$ matrix and B is a $n \times p$ matrix, where m, n, p are (possibly equal, possibly distinct) positive integers. Suppose the rank of A is r and the rank of B is s. Recall that the rank equals the dimension of the image of the associated linear transformation. The difference between the dimension of the domain and the rank equals the dimension of the kernel of the associated linear transformation. We will later on use the term *nullity* to describe the dimension of the kernel. The nullity of A is $n - r$ and the nullity of B is $p - s$.

The matrix AB is a $m \times p$ matrix. We can say the following:

- The rank of AB is at *most* equal to the minimum of the ranks of A and B. In this case, it is at most equal to $\min\{r, s\}$. This follows from the observation about the dimension of the image of a composite, made in the preceding section.
- The nullity of AB is at *least* equal to the nullity of B. In particular, in this case, it is at least equal to $p - s$. Note that the information about the maximum value of the rank actually allows us to improve this lower bound to $p - \min\{r, s\}$, though that is not directly obvious at all.
- In particular, if the product AB has full column rank p, then B has full column rank p, $s = p$, and $r \geq p$ (so in particular $m \geq p$ and $n \geq p$).

  Note that this is the linear algebra version of the statement that if the composite $g \circ f$ is injective, then f must be injective.
- If the product AB has full row rank m, then A has full row rank m, $r = m$, and $s \geq m$ (so in particular $n \geq m$ and $p \geq m$).

  Note that this is the linear algebra version of the statement that if $g \circ f$ is surjective, then g is surjective.

LINEAR DEPENDENCE, BASES, AND SUBSPACES

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Sections 3.2 and 3.3.

Executive summary

(1) A *linear relation* between a set of vectors is defined as a linear combination of these vectors that is zero. The *trivial* linear relation refers to the trivial linear combination being zero. A nontrivial linear relation is any linear relation other than the trivial one.
(2) The trivial linear relation exists between any set of vectors.
(3) A set of vectors is termed *linearly dependent* if there exists a nontrivial linear relation between them, and *linearly independent* otherwise.
(4) Any set of vectors containing a linearly dependent subset is also linearly dependent. Any subset of a linearly independent set of vectors is a linearly independent set of vectors.
(5) The following can be said of sets of small size:
  - The empty set (the only possible set of size zero) is considered linearly independent.
  - A set of size one is linearly dependent if the vector is the zero vector, and linearly independent if the vector is a nonzero vector.
  - A set of size two is linearly dependent if either one of the vectors is the zero vector or the two vectors are scalar multiples of each other. It is linearly independent if both vectors are nonzero and they are not scalar multiples of one another.
  - For sets of size three or more, a *necessary* condition for linear independence is that no vector be the zero vector and no two vectors be scalar multiplies of each other. However, this condition is not sufficient, because we also have to be on the lookout for other kinds of linear relations.

(6) Given a nontrivial linear relation between a set of vectors, we can use the linear relation to write one of the vectors (any vector with a nonzero coefficient in the linear relation) as a linear combination of the other vectors.
(7) We can use the above to prune a spanning set as follows: given a set of vectors, if there exists a nontrivial linear relation between the vectors, we can use that to write one vector as a linear combination of the others, and then remove it from the set *without affecting the span*. The vector thus removed is termed a *redundant vector*.
(8) A *basis* for a subspace of $\mathbb{R}^n$ is a linearly independent spanning set for that subspace. Any finite spanning set can be pruned down (by repeatedly identifying linear relations and removing vectors) to reach a basis.
(9) The size of a basis for a subspace of $\mathbb{R}^n$ depends only on the choice of subspace and is *independent* of the choice of basis. This size is termed the *dimension* of the subspace.
(10) Given an ordered list of vectors, we call a vector in the list *redundant* if it is redundant relative to the preceding vectors, i.e., if it is in the span of the preceding vectors, and *irredundant* otherwise. The irredundant vectors in any given list of vectors form a basis for the subspace spanned by those vectors.
(11) Which vectors we identify as redundant and irredundant depends on how the original list was ordered. However, the *number* of irredundant vectors, insofar as it equals the dimension of the span, does not depend on the ordering.
(12) If we write a matrix whose column vectors are a given list of vectors, the linear relations between the vectors correspond to vectors in the kernel of the matrix. Injectivity of the linear transformation given by the matrix is equivalent to linear independence of the vectors.

(13) Redundant vector columns correspond to non-leading variables and irredundant vector columns correspond to leading variables if we think of the matrix as a coefficient matrix. We can row-reduce to find which variables are leading and non-leading, then look at the irredundant vector columns in the *original* matrix.

(14) *Rank-nullity theorem*: The nullity of a linear transformation is defined as the dimension of the kernel. The nullity is the number of non-leading variables. The rank is the number of leading variables. So, the sum of the rank and the nullity is the number of columns in the matrix for the linear transformation, aka the dimension of the domain. See Section 3.7 of the notes for more details.

(15) The problem of finding all the vectors orthogonal to a given set of vectors can be converted to solving a linear system where the rows of the coefficient matrix are the given vectors.

1. Linear relation

1.1. **Preliminaries.**

Previously, we have seen the concepts of *linear combination*, *span*, and *spanning set*. We also saw the concept of the *trivial* linear combination: this is the linear combination where all the coefficients we use are zero. The trivial linear combination gives rise to the zero vector.

We now move to a disturbing observation: it is possible that a nontrivial linear combination of vectors give rise to the zero vector. For instance, consider the three vectors:

$$\vec{v_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \vec{v_2} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \vec{v_3} = \begin{bmatrix} 3 \\ 10 \end{bmatrix}$$

We note that:

$$\vec{v_3} = 3\vec{v_1} + 5\vec{v_2}$$

Thus, we get that:

$$\vec{0} = 3\vec{v_1} + 5\vec{v_2} + (-1)\vec{v_3}$$

In other words, the zero vector arises as a *nontrivial* linear combination of these vectors.

We will now codify and study such situations.

1.2. **Linear relation and nontrivial linear relation.**

Suppose $\vec{v_1}, \vec{v_2}, \dots, \vec{v_r}$ are vectors. A *linear relation* between $\vec{v_1}, \vec{v_2}, \dots, \vec{v_r}$ involves a choice of (possibly equal, possibly distinct, and possibly some zero) real numbers $a_1, a_2, \dots, a_r$ such that:

$$a_1\vec{v_1} + a_2\vec{v_2} + \dots + a_r\vec{v_r} = \vec{0}$$

The linear relation is termed a *trivial* linear relation if *all* of $a_1, a_2, \dots, a_r$ are 0. Note that for *any* collection of vectors, the trivial linear relation between them exists. Thus, the trivial linear relation is not all that interesting, but it is included for completeness' sake.

The more interesting phenomenon is that of a *nontrivial* linear relation. Note here that nontriviality requires that at least one of the coefficients be nonzero, but it does not require that *all* coefficients be nonzero.

The existence of nontrivial linear relations is not a given; there may be sets of vectors with *no* nontrivial linear relation. Let's introduce some terminology first, then explore the meaning of the ideas.

1.3. **Linear dependence and linear independence.**

Consider a non-empty set of vectors. We say that the set of vectors is *linearly dependent* if there exists a nontrivial linear relation between the vectors. A non-empty set of vectors is called *linearly independent* if it is not linearly dependent, i.e., there exists *no* nontrivial linear relation between the vectors in the set.

We begin with this observation: If a subset of a set of vectors is linearly dependent, then the whole set of vectors is also linearly dependent. The justification is that a nontrivial linear relation within a subset gives a nontrivial linear relation in the whole set by extending to zero coefficients for the remaining vectors. For instance, suppose $\vec{v_1}$, $\vec{v_2}$, $\vec{v_3}$, and $\vec{v_4}$ are vectors and suppose we have a linear relation between $\vec{v_1}$, $\vec{v_2}$, and $\vec{v_3}$:

$$3\vec{v_1} + 4\vec{v_2} + 6\vec{v_3} = \vec{0}$$

Then, we also have a linear relation between the four vectors $\vec{v_1}$, $\vec{v_2}$, $\vec{v_3}$, and $\vec{v_4}$:

$$3\vec{v_1} + 4\vec{v_2} + 6\vec{v_3} + 0\vec{v_4} = \vec{0}$$

An obvious corollary of this is that any subset of a linearly *independent* set is linearly independent.

1.4. **Sets of size zero.** By convention, the empty set is considered linearly independent.

1.5. **Sets of size one: linear dependence and independence.** A set of size one is:

- linearly *dependent* if the vector is the zero vector
- linearly *independent* if the vector is a nonzero vector

In particular, this means that any set of vectors that contains the zero vector must be linearly dependent.

1.6. **Sets of size two: linear dependence and independence.** A set of size two is:

- linearly *dependent* if either one of the vectors is zero or both vectors are nonzero but the vectors are scalar multiples of each other, i.e., they are in the same line.
- linearly *independent* if both vectors are nonzero and they are not scalar multiples of each other.

Pictorially, this means that the vectors point in different directions.

A corollary is that if we have a set of two or more vectors and two vectors in the set are scalar multiples of each other, then the set of vectors is linearly dependent.

1.7. **Sets of size more than two.** For sets of size more than two, linear relations could be fairly elaborate. For instance, a linear relation involving three vectors may occur even if no individual vector is a multiple of any other. Such a linear relation relies on one vector being "in the plane" of the other two vectors. For instance, if one vector is the average of the other two vectors, that creates a linear relation. Explicitly, consider the case of three vectors:

$$\vec{v_1} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \vec{v_2} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \vec{v_3} = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$$

Notice that the middle vector is the average of the first and third vector. Thus, we get the linear relation:

$$\vec{v_2} = \frac{\vec{v_1} + \vec{v_3}}{2}$$

We can rearrange this as a linear relation:

$$\vec{0} = \frac{1}{2}\vec{v_1} - \vec{v_2} + \frac{1}{2}\vec{v_3}$$

Note that this is not the only linear relation possible. Any multiple of this also defines a linear relation, albeit, an *equivalent* linear relation. For instance, we also have the following linear relation:

$$\vec{0} = \vec{v_1} - 2\vec{v_2} + \vec{v_3}$$

1.8. **Rewriting a linear relation as one vector in terms of the others.** Given a *nontrivial* linear relation between vectors, we can rewrite that relation in the form of expressing one vector as a linear combination of the other vectors. Here's the reasoning:

- We can find a vector that is being "used" nontrivially, i.e., the coefficient in front of that vector is nonzero.
- We can move that vector to the other side of the equality.
- Divide both sides by its coefficient.

For instance, consider the linear relation:

$$3\vec{v_1} + 7\vec{v_2} + 0\vec{v_3} + 9\vec{v_4} = \vec{0}$$

Note that the coefficient on $\vec{v_3}$ is 0. So, we cannot use this linear relation to write $\vec{v_3}$ in terms of the other vectors. However, we can write $\vec{v_1}$ in terms of the other vectors, or we can write $\vec{v_2}$ in terms of the other vectors, or we can write $\vec{v_4}$ in terms of the other vectors. Let's take the example of $\vec{v_2}$.

We have:

$$3\vec{v_1} + 7\vec{v_2} + 9\vec{v_4} = \vec{0}$$

We can isolate the vector $\vec{v_2}$:

$$3\vec{v_1} + 9\vec{v_4} = -7\vec{v_2}$$

We can now divide both sides by -7 to get:

$$\frac{-3}{7}\vec{v_1} + \frac{-9}{7}\vec{v_4} = \vec{v_2}$$

or, written the other way around:

$$\vec{v_2} = \frac{-3}{7}\vec{v_1} + \frac{-9}{7}\vec{v_4}$$

2. Span, basis, and redundancy

2.1. **Span and basis: definitions.** Suppose V is a vector subspace of $\mathbb{R}^n$. Let's recall what this means: V contains the zero vector, and it is closed under addition and scalar multiplication.

Recall that a subset S of V is termed a *spanning set* for V if the span of S is V, i.e., if every vector in V, and no vector outside V, can be expressed as a linear combination of the vectors in S.

A *basis* for V is a spanning set for V that is linearly independent. Note that any linearly independent set is a basis for the subspace that it spans.

2.2. **Pruning our spanning set.** As before, suppose V is a vector subspace of $\mathbb{R}^n$ and S is a spanning set for V. Suppose that S is not a basis for V, because S is not linearly independent. In other words, there is at least one nontrivial linear relation between the elements of S.

Pick one nontrivial linear relation between the elements of S. As described in an earlier section, we can use this relation to write one vector as a linear combination of the others. Once we have achieved this, we can "remove" this vector without affecting the span, because it is *redundant* relative to the other vectors. In other words, removing a redundant vector (a vector in S that is a linear combination of the other vectors in S) does not affect the span of S. This is because for any vector in the span of S that can be expressed as a linear combination using the redundant vector, the redundant vector can be replaced by the linear combination of the other vectors that it is.

Explicitly, suppose we have a relation of the form:

$$\vec{v_1} + 2\vec{v_2} - \vec{v_3} = \vec{0}$$

We use this to write $\vec{v_3}$ in terms of $\vec{v_1}$ and $\vec{v_2}$:

$$\vec{v_3} = \vec{v_1} + 2\vec{v_2}$$

Now, consider an arbitrary vector $\vec{v}$ expressible in terms of these:

$$\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + a_3\vec{v_3}$$

Using the expression above, replace $\vec{v_3}$ by $\vec{v_1} + 2\vec{v_2}$ to get:

$$\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + a_3(\vec{v_1} + 2\vec{v_2})$$

This simplifies to:

$$\vec{v} = (a_1 + a_3)\vec{v_1} + (a_2 + 2a_3)\vec{v_2}$$

In other words, getting rid of $\vec{v_3}$ doesn't affect the span: if something can be written as a linear combination using $\vec{v_3}$, it can also be written as a linear combination without using $\vec{v_3}$. So, we can get rid of $\vec{v_3}$.

2.3. **We can get rid of vectors only one at a time!** In the example above, we noted that it is possible to get rid of the vector $\vec{v_3}$ based on the linear relation that nontrivially uses $\vec{v_1}, \vec{v_2}, \vec{v_3}$. Thus, $\vec{v_3}$ is redundant relative to $\vec{v_1}$ and $\vec{v_2}$, so we can remove $\vec{v_3}$ from our spanning set.

We could have similarly argued that $\vec{v_2}$ is redundant relative to $\vec{v_1}$ and $\vec{v_3}$, and therefore, that $\vec{v_2}$ can be removed while preserving the span.

We could also have similarly argued that $\vec{v_1}$ is redundant relative to $\vec{v_2}$ and $\vec{v_3}$, and therefore, that $\vec{v_1}$ can be removed while preserving the span.

In other words, we could remove *any* of the vectors $\vec{v_1}, \vec{v_2}, \vec{v_3}$ that are involved in a nontrivial linear relation.

However, we *cannot* remove them together. The reason is that once one of the vectors is removed, that destroys the linear relation as well, so the other two vectors are no longer redundant based on this particular linear relation (they may still be redundant due to other linear relations). In a sense, every time we use a linear relation to remove one redundant vector, we have "used up" the linear relation and it cannot be used to remove any other vectors.

This suggests something: it is not so much an issue of *which* vectors are redundant, but rather a question of *how many.* At the core is the idea of dimension as a size measure. We now turn to that idea.

2.4. **Repeated pruning, and getting down to a basis.** As before, suppose V is a vector subspace of $\mathbb{R}^n$ and S is a *finite* spanning set for V. Our goal is to find a subset of S that is a basis for V.

If S is already linearly independent, that implies in particular that it is a basis for V, and we are done.

If S is *not* already linearly independent, there exists a nontrivial linear relation in S. Then, by the method discussed in the preceding section, we can get rid of one element of S and get a smaller subset that still spans V.

If this new subset is linearly independent, then we have a basis. Otherwise, repeat the process: find a nontrivial linear relation within this smaller spanning set, and use that to get rid of another vector.

The starting set S was finite, so we can perform the process only finitely many times. Thus, after a finite number of steps, we will get to a subset of S that is linearly independent, and hence a basis for V.

In the coming section, we will discuss various computational approaches to this pruning process. Understanding the process conceptually, however, is important for a number of reasons that shall become clear later.

2.5. **Basis and dimension.** Recall that we had defined the *dimension* of a vector space as the minimum possible size of a spanning set for the vector space.

The following are equivalent for a subset S of $\mathbb{R}^n$:

- S is a linearly independent set.
- S is a basis for the subspace that it spans.
- The size of S equals the minimum possible size of a spanning set for the span of S.
- The size of S equals the dimension of the span of S.

Now, given a subspace V of $\mathbb{R}^n$, there are many different possibilities we can choose for a basis of V. For instance, if V has a basis comprising the vectors $\vec{v_1}$ and $\vec{v_2}$, we could choose another basis comprising the vectors $\vec{v_1}$ and $\vec{v_1} + \vec{v_2}$. Even for one-dimensional spaces, we have many different choices for a basis of size one: any nonzero vector in the space will do.

Although there are many different possibilities for the basis, the *size* of the basis is an invariant of the subspace, namely, it is the dimension. The specific vectors used can differ, but the number needed is determined.

The concept of dimension can be understood in other related ways. For instance, the dimension is the number of independent parameters we need in a parametric description of the space. The natural parameterization of the subspace is by specifying a basis and using the coefficients for an arbitrary linear combination as the parameters. For instance, if $\vec{v}_1$ and $\vec{v}_2$ form a basis for a subspace V of $\mathbb{R}^n$, then any vector $\vec{v} \in V$ can be written as:

$$\vec{v} = a_1\vec{v}_1 + a_2\vec{v}_2, \qquad a_1, a_2 \in \mathbb{R}$$

We can think of the coefficients a_1, a_2 (which we will later call the *coordinates*, but that's for next time) as the parameters in a parametric description of V. Different choices of these give different vectors in V, and as we consider all the different possible choices, we cover everything in V.

Note: We have *not* proved all parts of the statement above. Specifically, it is not *prima facie* clear why every basis should have the minimum possible size. In other words, we have not ruled out the *prima facie* possibility that there is a basis of size two and also a basis of size three. The abstract proof that any two bases must have the same size follows from a result called the "exchange lemma" that essentially involves a gradual replacement of the vectors in one basis by the vectors in the other. The proof uses the same sort of reasoning as our pruning idea. There are also other concrete proofs that rely on facts you have already seen about linear transformations and matrices.

Another way of framing this is that the dimension is something *intrinsic* to the subspace rather than dependent on how we parameterize it. It is an intrinsic geometric invariant of the subspace, having to do with the innards of the underlying linear structure.

3. Finding a basis based on a spanning set

3.1. **Redundant vectors in order.** The method above gives an *abstract* way of concluding that any spanning set can be trimmed down to a basis. The version stated above, however, is not a *practical* approach. The problem is that we don't yet know how to find a nontrivial linear relation. Or at least, we know it ... but not consciously. Let's make it conscious.

First, let's introduce a new, more computationally relevant notion of the redundancy of a vector. Consider an *ordered* list of vectors. In other words, we are given the vectors in a particular order. A vector in this list is termed *redundant* if it is redundant relative to the vectors that appear *before* it. Intuitively, we can think of it as follows: we are looking at our vectors one by one, reading from left to right along our list. Each time, we throw in the new vector, and *potentially* expand the span. In fact, one of these two cases occurs:

- The vector is redundant relative to the set of the preceding vectors, and therefore, it contributes nothing new to the span. Therefore, we do not actually need to add it in.
- The vector is irredundant relative to the set of the preceding vectors, and therefore, it adds a new dimension (literally and figuratively) to the span.

If we can, by inspection, determine whether a given vector is redundant relative to the vectors that appear before it, we can use that to determine the span. Basically, each time we encouter a redundant vector, we don't add it.

Thus, the sub-list comprising those vectors that are irredundant in the original ordered list gives a basis for the span of the original list.

For instance, suppose we have a sequence of vectors:

$$\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_5}, \vec{v_6}, \vec{v_7}$$

Let's say that $\vec{v_1}$ is the zero vector. Then, it is redundant, so we don't add in. Let's say $\vec{v_2}$ is nonzero. So it is irredundant relative to what's appeared before (which is nothing), so we have so far built:

$$\vec{v_2}$$

Now, let's say $\vec{v_3}$ is a scalar multiple of $\vec{v_2}$. In that case, $\vec{v_3}$ is redundant and will not be added. Let's say $\vec{v_4}$ is again a scalar multiple of $\vec{v_2}$. Then, $\vec{v_4}$ is also redundant, and should not be added. Suppose now that $\vec{v_5}$ is not a scalar multiple of $\vec{v_2}$. Then, $\vec{v_5}$ is irredundant relative to the vectors that have appeared so far, so it deserves to be added:

$$\vec{v_2}, \vec{v_5}$$

We now consider the sixth vector $\vec{v_6}$. Suppose it is expressible as a linear combination of $\vec{v_2}$ and $\vec{v_5}$. Then, it is redundant, and should not be included. Now, let's say $\vec{v_7}$ is not a linear combination of $\vec{v_2}$ and $\vec{v_5}$. Then, $\vec{v_7}$ is irredundant relative to the preceding vectors, so we get:

$$\vec{v_2}, \vec{v_5}, \vec{v_7}$$

This forms a basis for the span of the original list of seven vectors. Thus, the original list of seven vectors spans a three-dimensional space, with the above as one possible basis.

Note that which vectors we identity as redundant and which vectors we identity as irredundant depends heavily on the manner in which we sequence our vectors originally. Consider the alternative arrangement of the original sequence:

$$\vec{v_4}, \vec{v_1}, \vec{v_3}, \vec{v_2}, \vec{v_7}, \vec{v_5}, \vec{v_6}$$

The irredundant vectors here are:

$$\vec{v_4}, \vec{v_7}, \vec{v_5}$$

Note that, because we ordered our original list differently, the *list* of irredundant vectors differs, so we get a different basis. But the *number* of irredundant vectors, i.e., the *size* of the basis, is the same. After all, this is the *dimension* of the space, and as such, is a geometric invariant of the space.

3.2. **The matrix and linear transformation formulation.** The problem we want to explicitly solve is the following:

> Given a collection of m vectors in $\mathbb{R}^n$, find which of the vectors are redundant and which are irredundant, and use the irredundant vectors to construct a basis for the spanning set for that collection of vectors.

Consider the $n \times m$ matrix A for a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$. We know that the columns of A are the images of the standard basis vectors under T, and thus, the columns of A form a spanning set for the image of T.

The problem that we are trying to solve is therefore equivalent to the following problem:

> Given a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ with matrix A, consider the columns of A, which coincide with the images of the standard basis vectors. Find the irredundant vectors there, and use those to get a basis for the image of T.

3.3. **Linear relations form the kernel.** We make the following observation regarding linear relations:

> Linear relations between the column vectors of a matrix A correspond to vectors in the kernel of the linear transformation given by A.

Let's understand this. Suppose A is the matrix for a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$, so that A is a $n \times m$ matrix. The columns of A are the vectors $T(\vec{e_1})$, $T(\vec{e_2})$, $\dots$, $T(\vec{e_m})$. These also form a spanning set for the image of T.

Now, suppose there is a linear relation between the vectors, namely a relation of the form:

$$x_1 T(\vec{e_1}) + x_2 T(\vec{e_2}) + \dots + x_m T(\vec{e_m}) = \vec{0}$$

Then, this is equivalent to saying that:

$$T(x_1\vec{e_1} + x_2\vec{e_2} + \dots + x_m\vec{e_m}) = \vec{0}$$

or equivalently:

$$T\left(\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_m \end{bmatrix}\right) = \vec{0}$$

In other words, the vector:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_m \end{bmatrix}$$

is in the kernel of T.

All these steps can be done in reverse, i.e., if a vector $\vec{x}$ is in the kernel of T, then its coordinates define a linear relation between $T(\vec{e_1})$, $T(\vec{e_2})$, $\dots$, $T(\vec{e_m})$.

3.4. **Special case of injective linear transformations.** Consider a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ with matrix A, which is a $n \times m$ matrix. The following are equivalent:

- T is an injective linear transformation.
- If you think of solving the linear system with coefficient matrix A, all variables are leading variables.
- A has full column rank m.
- The kernel of T is zero-dimensional, i.e., it comprises only the zero vector.
- The images of the standard basis vectors are linearly independent.
- The images of the standard basis vectors form a basis for the image.
- The image of T is m-dimensional.

In particular, all these imply that $m \leq n$.

3.5. **Back to finding the irredundant vectors.** Recall that when we perform row reductions on the coefficient matrix of a linear system, we *do not change the solution set*. This is exactly why we can use row reduction to solve systems of linear equations, and hence, also find the kernel.

In particular, this means that when we row reduce a matrix, we *do not change the pattern of linear relations between the vectors*. This means that the information about which columns are redundant and which columns are irredundant does not change upon row reduction.

For a matrix in reduced row-echelon form, the columns corresponding to the leading variables are irredundant and the columns corresponding to the non-leading variables are redundant. The leading variable columns are irredundant relative to the preceding columns because each leading variable columns uses a new row for the first time. The non-leading variable columns are redundant because they use only the existing rows for which standard basis vectors already exist in preceding leading variables. Consider, for instance:

$$\begin{bmatrix} 1 & 2 & 7 & 0 & 4 \\ 0 & 0 & 0 & 1 & 6 \end{bmatrix}$$

The first and fourth variable here are leading variables. The second, third, and fifth variable are non-leading variables. The column vectors are:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 7 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

- The first column vector corresponds to a leading variable and is irredundant. In fact, it is the first standard basis vector.
- The second column vector corresponds to a non-leading variable and is redundant: Note that it does not use any new rows. It uses only the first row, for which we already have the standard basis vector. In fact,the second column vector is explicitly twice the first column vector.
- The third column vector corresponds to a non-leading varable and is redundant: The reasoning is similar to that for the second column vector. Explicitly, this third column vector is 7 times the first column vector.
- The fourth column vector corresponds to a leading variable and is irredundant: Note that it is the first vector to use the second row. Hence, it is not redundant relative to the preceding vectors.
- The fifth column vector corresponds to a non-leading variable and is redundant: It uses both rows, but we already have standard basis vectors for both rows from earlier. Hence, it is a linear combination

of those. Explicitly, it is 4 times the first column vector plus 6 times the fourth column vector. Thus, it is redundant.

The detailed example above hopefully illustrates quite clearly the *general* statement that the column vectors corresponding to leading variables are irredundant whereas the column vectors corresponding to non-leading variables are redundant. Note that all this is being said for a matrix that is already in reduced row-echelon form. But we already noted that the linear relations between the columns are invariant under row reductions. So whatever we conclude after converting to rref about which columns are redundant and irredundant *also* applies to the original matrix.

Thus, the following algorithm works:

- Convert the matrix to reduced row-echelon form. Actually, it suffices to convert the matrix to row-echelon form because all we really need to do is identify which variables are leading variables and which variables are non-leading variables.
- The columns in the *original* matrix corresponding to the leading variables are the irredundant vectors, and form a basis for the image. Please note that the actual column vectors we use are the column vectors of the original matrix, not of the rref.

3.6. **Procedural note regarding the kernel.** We had earlier seen a procedure to find a spanning set for the kernel of a linear transformation. It turns out that the spanning set obtained that way, providing one vector for each non-leading variable, is actually linearly independent, and hence, gives a basis for the kernel. The dimension of the kernel is thus equal to the number of non-leading variables, or equivalently, equals the total number of columns minus the rank.

3.7. **Rank and nullity.** We define the *nullity* of a linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ as the dimension of the kernel of T. We will return a while later to the concept of nullity in more gory detail. For now, we state a few simple facts about rank and nullity that will hopefully clarify much of what will come later.

Suppose A is the matrix of T, so A is a $n \times m$ matrix. The following are true:

- The nullity of A is the dimension of the kernel of T.
- The rank of A is the dimension of the image of T.
- The sum of the rank of A and the nullity of A is m.
- The nullity of A is at most m.
- The rank of A is at most $\min\{m, n\}$.
- The nullity of A is 0 (or equivalently, the rank of A is m, so full column rank) if and only if T is injective. See the preceding section on injective transformations for more on this. Note that this forces $m \leq n$.
- The rank of A is n (so full row rank) if and only if T is surjective.

3.8. **Finding all the vectors orthogonal to a given set of a vectors.** Suppose we are given a bunch of vectors in $\mathbb{R}^n$. We want to find all the vectors in $\mathbb{R}^n$ whose dot product with any vector in this collection is zero. This process is relatively straightforward: set up a matrix whose *rows* are the given vectors, and find the kernel of the linear transformation given by that matrix.

COORDINATES

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 3.4.

Note: Section 4 of the lecture notes, added Sunday December 8, is not included in the executive summary. It is related to the material in the advanced review sheet, Section 2, discussed Saturday December 7.

Executive summary

(1) Given a basis $\mathcal{B} = (\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m)$ for a subspace $V \subseteq \mathbb{R}^n$ (note that this forces $m \le n$), every vector $\vec{x} \in V$ can be written in a unique manner as a linear combination of the basis vectors $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$. The fact that there exists a way of writing it as a linear combination follows from the fact that $\mathcal{B}$ spans V. The uniqueness follows from the fact that $\mathcal{B}$ is linearly independent. The coefficients for the linear combination are called the *coordinates* of $\vec{x}$ in the basis $\mathcal{B}$.

(2) Continuing notation from point (1), finding the coordinates amounts to solving the linear system with coefficient matrix columns given by the basis vectors $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$ and the augmenting column given by the vector $\vec{x}$. The linear transformation of the matrix is injective, because the vectors are linearly independent. The matrix, a $n \times m$ matrix, has full column rank m. The system is consistent if and only if $\vec{x}$ is actually in the span, and injectivity gives us uniqueness of the coordinates.

(3) A canonical example of a basis is the *standard* basis, which is the basis comprising the standard basis vectors, and where the coordinates are the usual coordinates.

(4) Continuing notation from point(1), in the special case that $m = n$, $V = \mathbb{R}^n$. So the basis is $\mathcal{B} = (\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n)$ and it is an alternative basis for all of $\mathbb{R}^n$ (here, alternative is being used to contrast with the standard basis; we will also use "old basis" to refer to the standard basis and "new basis" to refer to the alternative basis). In this case, the matrix S whose columns are the basis vectors $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$ is a $n \times n$ square matrix and is invertible. We will denote this matrix by S (following the book).

(5) Continuing notation from point (4), if we denote by $[\vec{x}]_{\mathcal{B}}$ the coordinates of $\vec{x}$ in the new basis, then $[\vec{x}]_{\mathcal{B}} = S^{-1}\vec{x}$ and $\vec{x} = S[\vec{x}]_{\mathcal{B}}$.

(6) For a linear transformation T with matrix A in the standard basis and matrix B in the new basis, then $B = S^{-1}AS$ or equivalently $A = SBS^{-1}$. The S on the right involves first converting from the new basis to the old basis, then we do the middle operation A on the old basis, and then we do S^{-1} to re-convert to the new basis.

(7) If A and B are $n \times n$ matrices such that there exists an invertible $n \times n$ matrix S satisfying $B = S^{-1}AS$, we say that A and B are *similar* matrices. Similar matrices have the same trace, determinant, and behavior with respect to invertibility and nilpotency. Similarity is an equivalence relation, i.e., it is reflexive, symmetric, and transitive.

(8) Suppose S is an invertible $n \times n$ matrix. The conjugation operation $X \mapsto SXS^{-1}$ from $\mathbb{R}^{n \times n}$ to $\mathbb{R}^{n \times n}$ preserves addition, scalar multiplication, multiplication, and inverses.

1. Coordinates

1.1. **Quick summary.** Suppose V is a subspace of $\mathbb{R}^n$ and $\mathcal{B}$ is a basis for V. Recall what this means: $\mathcal{B}$ is a linearly independent subset of V and the span of $\mathcal{B}$ is V. We will show that every element of V can be written in a *unique* manner as a linear combination of the vectors in $\mathcal{B}$. The coefficients used in that linear combination are the *coordinates* of the vector in the basis $\mathcal{B}$.

Although we will deal with finite-dimensional spaces here, the arguments used can be generalized to the infinite-dimensional setting.

1.2. **Justification for uniqueness.** Suppose the basis $\mathcal{B}$ has vectors $\vec{v}_1, \vec{v}_2, \dots \vec{v}_m$. A vector $\vec{x} \in V$ is to be written as a linear combination of the basis vectors. Note that there is *at least* one way of writing $\vec{x}$ as a linear combination of the vectors because that is what it means for $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$ to span the space. We now want to show that there is in fact *exactly one* such way.

Suppose there are two ways of writing $\vec{x}$ as a linear combination of these vectors:

$$\begin{aligned} \vec{x} &= a_1\vec{v}_1 + a_2\vec{v}_2 + \cdots + a_m\vec{v}_m \\ \vec{x} &= b_1\vec{v}_1 + b_2\vec{v}_2 + \cdots + b_m\vec{v}_m \end{aligned}$$

We thus get:

$$a_1\vec{v}_1 + a_2\vec{v}_2 + \cdots + a_m\vec{v}_m = b_1\vec{v}_1 + b_2\vec{v}_2 + \cdots + b_m\vec{v}_m$$

Rearranging gives us:

$$(a_1 - b_1)\vec{v}_1 + (a_2 - b_2)\vec{v}_2 + \cdots + (a_m - b_m)\vec{v}_m = 0$$

Since the vectors are linearly independent, this forces the linear relation above to be trivial, so that $a_1 - b_1 = a_2 - b_2 = \cdots = a_m - b_m = 0$. Thus, $a_1 = b_1$, $a_2 = b_2$, $\dots$, $a_m = b_m$. In other words, the choice of coefficients for the linear combination is unique.

1.3. **Another way of thinking about uniqueness.** Finding the coefficients for the linear combination is tantamount to solving the linear system for $a_1, a_2, \dots, a_m$:

$$\begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_m \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_m \end{bmatrix} = \begin{bmatrix} \vec{x} \end{bmatrix}$$

The fact that the vectors are linearly independent tells us that the kernel of the linear transformation given by the matrix:

$$\begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_m \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}$$

is the zero subspace. That's because elements of the kernel of that linear transformation correspond to linear relations between the vectors.

This in turn is equivalent to saying that the matrix above has full column rank m, and that the linear transformation is *injective*. Thus, for *every* vector $\vec{x}$, there is at most one solution to the linear system. Note that there exists a solution if and only if $\vec{x}$ is in V, the span of $\mathcal{B}$. And if there exists a solution, it is unique. (Note: We saw a number of equivalent formulations of injectivity in the "linear dependence, bases and subspaces" lectures that correspond to Sections 3.2 and 3.3 of your book).

1.4. **Finding the coordinates.** Suppose V is a subspace of $\mathbb{R}^n$ and $\mathcal{B}$ is a basis for V. For convenience, we will take $\mathcal{B}$ as an *ordered basis*, and list the vectors of $\mathcal{B}$ as $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$. For any vector $\vec{x}$ in V, we have noted above that there is a unique way of writing $\vec{x}$ in the form:

$$\vec{x} = a_1\vec{v}_1 + a_2\vec{v}_2 + \cdots + a_m\vec{v}_m$$

The coefficients $a_1, a_2, \dots, a_m$ are called the *coordinates* of the vector $\vec{x}$ in terms of the basis $\mathcal{B} = \vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$. Verbally, they describe $\vec{x}$ by saying "how much" of each vector $\vec{v}_i$ there is in $\vec{x}$. Note that the coordinates in terms of the standard basis are just the usual coordinates.

Finding the coordinates is easy: we already spilled the beans on that a while ago. We simply have to solve the linear system:

$$\begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_m \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_m \end{bmatrix} = \begin{bmatrix} \uparrow \\ \vec{x} \\ \downarrow \end{bmatrix}$$

Note that if $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_m$ are known in advance, we can perform the row reduction on that matrix in advance and store all the steps we did, then apply them to $\vec{x}$ when it is known to find the coordinates.

1.5. **The standard basis.** In the vector space $\mathbb{R}^n$, the standard basis vectors $\vec{e}_1, \vec{e}_2, \dots, \vec{e}_n$ form a basis for the *whole space* $\mathbb{R}^n$, and this basis is called the *standard basis.* Moreover, the coordinates of a vector in the standard basis are precisely its coordinates as it is usually written. For instance:

$$\begin{bmatrix} 2 \\ -1 \\ 4 \end{bmatrix} = 2\vec{e}_1 + (-1)\vec{e}_2 + 4\vec{e}_3$$

The coefficients used in the linear combination, which is what we would call the "coordinates" in the standard basis, are precisely the same as the usual coordinates: they are $2, -1, 4$ respectively.

That's why we call them the standard basis vectors! At the time we first started using the terminology "standard basis vectors" we did not have access to this justification, so it was just a phrase to remember. This also sheds some light on why we use the term *coordinates*: it generalizes to an arbitrary basis the role that the usual coordinates play with respect to the standard basis.

1.6. **The special case of a basis for the whole space.** Suppose $\mathcal{B}$ is a basis for a subspace of $\mathbb{R}^n$. Then, $\mathcal{B}$ has size n if and only if the subspace is all of $\mathbb{R}^n$. In this case, every vector in $\mathbb{R}^n$ has unique coordinates with respect to $\mathcal{B}$, and we can go back and forth between coordinates in the standard basis. Let S be the matrix:

$$S = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix}$$

For a vector $\vec{x}$, if $[\vec{x}]_{\mathcal{B}}$ denotes the vector describing the coordinates for $\vec{x}$ in the new basis, then we have:

$$S[\vec{x}]_{\mathcal{B}} = \vec{x}$$

So we obtain $[\vec{x}]_{\mathcal{B}}$ be solving the linear system with augmented matrix:

$$\begin{bmatrix} S & | & \vec{x} \end{bmatrix}$$

In this case, since the matrix S is a full rank square matrix, we can write:

$$[\vec{x}]_{\mathcal{B}} = S^{-1}\vec{x}$$

2. Coordinate transformations: back and forth

2.1. **The setup.** Suppose $T : \mathbb{R}^n \to \mathbb{R}^n$ is a linear transformation with matrix A. Suppose $\mathcal{B}$ is an ordered basis for $\mathbb{R}^n$. $\mathcal{B}$ must have size n. Denote by S the matrix whose columns are the vectors of $\mathcal{B}$.

We want to write the matrix for T with respect to basis $\mathcal{B}$. What does this mean? Let's first recall in what sense A is the matrix for T: A is the matrix for T with respect to the standard basis (i.e., the ordered basis comprising the standard basis vectors). Explicitly the matrix A takes in a vector $\vec{x}$ in $\mathbb{R}^n$ written with coordinates in the standard basis, and outputs $T(\vec{x})$, again written with coordinates in the standard basis.

The matrix for T in the basis $\mathcal{B}$ should be capable of taking as input a vector expressed using coordinates in $\mathcal{B}$ and give an output that also uses coordinates in $\mathcal{B}$. In other words, it should be of the form:

$$[\vec{x}]_{\mathcal{B}} \mapsto [T(\vec{x})]_{\mathcal{B}}$$

Here's how we go about doing this. First, we convert $[\vec{x}]_{\mathcal{B}}$ to $\vec{x}$, i.e., we rewrite the vector in the standard basis. We know from before how this works. We have:

$$\vec{x} = S[\vec{x}]_{\mathcal{B}}$$

Now that the vector is in the standard basis, we can apply the linear transformation T using the matrix A, which is designed to operate in the standard basis. So we get:

$$T(\vec{x}) = A(S[\vec{x}]_{\mathcal{B}})$$

We have now obtained the obtained, but we need to re-convert to the new basis. So we multiply by S^{-1}, and get:

$$[T(\vec{x})]_{\mathcal{B}} = S^{-1}(A(S[\vec{x}]_{\mathcal{B}}))$$

By associativity of matrix multiplication, we can reparenthesize the right side, and we obtain:

$$[T(\vec{x})]_{\mathcal{B}} = (S^{-1}AS)[\vec{x}]_{\mathcal{B}}$$

Thus, the matrix for T in the new basis is:

$$B = S^{-1}AS$$

Explicitly:

- The right-most S, which is the operation that we do first, involves converting from the new basis to the old basis (the standard basis).
- We then apply the matrix A, which describes T in the standard basis.
- The left-most S^{-1} involves re-converting to the new basis.

We can also visualize this using the following diagram:

$$\begin{tikzcd}
\vec{x} \arrow[r, "A"] & T(\vec{x}) \\
\vec{x}_{\mathcal{B}} \arrow[u, "S"] \arrow[r, "B"] & {[T(\vec{x})]_{\mathcal{B}}} \arrow[u, "S"]
\end{tikzcd}$$

Our goal is to traverse the bottom B. To do this, we go up, right, and down, i.e., we do:

$$[\vec{x}]_{\mathcal{B}} \overset{S}{\rightsquigarrow} \vec{x} \overset{A}{\rightsquigarrow} T(\vec{x}) \overset{S^{-1}}{\rightsquigarrow} [T(\vec{x})]_{\mathcal{B}}$$

We are doing S, then A, then S^{-1}. But remember that we compose from right to left, so we get:

$$B = S^{-1}AS$$

2.2. **A real-world analogy.** Suppose you have a document in Chinese and you want to prepare an executive summary of it, again in Chinese. Unfortunately, you do not have access to any worker who can prepare executive summaries of documents in Chinese. However, you *do* have access to a person who can translate from Chinese to English, a person who can translate from English to Chinese, and a person who can do document summaries of English documents (with the summary also in English). The obvious solution is three-step:

- First, translate the document from Chinese to English
- Then, prepare the summary of the document in English, giving an English summary.
- Now, translate the summary from English to Chinese

This is analogous to the change-of-basis transformation idea: here, Chinese plays the role of the "new basis", English plays the role of the "old basis", the English summary person plays the role of the matrix A (performing the transformation in the old basis), and the overall composite process corresponds to the matrix B (performing the transformation in the new basis).

3. Similar matrices and the conjugation operation

This section will be glossed over quickly in class, but may be relevant to a better understanding of quizzes and class-related material.

3.1. **Similarity as an equivalence relation.** Suppose A and B are $n \times n$ matrices. We say that A and B are *similar matrices* if there exists an invertible $n \times n$ matrix S such that the following equivalent conditions are satisfied:

- $AS = SB$
- $B = S^{-1}AS$
- $A = SBS^{-1}$

Based on the preceding discussion, this means that there is a linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ whose matrix in the standard basis is A and whose matrix in the basis given by the columns of S is B.

Similarity defines an *equivalence relation*. In particular, the following are true:

- *Reflexivity*: Every matrix is similar to itself. In other words, if A is a $n \times n$ matrix, then A is similar to A. The matrix S that we can use for similarity is the identity matrix.

  The corresponding "real-world" statement would be that a document summary person who works in English is similar to himself.
- *Symmetry*: If A is similar to B, then B is similar to A. Indeed, the matrices we use to go back and forth are inverses of each other. Explicitly, if $B = S^{-1}AS$, then $A = SBS^{-1} = (S^{-1})^{-1}BS^{-1}$.

  The corresponding "real-world" statement would involve noting that since translating between English and Chinese allows us to do Chinese document summaries using the English document summary person, we can also go the other way around: if there is a person who does document summaries in Chinese, we can use that person and back-and-forth translators to carry out document summaries in English.
- *Transitivity*: If A is similar to B, and B is similar to C, then A is similar to C. Explicitly, if $S^{-1}AS = B$ and $T^{-1}BT = C$, then $(ST)^{-1}A(ST) = C$.

  The corresponding "real-world" statement would note that since Chinese document summary persons are similar to English document summary persons, and English document summary persons are similar to Spanish document summary persons, the Chinese and Spanish document summary persons are similar to each other.

3.2. **Conjugation operation preserves addition and multiplication.** For an invertible $n \times n$ matrix S, define the following mapping:

$$\mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$$

by:

$$X \mapsto SXS^{-1}$$

This is termed the *conjugation mapping* by S. Intuitively, it involves moving from the new basis to the old basis (it's the reverse of moving from the old basis to the new basis). It satisfies the following:

- It preserves addition: $S(X + Y)S^{-1} = SXS^{-1} + SYS^{-1}$ for any two $n \times n$ matrices X and Y. We can check this algebraically, but conceptually it follows from the fact that the sum of linear transformations should remain the sum regardless of what basis we use to express the linear transformations.
- It preserves multiplication: $(SXS^{-1})(SYS^{-1}) = S(XY)S^{-1}$. We can check this algebraically, but conceptually it follows from the fact that the composite of linear transformations remains the composite regardless of what basis we write them in.
- It preserves scalar multiplication: $S(\lambda X)S^{-1} = \lambda(SXS^{-1})$ for any real number λ and any $n \times n$ matrix X.
- It preserves inverses: If X is an invertible $n \times n$ matrix, then $SX^{-1}S^{-1} = (SXS^{-1})^{-1}$.
- It preserves powers: For any $n \times n$ matrix X and any positive integer r, $(SXS^{-1})^r = SX^rS^{-1}$. If X is invertible, the result holds for negative powers as well.

Intuitively, the reason why it preserves everything is the same as the reason that translating between languages is supposed to preserve all the essential features and operations. We're just using a different language and different labels, but the underlying structures remain the same. More technically, we can think of conjugation mappings as isomorphisms of the additive and scalar-multiplicative structure.

A quick note: Real-world language translation fails to live up to these ideas, because languages of the world are not structurally isomorphic. There are some ideas expressible in a certain way in English that have no equivalent expression in Chinese, and conversely, there are some ideas expressible in a certain way in Chinese that have no equivalent expression in English. Further, the translation operations between the languages, as implemented in human or machine translators, are *not* exact inverses: if you translate a document from English to Chinese and then translate that back to English, you are unlikely to get precisely the same document. So while language translation is a good example to help build real-world intuition for why we need both the S and the S^{-1} sandwiching the original transformation, we should not take the language analogy too literally. The real-world complications of language translation far exceed the complications arising in linear algebra.[1] Thus, language translation is unlikely to be a useful guide in thinking about similarity of linear transformations. The only role is the initial pedagogy, and we have (hopefully!) accomplished that.

3.3. **What are the invariants under similarity?** Conjugation can change the appearance of a matrix a lot. But there are certain attributes of matrices that remain invariant under conjugation. In fact, we can provide a complete list of invariants under conjugation, but this will take us too far afield. So, instead, we note a few attributes that remain invariant under conjugation.

For the discussion below, we assume A and B are similar $n \times n$ matrices, and that S is a $n \times n$ matrix such that $SBS^{-1} = A$.

- *Trace*: Recall that for any two matrices X and Y, XY and YX have the same trace. Thus, in particular, $(SB)S^{-1} = A$ and $S^{-1}(SB) = B$ have the same trace. So, A and B have the same trace.
- *Invertibility*: A is invertible if and only if B is invertible. In fact, their inverses are similar via the same transformation, as mentioned earlier.
- *Nilpotency*: A is nilpotent if and only B is nilpotent. Further, if r is the smallest positive integer such that $A^r = 0$, then r is also the smallest positive integer such that $B^r = 0$.
- *Other stuff related to powers*: A is idempotent if and only if B is idempotent. A has a power equal to the identity matrix if and only if B has a power equal to the identity matrix.
- *Determinant*: The determinant is a complicated invariant that controls the invertibility and some other aspects of the matrix. We have already seen the explicit description of the determinant for 2×2 matrices. We briefly discuss the significance of the determinant in the next section.

3.4. **The determinant: what it means.** *Not covered in class, but relevant for some quizzes.*

The determinant is a number that can be computed for any $n \times n$ (square) matrix, with the following significance:

- *Whether or not the determinant is zero* determines whether the linear transformation is invertible. In particular:
  - If the determinant is zero, the linear transformation is non-invertible.
  - If the determinant is nonzero, the linear transformation is invertible.
- The *sign* of the determinant determines whether the linear transformation preserves orientation. In particular:
  - If the determinant is positive, the linear transformation is orientation-preserving.
  - If the determinant is negative, the linear transformation is orientation-reversing.
- The *magnitude* of the determinant is the factor by which volumes get scaled.

Let's consider a method to compute the determinant. Recall the various row operations that we perform in order to row reduce a matrix. These operations include:

(1) Multiply or divide a row by a nonzero scalar.
(2) Add or subtract a multiple of a row to another row.

[1]The claim may seem strange if you're finding linear algebra a lot harder than English or Chinese, but you've probably spent a lot more time in total mastering your first language than you have mastering linear algebra. Linear algebra is also easier to systematize and abstract than natural language.

(3) Swap two rows.

We now keep track of some rules for the determinant:

(1) Each time we multiply a row of a matrix by a scalar λ, the determinant gets multiplied by λ.
(2) Adding or subtracting a multiple of a row to another row preserves the determinant.
(3) Swapping two rows multiplies the determinant by -1.

Finally, the following two all-important facts:

- The determinant of a non-invertible matrix is 0.
- The determinant of the identity matrix is 1.

The procedure is now straightforward:

- Convert the matrix to rref. Keep track of the operations and note what the determinant overall gets multiplied by.
- If the rref is non-invertible, the original matrix is also non-invertible and its determinant is 0.
- If the rref is invertible, it must be the identity matrix. We thus get 1 equals (some known number) times (the unknown determinant). The unknown determinant is thus the reciprocal of that known number. For instance, if we had to multiply by $1/2$, then by -1, then by 3, to get to rref, then we have overall multiplied by $-3/2$ and the determinant is thus $-2/3$.

Here is how the determinant interacts with addition, mutliplication, and inversion of mtarices:

(1) There is no formula to find the determinant of the sum in terms of the individual determinants. Rather, it is necessary to know the actual matrices. In fact, as you know, a sum of non-invertible matrices may be invertible or non-invertible, so that rules out the possibility of a formula.
(2) The determinant of a product of two $n \times n$ matrices is the product of the determinants. In symbols, if we use det to denote the determinant, then:

$$\det(AB) = \det(A)\det(B)$$

(3) The determinant of the inverse of an invertible $n \times n$ matrix is the inverse (i.e., the reciprocal) of the determinant. In symbols:

$$\det(A^{-1}) = \frac{1}{\det A}$$

We can reconcile the observations about the product and inverse with each other, and also reconcile them with earlier observations about the significance of the magnitude and sign of the determinant.

3.5. Similarity and commuting with the change-of-basis matrix.

This subsection was added December 8.

In the special case that the change-of-basis matrix S commutes with the matrix A, we have that $S^{-1}AS = S^{-1}SA = A$.

Two other further special cases are worth noting:

- The case that S is a scalar matrix: In this case, S commutes with all possible choices of A, so this change of basis does not affect any of the matrices. Essentially, the matrix S is causing the same scaling on both the inputs and the outputs, so it does not affect the description of the linear transformation.
- The case that A is a scalar matrix: In this case, S commutes with A for all possible choices of S, so the change of basis does not affect A. Intuitively, this is because a scalar multiplication looks the same regardless of the basis.

  Another way of formulating this is that a scalar matrix is the *only* matrix in its similarity class, i.e., a scalar matrix can be similar to no *other* matrix (scalar or non-scalar).

4. Constructing similar matrices

This section was added later, and is related to material covered in the Saturday (December 7) review session.

4.1. **Similarity via the identity matrix.** This might seem too obvious to mention, but it is still worth mentioning when trying to construct examples and counterexamples: for any $n \times n$ matrix A, A is similar to itself, and we can take the change-of-basis matrix S to be the identity matrix I_n.

4.2. **Similarity via coordinate interchange.** We begin by considering the matrix:

$$S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The matrix S describes the linear transformation that interchanges the standard basis vectors $\vec{e}_1$ and $\vec{e}_2$. Explicitly, this linear transformation interchanges the coordinates, i.e., it is described as:

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} y \\ x \end{bmatrix}$$

Note that $S = S^{-1}$, or equivalently, S^2 is the identity matrix.

Now, consider two matrices A and B that are similar via S, so that:

$$A = SBS^{-1}, B = S^{-1}AS$$

We have $S = S^{-1}$, and we obtain:

$$A = SBS, B = SAS$$

Let's understand carefully what left and right multiplication by S are doing. Consider:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

We have:

$$AS = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} b & a \\ d & c \end{bmatrix}$$

We thereby obtain:

$$SAS = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} b & a \\ d & c \end{bmatrix} = \begin{bmatrix} d & c \\ b & a \end{bmatrix}$$

We could also do the multiplication in the other order.

$$SA = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} c & d \\ a & b \end{bmatrix}$$

We thereby obtain:

$$SAS = \begin{bmatrix} c & d \\ a & b \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} d & c \\ b & a \end{bmatrix}$$

The upshot is that:

$$SAS = \begin{bmatrix} d & c \\ b & a \end{bmatrix}$$

Conceptually:

- Right multiplication by S interchanges the columns of the matrix. Conceptually, AS takes as input $[\vec{x}]_{\mathcal{B}}$ ($\vec{x}$ written in the *new* basis) and gives as output the vector $T(\vec{x})$ (written in the old, i.e. standard, basis). AS has the same columns as A but in a different order, signifying that which standard basis vector goes to which column vector gets switched around.
- Left multiplication by S (conceptually, S^{-1}, though $S = S^{-1}$ in this case) interchanges the rows of the matrix. Conceptually, $S^{-1}A$ takes as input $\vec{x}$ and outputs $[T(\vec{x})]_{\mathcal{B}}$. The left multiplication by $S = S^{-1}$ signifies that after we obtain the output, we swap its two coordinates.

- Left and right multiplication by S signifies that we interchange the rows and interchange the columns. In other words, we change the row and column of each entry. This can also be thought of as "reflecting" the entries of the square matrix about the center: the entries on the main diagonal a and d get interchanged, and the entries on the main *anti*-diagonal b and c get interchanged.

To summarize, the matrices:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, SAS = \begin{bmatrix} d & c \\ b & a \end{bmatrix}$$

are similar via S.

We can now do some sanity checks for similarity.

- *Same trace?*: The trace of A is $a + d$ and the trace of SAS is $d + a$. Since addition is commutative, these are equal.
- *Same determinant?*: The matrix A has determinant $ad - bc$ whereas the matrix SAS has determinant $da - cb$. Since multiplication is commutative, these are the same.

Note that these are *sanity checks*: they don't prove anything new. Rather, their purpose is to make sure that things are working as they "should" if our conceptual framework and computations are correct.

4.3. **Linear transformations, finite state automata, and similarity.**

Recall the setup of linear transformations and finite state automata described in your quizzes as well as in the lecture notes for matrix multiplication and inversion.

We will now discuss how we can judge similarity of the linear transformations given by these matrices.

4.3.1. *The case $n = 2$: the nonzero nilpotent matrices.*

Consider the functions f and g given as follows:

$$f(0) = 0, f(1) = 2, f(2) = 0, \qquad g(0) = 0, g(1) = 0, g(2) = 1$$

The finite state automaton diagrams for f and g are as follows:

$$f : 1 \to 2 \to 0, \qquad g : 2 \to 1 \to 0$$

(there are loops at 0 that I did not write above).

The matrices are as follows:

$$M_f = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, M_g = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Both of these square to zero, i.e., we have $M_f^2 = 0$ and $M_g^2 = 0$.

The following three equivalent observations can be made:

1. If we interchange 1 and 2 on the input side and *also* interchange 1 and 2 on the output side for f, then we obtain the function g. Equivalently, if we do the same starting with g, we obtain f.
2. If we take the finite state automaton diagram for f, and interchange the labels 1 and 2, we obtain the finite state automaton diagram for g. Equivalently, if we interchange the labels starting with the diagram for g, we obtain the diagram for f.
3. If we consider the matrix S of Section 4.2:

$$S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

   then (recall that $S = S^{-1}$):

$$M_g = SM_fS = S^{-1}M_fS = SM_fS^{-1}$$

Note that the matrices M_f and M_g being similar fits in well with other observations about the matrices, namely:

(a) *Same trace*: Both matrices have trace 0.
(b) *Same determinant*: Both matrices have determinant 0 (note that this follows from their not being full rank).

(c) *Same rank*: Both matrices have rank 1.
(d) *Both are nilpotent with the same nilpotency*: Both matrices square to zero.

4.3.2. *The case $n = 2$: idempotent matrices.* Consider the functions f and g given as follows:

$$f(0) = 0, f(1) = 1, f(2) = 0, \qquad g(0) = 0, g(1) = 0, g(2) = 2$$

The finite state automaton diagram for f loops at 1, loops at 0, and sends 2 to 0. The finite state automaton for g loops at 2, loops at 0, and sends 1 to 0. The corresponding matrices are:

$$M_f = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, M_g = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

For this f and g, observations of similarity similar to the preceding case (the numbered observations (1)-(3)) apply.

Consider now two other functions h and j:

$$h(0) = 0, h(1) = 1, h(2) = 1, \qquad j(0) = 0, j(1) = 2, j(2) = 2$$

The corresponding matrices are:

$$M_h = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, M_j = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

It is clear that observations analogous to the numbered observations (1)-(3) apply to the functions h and j. Thus, M_h and M_j are similar via the change of basis matrix S.

However, the somewhat surprising fact that if the matrices M_f, M_g, M_h, and M_j are *all* similar. We have already established the similarity of M_f and M_g (via S) and of M_h and M_j (via S). To establish the similarity of all four, we need to show that M_f and M_h are similar. This is a little tricky. The idea is to use:

$$S = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

We obtain:

$$S^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

We then get:

$$S^{-1} M_f S = M_h$$

Thus, M_f and M_h are similar matrices.

Intuition behind the discovery of S: S should send $\vec{e}_1$ to $\vec{e}_1$ because that is a basis vector for the unique fixed line of M_f and M_h. So, the first column of S is $\vec{e}_1$. S should send $\vec{e}_2$ to a vector for which the second column of M_h applies, i.e., a vector that gets sent to $\vec{e}_1$ under M_f. One such vector is $\vec{e}_1 + \vec{e}_2$. Thus, the second column of S is $\vec{e}_1 + \vec{e}_2$. The matrix for S looks like:

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

We can make the following observations similar to the earlier observations in the previous subsubsection, but now for all *four* of the matirces M_f, M_g, M_h, and M_j.

(a) *Same trace*: All four matrices have trace 1.
(b) *Same determinant*: All four matrices have determinant 0. This follows from their not being of full rank.
(c) *Same rank*: All four matrices have rank 1.
(d) *All are idempotent*: $M_f^2 = M_f$, $M_g^2 = M_g$, $M_h^2 = M_h$, $M_j^2 = M_j$.

4.3.3. *The case $n = 3$: nilpotent matrices of nilpotency three.* For $n = 3$, we can construct many different choices of f such that $M_f^2 \neq 0$ but $M_f^3 = 0$. A typical example is a function f with the following finite state automaton diagram:

$$3 \to 2 \to 1 \to 0$$

The corresponding matrix M_f is:

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

We can obtain other such matrices that are similar to this matrix via coordinate permutations, i.e., via interchanging the labels 1, 2, and 3 (there is a total of 6 different matrices we could write down of this form in this similarity class).

4.3.4. *The case $n = 3$: 3-cycles.* The 3-cycles are permutations that cycle the three coordinates around. There are two such 3-cycles possible, namely the functions f and g described below.

$$f(0) = 0, f(1) = 2, f(2) = 3, f(3) = 1, \qquad g(0) = 0, g(1) = 3, g(2) = 1, g(3) = 2$$

The corresponding matrices are:

$$M_f = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, M_g = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

These matrices are similar to each other via any interchange of two coordinates. Explicitly, for instance, M_f and M_g are similar via the following matrix S (that equals its own inverse):

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Note also that $M_f^3 = M_g^3 = I_3$, and $M_g = M_f^2 = M_f^{-1}$ while at the same time $M_f = M_g^2 = M_g^{-1}$.

4.4. **Counterexample construction for similarity using finite state automata and other methods.** Below are some situations where you need to construct examples of matrices, and how finite state automata can be used to motivate the construction of relevant examples. Finite state automata are definitely not *necessary* for the construction of the examples, and in fact, in many cases, randomly written examples are highly likely to work. But using examples arising from finite state automata allows us to see exactly *where, why, and how* things fail.

4.4.1. *Similarity does not behave well with respect to addition, subtraction, and multiplication.* Our goal here is to construct matrices A_1, A_2, B_1, and B_2 such that A_1 is similar to B_1 and A_2 is similar to B_2, but the following failures of similarity hold:

(a) $A_1 + A_2$ is not similar to $B_1 + B_2$
(b) $A_1 - A_2$ is not similar to $B_1 - B_2$
(c) $A_1 A_2$ is not similar to $B_1 B_2$

Note that we need to crucially make sure that we *must* use *different* change-of-basis matrices for the change of basis from A_1 to B_1 and the change of basis from A_2 to B_2. Denote by S_1 the change-of-basis matrix that we use between A_1 and B_1 and denote by S_2 the change-of-basis matrix that we use between A_2 and B_2. The simplest strategy will be to choose matrices such that:

$$n = 2, S_1 = I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, S_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

In other words, we choose $A_1 = B_1$, and B_2 is obtained by swapping rows *and* swapping columns in A_2. For more on how the change of basis given by S_2 works, see Section 4.2.

The following examples *each* work for *all* the points (a)-(c) above:

(1) $A_1 = A_2 = B_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$, $B_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. These example matrices are discussed in Section 4.3.1. You can verify the conditions (a)-(c) manually (trace, determinant, and rank can be used to rule out similarity).

(2) $A_1 = A_2 = B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $B_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. These example matrices are discussed in Section 4.3.2. You can verify the conditions (a)-(c) manually (trace, determinant, and rank can be used to rule out similarity).

4.4.2. *Similarity behaves well with respect to matrix powers but not with respect to matrix roots.* Suppose A and B are similar $n \times n$ matrices, and p is a polynomial. Then, $p(A)$ and $p(B)$ are similar matrices, and in fact, the same matrix S for which $A = SBS^{-1}$ also satisfies $f(A) = Sf(B)S^{-1}$. In particular, for any positive integer r, $A^r = SB^rS^{-1}$. (Proof-wise, we *start* with proving the case of positive integer powers, then combine with the case for scalar multiples and addition, and obtain the statement for arbitrary polynomials).

In the case that $r = 1$, the implication works both ways, but for $r > 1$, A^r and B^r being similar does not imply that A and B are similar.

We discuss some examples:

- For r even, we can choose $n = 1$ and take $A = [1]$ and $B = [-1]$.
- *Non-invertible example based on finite state automata*: For all $r > 1$, we can choose $A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. In this case, $A^r = B^r = 0$, but A is not similar to B because the zero matrix is not similar to any nonzero matrix. Here, B is a matrix of the type discussed in Section 4.3.1.
- *Invertible example*: For all $r > 1$, we can choose $n = 2$ and take A to be the identity matrix and B to be the matrix that is counter-clockwise rotation by an angle of $2\pi/r$. In this case, $A^r = B^r = I_2$, but A is not similar to B because the identity matrix is not similar to any non-identity matrix.
- *Invertible example based on finite state automata*: For $r = 3$, we can construct examples using finite state automata. We take A as the identity matrix and to take B as one of the automata based on the 3-cycle (as described in Section 4.3.4). $A^3 = B^3 = I_3$ but A is not similar to B because the identity matrix is not similar to any non-identity matrix.

4.5. **Similarity via negation of the second coordinate.** Another example of a change-of-basis matrix S that is worth keeping handy is:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

This matrix fixes $\vec{e}_1$ (in more advanced jargon, we would say that $\vec{e}_1$ is an eigenvector with eigenvalue 1) and sends $\vec{e}_2$ to its negative (in more advanced jargon, we would say that $\vec{e}_2$ is an eigenvector with eigenvalue -1).

Note that $S = S^{-1}$ in this case.

Multiplying on the left by S means multiplying the second row by -1. Multiplying on the right by S means multiplying the second column by -1. Multiplying on both the left and the right by S multiplies both the off-diagonal entries by -1 (note that the bottom right entry gets multiplied by -1 twice, so the net effect is that it returns to its original value). Explicitly, the map sending a matrix A to the matrix SAS is:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \mapsto \begin{bmatrix} a & -b \\ -c & d \end{bmatrix}$$

To summarize, the matrices:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, SAS = \begin{bmatrix} a & -b \\ -c & d \end{bmatrix}$$

are similar.

We can now do some sanity checks for similarity.

- *Same trace*: The matrices A and SAS have the same diagonal (diagonal entries a and d). Both matrices therefore have the same trace, namely $a + d$.
- *Same determinant*: The determinant of A is $ad - bc$. The determinant of SAS is $ad - (-b)(-c)$. The product $(-b)(-c)$ is bc, so we obtain that the determinant is $ad - bc$.

Note that these are *sanity checks*: they don't prove anything new. Rather, their purpose is to make sure that things are working as they "should" if our conceptual framework and computations are correct.

Similarity via reflection plays a crucial role in explaining why the counter-clockwise rotation matrix and clockwise rotation matrix for the same angle are similar. Explicitly, if we denote the rotation matrix for θ as $R(\theta)$, then the matrices:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, R(-\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

are similar via the matrix S discussed above. Note that both these matrices have trace $2\cos\theta$ and determinant 1.

ABSTRACT VECTOR SPACES AND THE CONCEPT OF ISOMORPHISM

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Sections 4.1 and 4.2.

Executive summary

General stuff ...

(1) There is an abstract definition of real vector space that involves a set with a binary operation playing the role of addition and another operation playing the role of scalar multiplication, satisfying a bunch of axioms. The goal is to axiomatize the key aspects of vector spaces.
(2) A subspace of an abstract vector space is a subset that contains the zero vector and is closed under addition and scalar multiplication.
(3) A linear transformation is a set map between two vector spaces that preserves addition and preserves scalar multiplication. It also sends zero to zero, but this follows from its preserving scalar multiplication.
(4) The *kernel* of a linear transformation is the subset of the domain comprising the vectors that map to zero. The kernel of a linear transformation is always a subspace.
(5) The *image* of a linear transformation is its range as a set map. The image is a subspace of the co-domain.
(6) The *dimension* of a vector space is defined as the size of any basis for it. The dimension provides an upper bound on the size of any linearly independent set in the vector space, with the upper bound attained (in the finite case) only if the linearly independent set is a basis. The dimension also provides a lower bound on the size of any spanning subset of the vector space, with the lower bound being attained (in the finite case) only if the spanning set is a basis.
(7) Every vector space has a particular subspace of interest: the zero subspace.
(8) The *rank* of a linear transformation is defined as the dimension of the image. The rank is the answer to the question: "how much survives the linear transformation?"
(9) The *nullity* of a linear transformation is defined as the dimension of the kernel. The nullity is the answer to the question: "how much gets killed under the linear transformation?"
(10) The sum of the rank and the nullity of a linear transformation equals the dimension of the domain. This fact is termed the *rank-nullity theorem*.
(11) We can define the *intersection* and *sum* of subspaces. These are again subspaces. The intersection of two subspaces is defined as the set of vectors that are present in both subspaces. The sum of two subspaces is defined as the set of vectors expressible as a sum of vectors, one in each subspace. The sum of two subspaces also equals the subspace spanned by their union.
(12) A linear transformation is *injective* if and only if its kernel is the zero subspace of the domain.
(13) A linear transformation is *surjective* if and only if its image is the whole co-domain.
(14) A *linear isomorphism* is a linear transformation that is *bijective*: it is both injective and surjective. In other words, its kernel is the zero subspace of the domain and its image is the whole co-domain.
(15) The dimension is an isomorphism-invariant. It is in fact a *complete isomorphism-invariant*: two real vector spaces are isomorphic if and only if they have the same dimension. Explicitly, we can use a bijection between a basis for one space and a basis for another. In particular, any n-dimensional space is isomorphic to $\mathbb{R}^n$. Thus, by studying the vector spaces $\mathbb{R}^n$, we have effectively studied all finite-dimensional vector spaces up to isomorphism.

Function spaces ...

(1) For any set S, consider the set $F(S, \mathbb{R})$ of *all* functions from S to $\mathbb{R}$. With pointwise addition and scalar multiplication of functions, this set is a vector space over $\mathbb{R}$. If S is finite (*not* our main case

of interest) this space has dimension $|S|$ and is indexed by a basis of S. We are usually interested in *subspaces* of this space.

(2) We can define vector spaces such as $\mathbb{R}[x]$ (the vector space of all polynomials in one variable with real coefficients) and $\mathbb{R}(x)$ (the vector space of all rational functions in one variable with real coefficients). These are both infinite-dimensional spaces. We can study various finite-dimensional subspaces of these. For instance, we can define P_n as the vector space of all polynomials of degree less than or equal to n. This is a vector space of dimension $n+1$ with basis given by the monomials $1, x, x^2, \dots, x^n$.

(3) There is a natural injective linear transformation $\mathbb{R}[x] \to F(\mathbb{R}, \mathbb{R})$.

(4) Denote by $C(\mathbb{R})$ or $C^0(\mathbb{R})$ the subspace of $F(\mathbb{R}, \mathbb{R})$ comprising the functions that are continuous everywhere. For k a positive integer, denote by $C^k(\mathbb{R})$ the subspace of $C(\mathbb{R})$ comrpising those functions that are at least k times continuously differentiable, and denote by $C^\infty(\mathbb{R})$ the subspace of $C(\mathbb{R})$ comprising all the functions that are *infinitely* differentiable. We have a descending chain of subspaces:

$$C^0(\mathbb{R}) \supseteq C^1(\mathbb{R}) \supseteq C^2(\mathbb{R}) \supseteq \dots$$

The image of $\mathbb{R}[x]$ inside $F(\mathbb{R}, \mathbb{R})$ lands inside $C^\infty(\mathbb{R})$.

(5) We can view differentiation as a linear transformation $C^1(\mathbb{R}) \to C(\mathbb{R})$. It sends each $C^k(\mathbb{R})$ to $C^{k-1}(\mathbb{R})$. It is surjective from $C^\infty(\mathbb{R})$ to $C^\infty(\mathbb{R})$. The kernel is constant functions, and the kernel of k-fold iteration is P_{k-1}. Differentiation sends $\mathbb{R}[x]$ to $\mathbb{R}[x]$ and is surjective to $\mathbb{R}[x]$.

(6) We can also define a formal differentiation operator $\mathbb{R}(x) \to \mathbb{R}(x)$. This is not surjective.

(7) Partial fractions theory can be formulated in terms of saying that some particular rational functions form a basis for certain finite-dimensional subspaces of the space of rational functions, and exhibiting a method to find the "coordinates" of a rational function in terms of this basis. The advantage of expressing in this basis is that the basis functions are particularly easy to integrate.

(8) We can define a vector space of sequences. This is a special type of function space where the domain is $\mathbb{N}$. In other words, it is the function space $F(\mathbb{N}, \mathbb{R})$.

(9) We can define a vector space of formal power series. The Taylor series operator and series summation operator are back-and-forth operators between this vector space (or an appropriate subspace therefore) and $C^\infty(\mathbb{R})$.

(10) Formal differentiation is a linear transformation $\mathbb{R}[[x]] \to \mathbb{R}[[x]]$. It is surjective but not injective. The kernel is the one-dimensional space of formal power series.

(11) We can consider linear differential operators from $C^\infty(\mathbb{R})$ to $C^\infty(\mathbb{R})$. These are obtained by combining the usual differentiation operator and multiplication operators using addition, multiplication (composition) and scalar multiplication. Finding the kernel of a linear differential operator is equivalent to solving a homogeneous linear differential equation. Finding the inverse image of a particular function under a linear differential operator amounts to solving a non-homogeneous linear differential equation, and the solution set here is a translate of the kernel (the corresponding solution in the homogeneous case, also called the *auxilliary solution*) by a particular solution. The first-order case is particularly illuminative because we have an explicit formula for the fibers.

1. Abstract definitions

1.1. **The abstract definition of a vector space.** A *real vector space* (just called *vector space* for short) is a set V equipped with the following structures:

- A binary operation $+$ on V called addition that is commutative and associative. By "binary operation" we mean that it is a map $V \times V \to V$, i.e., it takes two inputs in V and gives an output in V. Explicitly, for any vectors $\vec{v}, \vec{w}$ in V, there is a vector $\vec{v} + \vec{w} \in V$. The operation is commutative and associative:
  - *Commutativity* means that for any vectors $\vec{v}, \vec{w} \in V$, $\vec{v} + \vec{w} = \vec{w} + \vec{v}$.
  - *Associativity* means that for any vectors $\vec{u}, \vec{v}, \vec{w} \in V$, $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$.
- A special element $\vec{0} \in V$ that is an identity for addition. Explicitly, for any vector $\vec{v} \in V$, we have $\vec{0} + \vec{v} = \vec{v} + \vec{0} = \vec{v}$.
- A scalar multiplication operation $\mathbb{R} \times V \to V$ denoted by concatenation such that:

– $0\vec{v} = \vec{0}$ (the $\vec{0}$ on the right side being the vector 0) for all $\vec{v} \in V$.
– $1\vec{v} = \vec{v}$ for all $\vec{v} \in V$.
– $a(b\vec{v}) = (ab)\vec{v}$ for all $a, b \in \mathbb{R}$ and $\vec{v} \in V$.
– $a(\vec{v} + \vec{w}) = a\vec{v} + a\vec{w}$ for all $a \in \mathbb{R}$ and $\vec{v}, \vec{w} \in V$.
– $(a + b)\vec{v} = a\vec{v} + b\vec{v}$ for all $a, b \in \mathbb{R}$, $\vec{v} \in V$.

Note that the vector spaces $\mathbb{R}^n$ that we have encountered are examples of real vector spaces in the sense above. However, there are many other vector spaces, such as spaces of functions, that at least superficially look very different.

1.2. **Abstract vector spaces: where do they live?** One of the difficulties that many people have with grasping abstract vector spaces is that it's not clear *where* this vector space is. With $\mathbb{R}^n$, we know what the elements (vectors) are: they are vectors with n coordinates, all of which are real numbers. We know the algebraic representation, and also have a vague geometric picture. Admittedly, the geometric picture is clear only for $n = 1, 2, 3$, but we can obtain our intuition from there and extend formally from there.

With abstract vector spaces, *where* they live could vary quite a bit based on what space we are considering. But in some sense, it doesn't matter *where they live*. What really matters is how the vectors interact with each other, i.e., how they add and scalar multiply. The addition and scalar multiplication are the essence of a vector space. How the vectors are written or what they are called is less relevant than how they add. Just like where you live or where you were born is not directly relevant to your grade: how you score on the test is. We judge vectors not by how they're written, but by the way they add and scalar multiply. We'll understand this better when we study the definition of *isomorphism* of vector spaces.

1.3. **What are all those axioms for?** The conditions such as commutativity, associativity, distributivity, etc. imposed in the abstract definition of vector space are there in order to make sure that the *key features* of the concrete vector spaces we have encountered so far are preserved in the abstract setting. Basically, what we want is that any algebraic identity or manipulation technique that we need to use in our usual proofs is available to us in the abstract setting.

1.4. **The abstract definition of subspace and linear transformation.** Fortunately, these definitions don't really differ from the definitions you are probably already familiar with from earlier. The reason is that it's only the beginning part (the foundation, so to speak) that gets more complicated in the abstract setting. The rest of it was already sufficiently abstract to begin with. Nonetheless, we review the definitions below.

A *(linear) subspace* of a vector space is defined as a nonempty subset that is closed under addition and scalar multiplication. In particular, any subspace must contain the zero vector. So, an alternative definition of subspace is that it is a subset that contains the zero vector, is closed under addition, and is closed under scalar multiplication. Note that if we just say *subspace* we are by default referring to a linear subspace.

A subspace of a vector space can be viewed as being a vector space in its own right. Note that there is one vector that we are sure every subspace must contain: the zero vector.

Suppose V and W are vector spaces. A function $T : V \to W$ is termed a *linear transformation* if T preserves addition and scalar multiplication, i.e., we have the following two conditions:

- $T(\vec{v_1} + \vec{v_2}) = T(\vec{v_1}) + T(\vec{v_2})$ for all vectors $\vec{v_1}, \vec{v_2} \in V$.
- $T(a\vec{v}) = aT(\vec{v})$ for all $a \in \mathbb{R}$, $\vec{v} \in V$.

Note that any linear transformation must send the zero vector to the zero vector. This need not be imposed as a separate condition: it follows from the scalar multiplication condition.

1.5. **Kernel and image.** The *kernel* of a linear transformation $T : V \to W$ is defined as the set of all vectors $\vec{v} \in V$ such that $T(\vec{v})$ is the zero vector. As we saw earlier, the kernel of any linear transformation is a *subspace* of V. In other words, it is non-empty (note that in particular it contains the zero vector of V), it is closed under addition, and it is closed under scalar multiplication.

The *image* of a linear transformation $T : V \to W$ is defined as the set of all vectors $\vec{w} \in W$ that can be written as $\vec{w} = T(\vec{v})$ for some vector $\vec{v} \in V$. In the language of functions, it is simply the range of T. The image of T is a *subspace* of W.

The proofs of both statements (the kernel is a subspace and the image is a subspace) are the same as those we saw earlier when introducing the concept of kernel. However, now that we are dealing with abstract vector spaces as opposed to the concrete setting, we need to be sure that every manipulation that we perform is included in, or justifiable from, the axioms in the abstract setting.

1.6. **Dimension and containment.** Suppose V is a real vector space. The *dimension* of V (as a real vector space) is defined in the following equivalent ways:

(1) It is the maximum possible size of a linearly independent set in V. Note that in the finite case, a linearly independent set has this maximum size if and only if it is a basis.
(2) It the size of any basis of V.
(3) It is the minimum possible size of a spanning set in V. Note that in the finite case, a spanning set has this minimum size if and only if it is a basis.

We call a vector space *finite-dimensional* if its dimension is finite, and *infinite-dimensional* otherwise.

The following are true for a subspace containment: suppose U is a subspace of a real vector space V. Then, the dimension of U is less than or equal to the dimension of V. If U is finite-dimensional, then the dimensions are equal if and only if $U = V$.

1.7. **The zero subspace.** Every vector space has a particular subspace of interest: the *zero subspace*. This is the subspace that contains only the zero vector. This is a zero-dimensional space. In other words, the empty set is a basis for it.

1.8. **Rank-nullity theorem.** The rank-nullity theorem holds for abstract vector spaces. Suppose $T : V \to W$ is a linear transformation from a real vector space V to a real vector space W. Suppose further that V is finite-dimensional. We do not need to assume anything regarding whether W is finite-dimensional.

Recall that the *rank* of T is defined as the dimension of the image of T. The *nullity* of T is defined as the dimension of the kernel of T. The rank-nullity theorem states that the sum of the rank of T and the nullity of T is the dimension of the domain space V.

This is the same as the old rank-nullity theorem, except that now, we are no longer thinking of things in terms of matrices, but simply in terms of abstract spaces and linear transformations between them.

1.9. **Sum and intersection of subspaces.** We had earlier defined the concepts of sum and intersection of subspaces. The same concepts apply with the same definition in the abstract setting. Explicitly:

- If U_1 and U_2 are subspaces of a real vector space V, then the intersection $U_1 \cap U_2$, defined as the set of vectors that are in both U_1 and U_2, is also a subspace of V.
- If U_1 and U_2 are subspaces of a real vector space V, then the sum $U_1 + U_2$, defined as the set of vectors that can be expressed as the sum of a vector in U_1 and a vector in U_2, is also a subspace of V.

  Note that the union $U_1 \cup U_2$ is just a subset and not in general a subspace, and in fact, $U_1 \cup U_2 \subseteq U_1 + U_2$ and $U_1 + U_2$ is the subspace spanned by $U_1 \cup U_2$. It is a subspace if and only if either $U_1 \subseteq U_2$ or $U_2 \subseteq U_1$, and further, that happens if and only if $U_1 \cup U_2 = U_1 + U_2$.

2. Isomorphism of vector spaces

2.1. **Definition of isomorphism.** Recall our general concept of *isomorphism* from earlier in the course: it is an invertible mapping that preserves the essence of the structure. In the context of vector spaces, a *linear isomorphism* between abstract vector spaces V and W is a bijective linear transformation $T : V \to W$.

If V and W are vector spaces and there exists a linear isomorphism $T : V \to W$, then we say that V and W are isomorphic.

2.2. **Isomorphism as an equivalence relation.** Isomorphism is an equivalence relation between vector spaces. Explicitly, it satisfies the following three conditions:

- *Reflexivity*: Every vector space is isomorphic to itself. We can choose the identity map as the isomorphism (though it is not the only possible isomorphism).
- *Symmetry*: If V and W are isomorphic, then W and V are isomorphic. Explicitly, if $T : V \to W$ is a linear isomorphism, then $T^{-1} : W \to V$ is a linear isomorphism.

- *Transivity*: If U and V are isomorphic, and V and W are isomorphic, then U and W are isomorphic. Explicitly, if $T_1 : U \to V$ is a linear isomorphism and $T_2 : V \to W$ is a linear isomorphism, then $T_2 \circ T_1 : U \to W$ is a linear isomorphism. Explicitly, $(T_2 \circ T_1)^{-1} = T_1^{-1} \circ T_2^{-1}$.

2.3. **Isomorphism and dimension.** The dimension of a vector space is an invariant that completely determines the isomorphism class. Explicitly, if V and W are vector spaces, then V and W are isomorphic if and only if they have the same dimension. Constructively, the isomorphism is obtained as follows: choose any set bijection from a basis of V to a basis of W, and then extend linearly to a linear isomorphism from V to W.

This is particularly useful for finite-dimensional vector spaces: given a n-dimensional vector space and a m-dimensional vector space, the vector spaces are isomorphic if and only if $n = m$. In particular, this also tells us that any n-dimensional real vector space is isomorphic to the "standard" n-dimensional vector space $\mathbb{R}^n$. Put another way, studying the spaces $\mathbb{R}^n, n \in \mathbb{N}_0$ is tantamount to studying all finite-dimensional vector spaces up to isomorphism. That's why our concreteness so far didn't really lose us much generality.

The particular case of dimension zero gives the zero space. This is isomorphic to the zero subspace in any vector space.

3. Function spaces

3.1. **The general idea of function spaces.** The idea of *function spaces* is as follows. For S any set, we can define the space of *all* functions from S to $\mathbb{R}$ and make this a real vector space with the following structure:

- The addition of functions is defined *pointwise*. Explicitly, given functions $f, g : S \to \mathbb{R}$, we define $f + g$ as the function $x \mapsto f(x) + g(x)$.
- Scalar multiplication is defined as follows: given $\lambda \in \mathbb{R}$ and $f : S \to \mathbb{R}$, λf is defined as function $x \mapsto \lambda f(x)$.

We will denote this vector space as $F(S, \mathbb{R})$.

Now, this is the space of *all* functions on the set S. We are usually interested in other vector spaces that arise as subspaces of this space. Specifically, we are interested in subsets of this space that contain the zero function (the function sending everything to zero), are closed under pointwise addition of functions, and are closed under scalar multiplication.

Note also that the "vectors" here are now "functions." This requires a bit of rethinking, because we are used to thinking of vectors as pointy arrows or equivalently as things with coordinates. Functions, on the other hand, do not look like that. But to make a vector space, we don't care about whether the things look like our preconceived notion of vectors, but rather, we care about whether they have the addition and scalar multiplication operations satisfying the conditions we have specified.

3.2. **A basis for the space of all functions when the set is finite.** If S is finite, then the space of all functions on a set S has a basis indexed by the elements of S. For each $s \in S$, define the characteristic function $\mathbf{1}_s$ as the function that sends s to 1 and sends all other elements of S to 0. For any function f, the expression for f in terms of this basis is:

$$f = \sum_{s \in S} f(s)\mathbf{1}_s$$

Note that this idea cannot be used when S is infinite because the required sum would be infinite, and vector spaces only permit finite sums.

3.3. **The vector space of polynomials.** The set of *all* polynomials with real coefficients in one variable x is a vector space, with the usual definition of addition and scalar multiplication of polynomials. This vector space is sometimes denoted $\mathbb{R}[x]$ (the book denotes this space by P). Note that there is *also* a definition of *multiplication* of polynomials but that definition is *not* part of the vector space structure.

Explicitly, an element of this vector space is of the form:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Now, any polynomial can be thought of as a function. In other words, given any polynomial p, we can think of the *function* $x \mapsto p(x)$. In other words, we have a mapping:

$$\mathbb{R}[x] \to F(\mathbb{R}, \mathbb{R})$$

that sends any polynomial to the corresponding function. We note the following about the mapping:

- The mapping is *linear*: It preserves sums and scalar multiples. What this means is that adding polynomials as *polynomials* and then considering them as functions is equivalent to considering them as functions and then adding as functions. This is no surprise: our formal method for polynomial addition is designed to mimic function addition. Also, scalar multiplying a polynomial and *then* viewing it as a function is equivalent to converting it to a function and then scalar multiplying that function.
- The mapping is *injective*: What this means is that two different polynomials can never give rise to the same function. Equivalently (since the mapping is linear) no nonzero polynomial can be the zero function. This is obvious: a nonzero polynomial of degree n has at most n real roots, hence cannot be zero at all points.

Thus, the image of $\mathbb{R}[x]$ is a subspace of $F(\mathbb{R}, \mathbb{R})$. Since the mapping is injective, we can think of this subspace as another copy of $\mathbb{R}[x]$, so we sometimes abuse notation and identify $\mathbb{R}[x]$ with that image.

3.4. **Subspaces of the space of polynomials.** The space $\mathbb{R}[x]$ of polynomials is infinite-dimensional. The following is the most convenient basis for it:

$$1, x, x^2, x^3, \dots$$

Note that this is a spanning set because every polynomial is expressible as a linear combination of a finite number of these. It is a basis because there are no linear relations between these.

We can define the following subspaces of $\mathbb{R}[x]$ of interest. For any nonnegative integer n, let P_n be the span of the subset:

$$1, x, x^2, \dots, x^n$$

In other words, P_n is the $(n+1)$-dimensional space comprising all polynomials of degree $\leq n$. The above set forms a basis for P_n.

We have the following subspace inclusions:

$$P_0 \subseteq P_1 \subseteq P_2 \subseteq P_3 \subseteq \dots$$

And the whole space is the union of all these subspaces, i.e., we have:

$$\mathbb{R}[x] = P = \bigcup_{i=0}^{\infty} P_i$$

3.5. **The vector space of rational functions.** We denote by $\mathbb{R}(x)$ the set of all rational functions, i.e., expressions of the form $p(x)/q(x)$ where p and q are both polynomials with q not the zero polynomial, up to the following *equivalence* relation: $p_1(x)/q_1(x)$ and $p_2(x)/q_2(x)$ are considered the "same" rational function if $p_1(x)q_2(x) = p_2(x)q_1(x)$.

With the usual definition of addition (take common denominator, then add numerators) and scalar multiplication (just multiply the scalar in the numerator), the space of rational functions is a vector space. Further, there is a natural injective linear transformation from the space of polynomials to the space of rational functions:

$$\mathbb{R}[x] \to \mathbb{R}(x)$$

that sends a polynomial $p(x)$ to $p(x)/1$.

The map is not surjective, because there do exist rational functions that are not polynomials.

We might be tempted to say that there is a natural map:

$$\mathbb{R}(x) \to F(\mathbb{R}, \mathbb{R})$$

However, this would be inaccurate, because the *function* defined by a rational function is not defined at the points where the denominator becomes zero. So, the above map does not *quite* make sense. There are ways of getting around the issue by fixing either the domain or the co-domain appropriately, but we shall not bother right now.

4. Evaluation functionals and related transformations

4.1. **Evaluation functional at a single point.** For any real number u, the evaluation functional eval_u is defined as a linear transformation:

$$\text{eval}_u : F(\mathbb{R}, \mathbb{R}) \to \mathbb{R}$$

given by:

$$\text{eval}_u(f) := f(u)$$

The term *linear functional* is used for a linear transformation from a vector space to the vecotr space $\mathbb{R}$ (viewed as a one-dimensional vector space over itself). The evaluation maps are linear functionals.

4.2. **Evaluation at multiple points simultaneously.** Consider a tuple $(u_1, u_2, \dots, u_n)$ of real numbers. We can define a linear transformation:

$$\text{eval}_{(u_1,u_2,\dots,u_n)} : F(\mathbb{R}, \mathbb{R}) \to \mathbb{R}^n$$

as follows:

$$f \mapsto \begin{bmatrix} f(u_1) \\ f(u_2) \\ \cdot \\ \cdot \\ \cdot \\ f(u_n) \end{bmatrix}$$

This linear transformation involves the simultaneous evaluation of f at multiple points, and it further involves storing the outputs as the coordinates of a vector.

4.3. **Evaluation transformations from smaller spaces.** Instead of considering evaluation transformations originating from $F(\mathbb{R}, \mathbb{R})$, we can consider evaluation transformations originating from smaller spaces. For instance, recall that we defined P_m as the vector space of polynomials of degree $\leq m$. This is a $(m+1)$-dimensional real vector space. Given n distinct points, we can define the evaluation transformation:

$$P_m \to \mathbb{R}^n$$

4.4. **Injectivity and surjectivity of the evaluation transformation.** The following are true:

- The evaluation transformation from a function space to $\mathbb{R}^n$ (based on evaluation at a collection of points) is *injective* if and only if the only function that evaluates to zero at all the points in that collection is the zero function.
- The evaluation transformation from a function space to $\mathbb{R}^n$ (based on evaluation at a collection of points) is *surjective* if and only if every possible tuple of output values at that collection of points arises from a function in that function space.

4.5. **Setting things up using matrices: need for choosing a basis of the space, and hence parameters.** Consider in more detail the evaluation transformation:

$$P_m \to \mathbb{R}^n$$

Note that P_m, the space of polynomials of degree $\leq m$, is an abstract vector space. Although it has dimension $(m+1)$ we do not think of it as being the same as $\mathbb{R}^{m+1}$. If we choose a basis for P_m, then we can write coordinates in that basis, and we can then think of the map as being like a map $\mathbb{R}^{m+1} \to \mathbb{R}^n$, and describe it with a $n \times (m+1)$ matrix.

The obvious choice of basis is:

$$1, x, x^2, \dots, x^m$$

Thus, for a polynomial:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

the corresponding coordinates are:

$$\begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_m \end{bmatrix}$$

Recall that, at the start of the course, we had talked about functional forms that are *linear in the parameters.* We can really think about such functional forms as describing vector spaces of the functions, with the functions appearing in front of the parameters as a basis for that function space, and the parameters specifying the coordinates of a particular function of the function space in terms of that basis. For the example of P_m, the functional form of a polynomial of degree $\leq m$ corresponds to the function space P_m. This space has basis $1, x, x^2, \dots, x^m$ and the coefficients that appear in front of these monomials in the description of a polynomial are the coordinates in that basis. These coordinates are our parameters.

Explicitly, the dictionary between our earlier jargon and our new jargon in as follows:

- Parameters for the general functional form $\leftrightarrow$ Coordinates in the chosen basis for the function space
- Inputs (of input-output pair fame) $\leftrightarrow$ Points at which we are performing the evaluation maps
- Outputs $\leftrightarrow$ Outputs of the evaluation maps

5. Function spaces: differentiation and integration

5.1. **Spaces of continuous and differentiable functions.** We denote by $C(\mathbb{R})$ or $C^0(\mathbb{R})$ the subset of $F(\mathbb{R}, \mathbb{R})$ comprising all the functions that are continuous on all of $\mathbb{R}$. This is a subspace. Here's why:

- The zero function is continuous.
- A sum of continuous functions is continuous: Remember that this follows from the fact that the limit of the sum is the sum of the limits, which in turn can be proved using the $\varepsilon - \delta$ definition of the limit (there are also alternative ways of thinking about it).
- A scalar multiple of a continuous function is continuous: This follows from the fact that the limit of a scalar multiple is the corresponding scalar multiple of the limit.

The elements ("vectors") of the vector space $C(\mathbb{R})$ are continuous functions from $\mathbb{R}$ to $\mathbb{R}$. Note that this space is pretty huge, but relative to $F(\mathbb{R}, \mathbb{R})$, it is still quite small: if you just picked a function with random values everywhere, it would probably be *very far* from continuous. In fact, it's unlikely to be continuous at *any* point.

The space $C(\mathbb{R})$, also denoted $C^0(\mathbb{R})$, has a number of interesting subspaces. For k a positive integer, we define $C^k(\mathbb{R})$ as the subspace of $C(\mathbb{R})$ comprising those continuous functions that are at least k times continuously differentiable on all of $\mathbb{R}$. Explicitly, $f \in C^k(\mathbb{R})$ if $f^{(k)}$ exists and is in $C(\mathbb{R})$. We thus have subspace inclusions:

$$C(\mathbb{R}) = C^0(\mathbb{R}) \supseteq C^1(\mathbb{R}) \supseteq C^2(\mathbb{R}) \supseteq \dots$$

The intersection of these spaces is the vector space of *infinitely* differentiable functions, denoted $C^\infty(\mathbb{R})$. Explicitly:

$$C^\infty(\mathbb{R}) = \bigcap_{k=0}^{\infty} C^k(\mathbb{R})$$

The vector space $C^\infty(\mathbb{R})$, though much smaller than $C(\mathbb{R})$, is still pretty big. It includes all polynomial functions (i.e., the image of $\mathbb{R}[x]$ in $F(\mathbb{R}, \mathbb{R})$ lives inside $C^\infty(\mathbb{R})$). It also includes rational functions where the denominator is never zero. It includes other functions involving exponentials, sines, and cosines, as long as these functions don't have zeros for their denominators.

5.2. **Differentiation as a linear transformation.** Differentiation can be defined as a linear transformation:

$$C^1(\mathbb{R}) \to C(\mathbb{R})$$

The following emerge from some thought:

- The *kernel* of this linear transformation is P_0, the space of constant functions. The kernel is one-dimensional.
- The linear transformation is surjective, i.e., its image is all of $C(\mathbb{R})$. This follows from the fact that every continuous function is the derivative of its integral.
- The fibers of differentiation, also called the "indefinite integral", are of the form (particular antiderivative) $+C$, where C is an arbitrary constant. The $+C$ arises from the fact that each fiber is a translate of the kernel, and the kernel is the space of constant functions.
- Note that $C(\mathbb{R})$ and $C^1(\mathbb{R})$ are infinite-dimensional spaces, and $C^1(\mathbb{R})$ is a proper subspace of $C(\mathbb{R})$. We thus have an interesting situation where there is a *surjective* linear transformation from a proper subspace to the whole space. Note that this kind of situation cannot arise with finite-dimensional spaces.
- For $k \geq 1$, the image of $C^k(\mathbb{R})$ under differentiation is $C^{k-1}(\mathbb{R})$. Moreover, the inverse image of $C^{k-1}(\mathbb{R})$ under differentiation is $C^k(\mathbb{R})$.
- The image of $C^\infty(\mathbb{R})$ under differentiation is $C^\infty(\mathbb{R})$. Moreover, the inverse image of $C^\infty(\mathbb{R})$ under differentiation is $C^\infty(\mathbb{R})$.
- The image of $\mathbb{R}[x]$ under differentiation is $\mathbb{R}[x]$. Moreover, the inverse image of $\mathbb{R}[x]$ under differentiation is also $\mathbb{R}[x]$. More detail: for $n \geq 1$, the image of P_n under differentiation is P_{n-1}, and the inverse image of P_{n-1} is P_n.
- The kernel of the k-fold iteration of differentiation is P_{k-1}, or rather, the image of P_{k-1} in $F(\mathbb{R}, \mathbb{R})$ (depending on whether we are thinking of differentiation as an operator $C^1(\mathbb{R}) \to C(\mathbb{R})$ or as an operator $\mathbb{R}[x] \to \mathbb{R}[x]$).
- We can also define a formal differentiation operator from $\mathbb{R}(x)$ to $\mathbb{R}(x)$ (recall the $\mathbb{R}(x)$ is not identifiable with a subspace of $C(\mathbb{R})$ because of the problem of denominator blow-up). The kernel is once again P_0. The image of $\mathbb{R}(x)$ under this differentiation operator is a subspace of $\mathbb{R}(x)$, but is *not* all of $\mathbb{R}(x)$. In other words, there do exist rational functions that do not have any rational function as their antiderivative. The function $1/(x^2+1)$, whose antiderivatives are all of the form $(\arctan x) + C$, is an example that is *not* in the image of $\mathbb{R}(x)$ under differentiation.

5.3. **Knowledge of derivatives and antiderivatives on a spanning set suffices.** Suppose V is a vector subspace of the vector space $C^\infty(\mathbb{R})$. We know that differentiation is linear. We now explore how that information is useful in computing the derivatives and antiderivatives of functions in V based on knowledge of derivatives and antiderivatives of functions in a spanning set S for V.

Let's tackle differentiation first. The first step is to express the function we want to differentiate as a linear combination of the functions in the spanning set S . Now, use the linearity of differentiation to express its derivative as the corresponding linear combination of the *derivatives* of functions in S.

For instance, suppose V is the span of sin and exp in $C^\infty(\mathbb{R})$ and we know that the derivative of sin is cos and the derivative of exp is exp. In this case, $S = \{\sin, \exp\}$. Then the derivative of the function:

$$f(x) = 2\sin x + 5\exp(x)$$

is:

$$f'(x) = 2\sin' x + 5\exp'(x) = 2\cos x + 5\exp(x)$$

More generally, given an arbitrary function:

$$f(x) = a\sin x + b\exp(x)$$

The derivative is:

$$f'(x) = a\cos x + b\exp(x)$$

Note that it is the fact of the functions *spanning* V that is crucial in allowing us to be able to write *any* function in V as a linear combination of the functions.

The computational procedure tells us a bit more: suppose S is a spanning set for a subspace V of $C^\infty(\mathbb{R})$. Suppose W is the image of V under differentiation. Then, the image of S under differentiation is a spanning set for W.

In particular, this means that if the image of S under differentiation lies inside V, then the image of V under differentiation is a subspace of V. This is what happens with the space $\mathbb{R}[x]$ of polynomials: the derivative of every power function (with nonnegative integer exponent) is a polynomial, and hence, the derivative of every polynomial is a polynomial.

Something similar applies to indefinite integration. We need to be a *little* more careful with indefinite integration, because the antiderivative is not unique. Instead, the indefinite integral of any function is a translate of the one-dimensional kernel. The obligatory $+C$ of indefinite integration is to account for the fact that the kernel of differentiation is the one-dimensional space of constant functions.

For instance, suppose we know that an antiderivative of sin is $-\cos$ and an antiderivative of exp is exp. Then, the indefinite integral of the function:

$$f(x) = 2\sin x + 5\exp(x)$$

is:

$$\int f(x)\,dx = 2(-\cos x) + 5\exp x + C$$

Let's consider some other settings where this idea has come in play:

- Consider $V = \mathbb{R}[x]$ and $S = \{1, x, x^2, \dots,\}$. The set S is a spanning set for V. In fact, S is a basis for V. We know how to diffferentiate any member of S: the power rule says that the derivative of x^n with respect to x is nx^{n-1}. This rule allows us to differentiate *any* polynomial, because a polynomial is a linear combination of these power functions. In other words, knowledge of how to differentiate functions in S tells us how to differentiate anything in V.
- Consder $V = \mathbb{R}[x]$ and $S = \{1, x, x^2, \dots,\}$ (same as above), but we are now interested in indefinite integration. Note that V already contains the constant functions (the kernel of differentiation) so we do not need to worry about the $+C$ taking us out of the space. The antiderivative of x^n is $x^{n+1}/(n+1)$. The formula is not correct at $n = -1$, but we are not considering that case here, since $n \geq 0$ here. We can use knowledge of antiderivatives of all functions in S to obtain antiderivatives of all functions in V. Further, since the antiderivatives of all elements of S are within V, every element of V has an antiderivative within V. And further, because constants are in V, *every* antiderivative of an element of V (aka a polynomial) is an element of V (aka a polynomial).
- Rational functions are somewhat similar: integration of rational functions relies on partial fractions theory, as discussed in the next section.

5.4. **How partial fractions help with integration.** Let's now revisit the topic of *partial fractions* as a tool for integrating rational functions. The idea behind partial fractions is to consider an integration problem with respect to a variable x with integrand of the following form:

$$\frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1}}{p(x)}$$

where p is a polynomial of degree n. For convenience, we may take p to be a monic polynomial, i.e., a polynomial with leading coefficient 1. For p fixed, the set of all rational functions of the form above forms a vector subspace of dimension n inside $\mathbb{R}(x)$. A natural choice of basis for this subspace is:

$$\frac{1}{p(x)}, \frac{x}{p(x)}, \dots, \frac{x^{n-1}}{p(x)}$$

The goal of partial fraction theory is to provide an *alternate basis* for this space of functions with the property that those basis elements are particularly easy to integrate (recurring to one of our earlier questions). Let's illustrate one special case: the case that p has n distinct real roots $\alpha_1, \alpha_2, \dots, \alpha_n$. The alternate basis in this case is:

$$\frac{1}{x-\alpha_1}, \frac{1}{x-\alpha_2}, \dots, \frac{1}{x-\alpha_n}$$

The explicit goal is to rewrite a partial fraction:

$$\frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1}}{p(x)}$$

in terms of the basis above. If we denote the numerator as $r(x)$, we want to write:

$$\frac{r(x)}{p(x)} = \frac{c_1}{x-\alpha_1} + \frac{c_2}{x-\alpha_2} + \cdots + \frac{c_n}{x-\alpha_n}$$

The explicit formula is:

$$c_i = \frac{r(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)}$$

Once we rewrite the original rational function as a linear combination of the new basis vectors, we can integrate it easily because we know the antiderivatives of each of the basis vectors. The antiderivative is thus:

$$\left(\sum_{i=1}^{n} \frac{r(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)} \ln|x - \alpha_i|\right) + C$$

where the obligatory $+C$ is put for the usual reasons.

Note that this process only handles rational functions that are proper fractions, i.e., the degree of the numerator must be less than that of the denominator. For other rational functions, we first convert the "improper fraction" to a mixed fraction form using Euclidean division, then we integrate the polynomial part in the typical way that we integrate polynomials, and integrate the remaining proper fraction as above.

We now consider cases where p is a polynomial of a different type.

Suppose p is a monic polynomial of degree n that is a product of pairwise distinct irreducible factors that are all either monic linear or monic quadratic. Call the roots for the linear polynomials $\alpha_1, \alpha_2, \dots, \alpha_s$ and call the monic quadratic factors $q_1, q_2, \dots, q_t$. We are interested in finding an alternate basis of the space of all rational functions of the form $r(x)/p(x)$ where the degree of r is less than n.

This should be familiar to you from the halcyon days of doing partial fractions. For instance, consider the example where $p(x) = (x-1)(x^2+x+1)$. In this case, the basis is:

$$\frac{1}{x-1}, \frac{1}{x^2+x+1}, \frac{2x+1}{x^2+x+1}$$

Note that an easy sanity check is that the *size* of the basis should be n. This is clear in the above example with $n = 3$, but let's reason generically.

We have that:

$$p(x) = \left[\prod_{i=1}^{s}(x - \alpha_i)\right]\left[\prod_{j=1}^{t} q_j(x)\right]$$

By degree considerations, we get that:

$$s + 2t = n$$

Now, the vector space for which we are trying to obtain a basis has dimension n. This means that the basis we are looking for should have size n, as it does, because we have a basis vector $1/(x - \alpha_i)$ for each linear factor $x - \alpha_i$ (total of s basis vectors here) and we have two basis vectors $1/q_j(x)$ and $q_j'(x)/q_j(x)$ for each quadratic factor $q_j(x)$ (total of $2t$ basis vectors here).

Now, recall that the reciprocals of the linear factors integrate to logarithms. The expressions of the form $1/q_j(x)$ integrate to an expression involving arctan. The expressions of the form $q_j'(x)/q_j(x)$ integrate to logarithms.

6. Spaces of sequences

6.1. Spaces of sequences of real numbers.

6.1. Spaces of sequences of real numbers. Recall that a sequence of real numbers is a function:

$$f : \mathbb{N} \to \mathbb{R}$$

We typically denote the function input as a subscript, so for instance we may denote $f(n)$ by a_n. The sequence is typically written by listing its elements, so the above function is written as:

$$f(1), f(2), f(3), \dots$$

or equivalently as:

$$a_1, a_2, a_3, \dots$$

We've already discussed that the space of all functions from *any* set to the reals has a natural structure of a vector space. The addition is pointwise: we add the values of the functions at each point. The scalar multiplication is also pointwise. For sequences, let's describe these operations more explicitly:

- The zero sequence is the sequence all of whose entries are zero.
- Given two sequences $a_1, a_2, a_3, \dots$ and $b_1, b_2, b_3, \dots$, the sum of the sequences is the sequence $a_1 + b_1, a_2 + b_2, a_3 + b_3, \dots$.
- Given a sequence $a_1, a_2, a_3, \dots$ and a real number λ, the sequence we get after scalar multiplication is $\lambda a_1, \lambda a_2, \lambda a_3, \dots$.

The vector space of sequences is infinite-dimensional.

6.2. Formal power series and convergence. A *formal power series* is defined as a series of the form:

$$\sum_{i=0}^{\infty} a_i x^i$$

where x is an indeterminate, and a_i are all real numbers. In other words, formal power series are like polynomials in x, except that they *can* go on indefinitely rather than being forced to terminate at a finite stage. Examples are:

$$1 + x + x^2 + \dots$$

$$1 - x + x^2 - x^3 + x^4 - \dots$$

The set of all formal power series forms a vector space under coefficient-wise addition and scalar multiplication. Note that this vector space looks a lot like (in fact, is isomorphic to) the vector space of sequences. The only difference is that we write the "vectors" as formal sums rather than as comma-separated lists.

The vector space of all formal power series is denoted $\mathbb{R}[[x]]$. Note that this vector space has a lot of additional structure to it beyond simply being a vector space.

Several statements about the nature of Taylor series and power series summation operators, which you have encountered in the past, can be framed as statements about the injectivity, surjectivity, kernel, and image of suitably defined linear transformations.

6.3. **Differentiation as an operator from formal power series to itself.** We can define a formal differentiation operator:

$$\mathbb{R}[[x]] \to \mathbb{R}[[x]]$$

that is done term-wise. Explicitly, the constant term falls off, and each $a_n x^n$ for $n \geq 1$ gets differentiated to $n a_n x^{n-1}$. In other words, the new coefficient for x^{n-1} is n times the old coefficient for x^n. Thus:

$$\frac{d}{dx}\sum_{i=0}^{\infty} a_i x^i = \sum_{i=0}^{\infty}(i+1)a_{i+1}x^i$$

Formal differentiation is a linear transformation. The kernel is the space of constant power series. These are power series where all coefficients $a_i, i \geq 1$ are equal to 0. The coefficient a_0 may be zero or nonzero. This kernel is one-dimensional, so formal differentiation is not injective.

On the other hand, formal differentiation is *surjective*. Every formal power series arises as the formal derivative of a formal power series. The "indefinite integral" is non-unique (we have flexibility in the choice of constant term, with the famed $+C$ out there). But it can be computed by term-wise integration. Explicitly:

$$\int \left(\sum_{i=0}^{\infty} a_i x^i\right) dx = \left(\sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i\right) + C$$

6.4. **Taylor series operator and power series summation operator.** The *Taylor series* operator is an operator whose domain is a space of function-like things and whose co-domain is a space of power series-like things. For our purposes, we choose a simple implementation, albeit not the best one. We view the Taylor series operator as an operator:

$$\text{Taylor series} : C^{\infty}(\mathbb{R}) \to \mathbb{R}[[x]]$$

The operator takes as input an infinitely differentiable function defined on all of $\mathbb{R}$ and outputs the Taylor series of the function centered at 0. Explicitly, the operator works as follows:

$$f \mapsto \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} x^k$$

Note that we need infinite differentiability at 0 in order to make sense of the expression. We do not really need infinite differentiability on all of $\mathbb{R}$, so the domain is in some sense too restrictive. But let's not get into the issue of trying to define new spaces to overcome this space.

For the power series summation operator, let Ω be the subspace of $\mathbb{R}[[x]]$ comprising the power series that converge globally. Then, we have a mapping:

$$\text{Summation} : \Omega \to C^{\infty}(\mathbb{R})$$

that sends a formal power series to the function to which it converges. It turns out to be true that if we start with an element of Ω, apply the summation operator to it, and then take the Taylor series of that, we land up with the same thing we started with. On the other hand, if we start with something in $C^{\infty}(\mathbb{R})$, take its Taylor series, and then try summing that up, we may land up with a completely different function, if we end up with a function at all. You should remember more of this from your study of Taylor series and power series in single variable calculus.

7. Differential operators and differential equations

7.1. **Linear differential operators with constant coefficients.** Denote by D the operator of differentiation. Note that D can be viewed as an operator in many contexts:

- It defines a linear transformation $C^1(\mathbb{R}) \to C(\mathbb{R})$
- It defines a linear transformation $C^\infty(\mathbb{R}) \to C^\infty(\mathbb{R})$
- It defines a linear transformation $\mathbb{R}[x] \to \mathbb{R}[x]$
- It defines a linear transformation $\mathbb{R}(x) \to \mathbb{R}(x)$
- It defines a linear transformation $\mathbb{R}[[x]] \to \mathbb{R}[[x]]$

We will consider the case of $D : C^\infty(\mathbb{R}) \to C^\infty(\mathbb{R})$. We will use I to denote the identity transformation $C^\infty(\mathbb{R}) \to C^\infty(\mathbb{R})$. We can then construct other linear transformations, loosely called *linear differential operators with constant coefficients*, by adding, multiplying, and scalar multiplying these. In other words, we can use polynomials in D. For instance:

- $I + D$ is the linear transformation $f \mapsto f + f'$.
- $I - D$ is the linear transformation $f \mapsto f - f'$.
- $I + 2D + 3D^2$ is the linear transformation $f \mapsto f + 2f' + 3f''$.

Finding the *kernel* of any such differential operator is equivalent to solving a homogeneous linear differential equation with constant coefficients. Finding the *fiber* over a specific nonzero function in $C^\infty(\mathbb{R})$ under any such differential operator is equivalent to solving a non-homogeneous linear differential equation with constant coefficients.

For instance, finding the kernel of $I + D$ is equivalent to solving the following linear differential equation, where x is the independent variable and y is the dependent variable. The linear differential equation is:

$$y + y' = 0$$

We know that this solves to:

$$y = Ce^{-x}, C \in \mathbb{R}$$

The kernel is thus the one-dimensional space spanned by the function e^{-x}.

On the other hand, if we are trying to find the fiber of the function, say $x \mapsto x^2$, that is equivalent to solving the non-homogeneous linear differential equation:

$$y + y' = x^2$$

A *particular solution* here is $y = x^2 - 2x + 2$. The general solution is the translate of the kernel by the particular solution, i.e., the general solution function is:

$$x^2 - 2x + 2 + Ce^{-x}, C \in \mathbb{R}$$

What this means is that each specific value of C gives a different particular solution. Pictorially, each such solution function is a point and the fiber we are thinking of is the line of all such points.

7.2. **The first-order linear differential equation: a full understanding.** We now move to the somewhat more general setting of first-order linear differential operators with nonconstant coefficients.

Consider a first-order linear differential equation with independent variable x and dependent variable y, with the equation having the form:

$$y' + p(x)y = q(x)$$

where $p, q \in C^\infty(\mathbb{R})$.

We solve this equation as follows. Let H be an antiderivative of p, so that $H'(x) = p(x)$.

$$\frac{d}{dx}\left(ye^{H(x)}\right) = q(x)e^{H(x)}$$

This gives:

$$ye^{H(x)} = \int q(x)e^{H(x)}\,dx$$

So:

$$y = e^{-H(x)} \int q(x)e^{H(x)}\,dx$$

The indefinite integration gives a $+C$, so overall, we get:

$$y = Ce^{-H(x)} + \text{particular solution}$$

It's now time to understand this in terms of linear algebra.

Define a linear transformation $L : C^\infty(\mathbb{R}) \to C^\infty(\mathbb{R})$ as:

$$f(x) \mapsto f'(x) + p(x)f(x)$$

This is a differential operator of a more complicated sort than seen earlier (explicitly, it is $L = D + pI$). The kernel of L is the solution set when $q(x) = 0$, and this just becomes the set:

$$Ce^{-H(x)}, C \in \mathbb{R}$$

In other words, each value of C gives a solution, i.e., a "point" in the kernel. The entire solution set is a line, i.e., a one-dimensional space, spanned by the function $e^{-H(x)}$.

The fiber over a function q is a translate of this kernel:

$$y = Ce^{-H(x)} + \text{particular solution}$$

Note that the fiber is non-empty because a particular solution can be obtained by integration. Thus, the linear transformation L is surjective. L is not injective because it has a one-dimensional kernel.

7.3. **Higher order differential operators and differential equations.** Here's a brief description of the theory of differential operators and differential equations that is relevant here.

Consider a linear differential equation of order n of the following form:

$$y^{(n)} + p_{n-1}(x)y^{(n-1)} + \cdots + p_1(x)y' + p_0(x)y = q(x)$$

The left side can be viewed as a linear differential operator of order n from $C^\infty(\mathbb{R})$ to $C^\infty(\mathbb{R})$. Explicitly, it is the operator:

$$L(y) = y^{(n)} + p_{n-1}(x)y^{(n-1)} + \cdots + p_1(x)y' + p_0(x)y$$

Here, all the functions $p_0, p_1, \dots, p_{n-1}, q$ are in $C^\infty(\mathbb{R})$.

The kernel of this operator is the solution set to the corresponding homogeneous linear differential equation:

$$y^{(n)} + p_{n-1}(x)y^{(n-1)} + \cdots + p_1(x)y' + p_0(x)y = 0$$

It turns out that we generically expect this kernel to be a n-dimensional subspace of $C^\infty(\mathbb{R})$. Explicitly, this means that the kernel has a basis comprising n functions, each of which is a solution to the system. This is consistent with the general principle that we expect n independent parameters in the general solution to a differential equation of order n.

The general solution to the original non-homogeneous linear differential equation, if it exists, is of the form:

(Particular solution) + (Arbitrary element of the kernel)

The elements of the kernel are termed "auxilliary solutions" so we can rewrite this as:

(General solution) = (Particular solution) + ((General) auxilliary solution)

The parameters (freely floating constants) all come from the choice of arbitrary element of the kernel. There are n of them, as expected. Unfortunately, unlike the first-order case, there is no generic integration formula for finding the particular solution. The first-order and second-order cases are the only cases where a generic integration formula is known for finding the auxilliary solutions. Also, it is known how to find the auxilliary solutions in the constant coefficients case for arbitrary order.

The linear differential operator L is thus *surjective but not injective*: it has n-dimensional kernel.

ORDINARY LEAST SQUARES REGRESSION

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: For the computational process, Section 5.3 of the book is relevant. For comparing notation:

- X in the notes corresponds to A in the book
- $\vec{\beta}$ in the notes corresponds to $\vec{x}$ in the book
- $\vec{y}$ in the notes corresponds to $\vec{b}$ in the book

Executive summary

Words ...

(1) Consider a model where the general functional form is linear in the parameters. Input-output pairs give a system of simultaneous linear equations in terms of the parameters. Each row of the coefficient matrix corresponds to a particular choice of input, and each corresponding entry of the augmenting column is the corresponding output. In the "no-error" case, what we would ideally like is that the coefficient matrix has full column rank (i.e., we get unique solutions for the parameters assuming consistency) and does *not* have full row rank (i.e., we have some extra input-output pairs so that consistency can be used as evidence in favor of the hypothesis that the given functional form is correct). If the model is correct, the system will be consistent (despite potential for inconsistency) and we will be able to deduce the values of the parameters.

(2) Once we introduce measurement error, we can no longer find the parameters with certainty. However, what we *can* hope for is to provide a "best guess" for the parameter values based on the given data points (input-output pairs).

(3) In the case where we have measurement error, we still aim to choose a large number of inputs so that the coefficient matrix has full column rank but does not have full row rank. Now, however, even if the model is correct, the system will probably be inconsistent. What we need to do is to replace the existin output vector (i.e., the existing augmenting column) by the vector closest to it that is in the image of the linear transformation given by the coefficient matrix. Explicitly, if $\vec{\beta}$ is the parameter vector that we are trying to find, X is the coefficient matrix (also called the design matrix), and $\vec{y}$ is the output vector, the system $X\vec{\beta} = \vec{y}$ may not be consistent. We try to find a vector $\vec{\varepsilon}$ of minimum length subject to the constraint that $\vec{y} - \vec{\varepsilon}$ is in the image of the linear transformation given by X, so that the system $X\vec{\beta} = \vec{y} - \vec{\varepsilon}$ is consistent. Because in our setup (if we chose it well), X had full column rank, this gives the unique "best"choice of $\vec{\beta}$. Note also that "length" here refers to Euclidean length (square root of sum of squares of coordinates) when we are doing an *ordinary least squares regression* (the default type of regression) but we could use alternative notions of length in other types of regressions.

(4) In the particular case that the system $X\vec{\beta} = \vec{y}$ is consistent, we choose $\vec{\varepsilon} = \vec{0}$. However, this does not mean that ths is the actual correct parameter vector. It is still only a guess.

(5) In general, the more data points (i.e., input-output pairs) that we have, the better our guess becomes. However, this is true only in a probabilistic sense. It may well happen that a particular data point worsens our guess because that data point has a larger error than the others.

(6) The corresponding geometric operation to finding the vector $\vec{\varepsilon}$ is orthogonal projection. Explicitly, the image of X is a subspace of the vector space $\mathbb{R}^n$ (where n is the number of input-output pairs). If there are m parameters (i.e., $\vec{\beta} \in \mathbb{R}^m$). and we chose X wisely, the image of X would be a m-dimensional subspace of $\mathbb{R}^n$. In the no-error case, the vector $\vec{y}$ would be in this subspace, and we would be able to find $\vec{\beta}$ uniquely and correctly. In the presence of error, $\vec{y}$ may be outside

the subspace. The vector $\vec{y} - \vec{\varepsilon}$ that we are looking for is the orthogonal projection of $\vec{y}$ onto this subspace. The error vector $\vec{\varepsilon}$ is the component of $\vec{y}$ that is perpendicular to the subspace.

(7) A fit is impressive in the case that m is much smaller than n and yet the error vector $\vec{\varepsilon}$ is small. This is philosophically for the same reason that consistency becomes more impressive the greater the excess of input-output pairs over parameters. Now, the rigid notion of consistency has been replaced by the more loose notion of "small error vector."

Actions ...

(1) Solving $X\vec{\beta} = \vec{y} - \vec{\varepsilon}$ with $\vec{\varepsilon}$ (unknown) as the vector of minimum possible length is equivalent to solving $X^TX\vec{\beta} = X^T\vec{y}$. Note that this process does not involve finding the error vector $\vec{\varepsilon}$ directly. The matrix X^TX is a square matrix that is symmetric.
(2) In the case that X has full column rank (i.e., we have unique solutions *if* consistent), X^TX also has full rank (both full row rank and full column rank – it is a square matrix), and we get a unique "best fit" solution.

1. The simple case of fitting a linear function

1.1. **Recalling the no-error case.** Suppose f is a function of one variable x, defined for all reals. Recall that f is called a *linear* function (in the affine linear, rather than the homogeneous linear, sense) if there exist constants m and c such that $f(x) = mx + c$. Suppose the only thing you have access to is a black box that will take an input x and return the value $f(x)$. You have no information about the inner working of the program.

You do not know whether f is a linear function, but you believe that that may be the case.

(1) Suppose you are given the value of f at one point. This is useful information, but does not, in and of itself, shed any light on whether f is linear.
(2) Suppose you are given the values of f at two points. This would be enough to determine f uniquely under the assumption that it is linear. However, it will not be enough to provide evidence in favor of the hypothesis that f is linear. Why? Because *whatever* the values of the function, we can always fit a straight line through them.
(3) Suppose you are given the values of f at three points. This allows for a serious test of the hypothesis of linearity. There are two possibilities regarding how the three points on the graph of f look:
    - The points on the graph are non-collinear: This is *conclusive* evidence *against* the hypothesis of linearity. We can definitely *reject* the hypothesis that the function is linear.
    - The points on the graph are collinear: This is evidence *in favor of* the hypothesis of linearity, but it is not conclusive. We can imagine a function that is not linear but such that the outputs for the three specific inputs that we chose happened to fall in a straight line. As we discussed earlier, there are good reasons to treat the collinearity of points on the graph as *strong evidence* in favor of linearity. But it is not conclusive. For instance, for the function $f(x) = x^3 + x$, the points $x = -1$, $x = 0$, and $x = 1$ appear to satisfy the collinearity condition, despite the function not being linear.
(4) Suppose you are given more than three points and the values of f at all these points. This allows for an even stronger test of the hypothesis of linearity. There are two possibilities regarding how the corresponding points on the graph of the function look:
    - The points on the graph are not all collinear: This is *conclusive* evidence *against* the hypothesis of linearity. We can definitely *reject* the hypothesis that the function is linear.
    - The points on the graph are collinear: This is evidence *in favor of* the hypothesis of linearity, but is not conclusive. After all, we can draw curves that are not straight lines to fill in the unknown gaps between the known points on the graph. However, it does constitute strong evidence in favor of the linearity hypothesis. And the more points we have evaluated the function at, the stronger the evidence.

Although the discussion that follows is general, we will use linear fitting as our motivating example.

We will use the term *data point* or *input-output pair* for a pair of values (x, y) with y the (actual or somewhat erroneous) value of $f(x)$. We will use the terms *line-fitting* (more generally, *curve-fitting*) and *parameter estimation.*

1.2. **The introduction of error.** Broadly speaking, there are two kinds of errors that introduce minor inaccuracies: *modeling errors (arising from the model being only an approximate representation of reality)* and *measurement errors*. Modeling errors refers to the fact that the linear model may be only an approximation to reality, albeit still a useful approximation. Measurement errors refers to the fact that the black box we use to measure the fuction values may not measure the values accurately. Note: There's another kind of modeling error, namely uncertainty about whether the model is right at all. We'll discuss that type of error later. The modeling errors we are talking about right now are errors where the model is a reasonable approximation of reality but not exactly correct.

In the physical science world, there are situations where modeling errors are zero (or near-zero): the world actually does conform to the actual model. The only errors we need to confront here are measurement errors. Note that measurement errors confound the process of *finding* the parameter values using input-output pairs. Once the parameters are found, though, the model can be applied perfectly.

In the social science world, both measurement and modeling errors occur. In fact, in some sense, it is hard to conceptually separate measurement errors from modeling errors. That's because in the social sciences, it is harder to pinpoint what the "true value" of something means. Consider, for instance, a construct of "happiness" that can be measured through surveys. A theoretical model might attempt to predict happiness in terms of socio-economic status, income, race, education levels, physiological measures (heart rate etc.) and other measures. The predicted values from the model may differ from the survey-measured happiness. One could argue that this difference arises from measurement error: the survey isn't measuring true happiness. One could also argue that there is modeling error: the model isn't calculating happiness correctly. Or, one could argue that it's a mix: the model is not correct, but there are also measurement errors. But a deeper critique would be to argue against the legitimacy of happiness as a construct itself, i.e., to argue against the idea that there is a "true value" of happiness that our model and measurement tools are getting at.

Explicitly, we can think of the relationship as follows:

Value predicted by model $\overset{\text{modeling error}}{\leftrightarrow}$ "True" value $\overset{\text{measurement error}}{\leftrightarrow}$ Measured value

The deeper critique would say that there is no true value we are getting at, and that the relationship is something of the form:

Value predicted by model $\overset{\text{all error}}{\leftrightarrow}$ Measured value

For our present purposes, the validity of the deeper critique is not relevant. All we are trying to do is use our measured value to find the parameters for the model (for instance, in the linear model case, we are using input-output pairs to find the values of m and c).

Thus, for our present purposes, *we will use the term measurement error for the total of measurement error and errors arising from the model being somewhat inaccurate.*

1.3. **The middle case for error is what's interesting.** There are three possibilities:

- The measurement and modeling errors are zero, or so close to zero, that for practical purposes they don't affect us at all.
- The measurement and modeling errors are so huge that the model is practically useless. For instance, huge measurement errors means that finding the parameters in the model from input-output pair observations is difficult. Huge modeling errors means that the model isn't that predictive of actual values anyway (over and above the problem of finding the parameters).
- The measurement and modeling errors are big enough that they do confound the process of determining the parameters and predicting outputs from inputs. They are not, however, big enough to make the model useless. So, it's hard enough to be challenging, but not hard enough to be useless.

1.4. **The core idea of regression.** Let's concentrate on the linear model $f(x) = mx + c$. In the absence of modeling and measurement errors, two input-output pairs suffice to determine f uniquely. Three or more input-output pairs can be used to obtain additional confirmation of the model, but they do not actually help with *finding* the parameters. In this sense, once we have the input-output values at two points, then additional input-output data is redundant.

In the *presence* of errors, just using two points is not good enough, because the values at those two points need not be the correct values. Sure, we *can* get a line through those two points, and that line would probably be somewhat related to the actual line we are trying to get. But once we introduce error, then we can use more input-output pairs to *improve* the accuracy of our guesses for m and c. If we're not absolutely sure, more data is (in a probabilistic sense) helpful. It's *possible* that the new data could reduce the accuracy of our estimate, but in general we expect our estimate for the line to improve as we have more points.

Let's think about this a bit more. Suppose we are given three input-output pairs (x_1, y_1), (x_2, y_2), (x_3, y_3). We plot these three points on the xy-plane. In the absence of errors, these three points would be collinear, and we could join them and get the line that describes the function. In the presence of errors, however, it is quite likely that the points will not be collinear. Further, even if they happen to be collinear, we cannot be completely sure that the line of these three points is the correct line. Perhaps the measurement errors at the points caused the values to get distorted in a collinear fashion.

So, finding the *actual* line for sure is not possible, but we can still ask a question such as: "what is the *best* line that fits the three points as closely as possible?" That's not necessarily the same as the actual line, but it's the closest fit we can find. In other words, what is the line such that the total distance between the observed points and the actual line is as little as possible? If that question sounds vague, that's a feature, not a bug. There are many ways of trying to make it precise, and we'll be getting into those in the next section. But you should first be on board with the *goal* that we have here: trying to find a line such that the observed points come as close to the line as possible.

What if we have more than three points? The idea above continues to work. As we add more new points, though, the best line fit may keep changing to incorporate the new data. In general, we expect the best line fit to come closer and closer to the *true* line. The process is probabilistic, so it may well happen that in some cases, adding new data leads to a somewhat worse fit. It's just that, overall, we expect the fit to get better and better as we add more and more data.

The description above seems wishy-washy and more like a test of artistic and visual skill than a procedure that can be handled mathematically. And, it's not clear where linear algebra comes in. Let's remedy that.

2. Getting linear algebra into the picture

2.1. **The no-error case with just enough information.**

As before, we're trying to find a function f of one variable that is known to be of the form $f(x) = mx + c$ where m and c are real numbers. Let's write $y = f(x)$. It suffices to know the values (x_1, y_1) and (x_2, y_2) for $x_1 \neq x_2$. We could then use these input-output pairs to find the values of m and c. Explicitly, we'd set up the following linear system:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

The coefficient matrix is:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}$$

If $x_1 \neq x_2$, the coefficient matrix has full rank. It's a square matrix, so this means full row rank and full column rank. Let's review what both these mean:

- Full *row* rank means that the system can always be solved for the parameters m and c. In particular, if there is any doubt about the model, just using two points cannot be used to test the validity of the model.
- Full *column* rank means that the solution, if it exists, is unique, i.e., we find a unique m and c satisfying the system.

Of course, in this case, we already knew this geometrically: given any two points, there is a unique line joining them. There is also an information-theoretic interpretation: there are two unknown parameters m and c, so we need two pieces of information to pin them down, and that information can be given in the form of two input-output pairs.

Let's be clear what is going on:

- Input values of input-output pairs $\leftrightarrow$ Rows of coefficient matrix
- Output values of input-output pairs $\leftrightarrow$ Entries of output vector, aka augmenting column

- Parameters for the line (namely, the slope and the intercept) $\leftrightarrow$ *coordinates* of the *parameter vector* $\begin{bmatrix} c \\ m \end{bmatrix}$

2.2. **The no-error case with extra information.** Continuing with the same setup as above, suppose that instead of just two input-output pairs, we are given n input-output pairs (x_1, y_1), (x_2, y_2), $\dots$, (x_n, y_n). We need to find the values of the parameters m and c. The linear system we set up is:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

We assume that $x_1, x_2, \dots, x_n$ are all distinct points. The coefficient matrix is:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix}$$

Note that although all the rows are distinct, the rank is still 2, since it is constrained by the number of columns. Thus, the coefficient matrix has full column rank but *not* full row rank. The significance is as follows:

- The fact that it has full column rank means that if a solution exists, then it is unique. This corresponds to the geometric fact that if the points are collinear, there is a unique line through them.
- The fact that it does *not* have full row rank means that a solution need not exist. What this means is that if our model is incorrect, it is potentially falsifiable using the model. This connects to our earlier observation that using more than two points gives us the opportunity of falsifying the model. If the system is consistent, that is evidence in favor of the hypothesis of linearity.

2.3. **The error case with just two points.** Return to the case of the linear model with just two input-output pairs (x_1, y_1) and (x_2, y_2) given to us. However, there is now measurement and/or modeling error. Thus, we have:

$$\begin{aligned} y_1 &= f(x_1) + \varepsilon_1 \\ y_2 &= f(x_2) + \varepsilon_2 \end{aligned}$$

where ε_1 and ε_2 are the error terms. Or equivalently:

$$\begin{aligned} f(x_1) &= y_1 - \varepsilon_1 \\ f(x_2) &= y_2 - \varepsilon_2 \end{aligned}$$

The problem, of course, is that we don't *know* what ε_1 and ε_2 are. We can now formulate the linear system:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}$$

A reasonable assumption is that the errors are symmetrically distributed about zero, and the median and modal value of each error is zero. In this case, our "best fit" comes from solving:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

We can now determine a unique best fit line. This is simply the line joining the two points (x_1, y_1) and (x_2, y_2). Note that this is not *the* line for f: it's simply our best guess given the available information.

2.4. **More than two data points, with error.** Suppose now that we have more than two points. Due to the presence of error, new points *do* add value. As before, we have points (x_1, y_1), (x_2, y_2), $\dots$, (x_n, y_n). The "true value" $f(x_i)$ equals $y_i - \varepsilon_i$ where ε_i is the (unknown) error term.

The linear system is:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} - \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_n \end{bmatrix}$$

Let's say the points actually happen to be collinear. In that case, the best fit line is obvious: it is the line that passes through the points (x_i, y_i). What that means in the context of this system is that the system:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

i.e., the system assuming no error, is consistent. Note that full column rank means that if a solution exists, it is unique, which in this case translates to saying that the no-error solution is a unique line, i.e., that the values of m and c are uniquely determined. Note that these still need not be the actual values. But they are the best guess we can make given the information.

In the domain-range language, the case where the points are all collinear corresponds to the case where the observed output vector is in the range of the linear transformation corresponding to the coefficient matrix.

What, now, if the system:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

is inconsistent? This means that we cannot get a line that fits all the data points. What we'd ideally like is a line that comes close to fitting the data points. In other words, we are trying to find an error vector $\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_n \end{bmatrix}$ such that the system below is consistent:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_n \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} - \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_n \end{bmatrix}$$

In other words, we want an error vector such that the difference:

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} - \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

is in the range (image) of the linear transformation given by the coefficient matrix.

Now, let's be clear: the choice of error vector is far from unique. An obvious choice that would work is to take the error vector $\vec{\varepsilon}$ to equal $\vec{y}$, so that the difference is the zero vector, giving $m = c = 0$. One could also imagine many other choices that work (this will become clearer in the subsequent discussion). Given lots of possible error vectors we can subtract, what vector should we choose?

The idea of the *best fit* is that we want to choose the error vector to be *as small as possible.* Small in what sense? That depends on the type of regression we are interested in doing. The most typical type of regression (both due to its computational ease and its conceptual significance) is called *ordinary least squares* regression: we try to find a candidate error vector of minimum possible length (in the sense of the square root of the sum of squares of the coordinates) so that subtracting it off gives us an output vector that is in the image.

2.5. **Taking stock: closest vector in the range.** Let's give names to the vectors. Suppose:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}, \vec{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_n \end{bmatrix}$$

The coefficient matrix can be written as:

$$X = \begin{bmatrix} \vec{1}_n & \vec{x} \end{bmatrix}$$

where $\vec{1}_n$ is used as shorthand for the all ones vector with n coordinates.

Define a new vector:

$$\vec{z} = \vec{y} - \vec{\varepsilon}$$

The vector $\vec{z}$ can be thought of as our *best estimate* for the "true value" of the output vector. $\vec{z}$ is the vector we want in the image of the linear transformation given by the coefficient matrix. Pictorially, $z_1, z_2, \dots, z_n$ are such that the points (x_1, z_1), (x_2, z_2), $\dots$, (x_n, z_n) are collinear, and the line we use to fit these points gives us our best guesses for m and c.

Note that finding the true vector $\vec{z}$ is equivalent to finding the vector $\vec{\varepsilon}$ once $\vec{x}$ and $\vec{y}$ are known. We can reformulate our goal as follows:

> Given the vectors $\vec{x}$ and $\vec{y}$, find the vector $\vec{z}$ in the image of the linear transformation given by the matrix $\begin{bmatrix} \vec{1}_n & \vec{x} \end{bmatrix}$ that is closest to the vector $\vec{y}$.

The matrix:

$$X = \begin{bmatrix} \vec{1}_n & \vec{x} \end{bmatrix}$$

defines a linear transformation $\mathbb{R}^2 \to \mathbb{R}^n$, so its image is a subspace of $\mathbb{R}^n$. Since the matrix has rank 2, the image is a two-dimensional subspace of $\mathbb{R}^n$, i.e., it is a plane in $\mathbb{R}^n$. Explicitly, it is a plane through the origin with basis given by the vectors:

$$\begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

The first basis vector corresponds to the outputs that we would get for the line $y = 1$ (i.e., the case $c = 1, m = 0$). The second basis vector corresponds to the outputs that we would get for the line $y = x$ (i.e., the case $c = 0, m = 1$).

The case $r = 3$ may be particularly easy to visualize: the image of the linear transformation (the set of possible outputs that could actually arise from lines) is a plane in $\mathbb{R}^3$, spanned by the vectors:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

The first basis vector corresponds to the outputs that we would get for the line $y = 1$ (i.e., the case $c = 1, m = 0$). The second basis vector corresponds to the outputs that we would get for the line $y = x$ (i.e., the case $c = 0, m = 1$). Various linear combinations of these possibilities correspond to various values of m and c.

The actual observed output $\vec{y}$ is a vector in $\mathbb{R}^3$, but it is not necessarily in the plane. We need to find the vector $\vec{z}$ in the plane that is "closest" to $\vec{y}$.

Pictorially, how would one find the closest vector in a plane to a vector not in the plane? We know the answer: orthogonal projection. Drop a perpendicular. That's the fundamental idea that generalizes, even to cases where we are operating in a generic $\mathbb{R}^n$ instead of $\mathbb{R}^3$.

Let's make the correspondence clear:

- Set of all vectors $\begin{bmatrix} c \\ m \end{bmatrix}$ $\leftrightarrow$ Set of all possible linear functions (we are trying to pick the best fit from this set of possibilities)
- Image of the linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^n$ $\leftrightarrow$ Set of all possible outputs for the given inputs for which the data points are collinear.
- Observed vector $\vec{y}$ is in the image plane $\leftrightarrow$ the data points are already collinear, so the "best fit" is obvious (even though it still may not be the correct fit)
- Observed output vector $\vec{y}$ is not in the image plane $\leftrightarrow$ the data points are not already collinear.
- Closest vector $\vec{z}$ in the image plane to the observed output vector $\vec{y}$ $\leftrightarrow$ the best line fit for the data points.

2.6. **Regressions, summarized.** Here is a reasonable high-level summary of what we have done:

- The original "no-error case" goal involved solving a linear system with a given coefficient matrix and a given output vector. The absence of error guaranteed that the output vector would indeed be in the image of the coefficient matrix.
- The "with-error case" has a coefficient matrix and an output vector. The output vector is no longer necessarily in the range of the coefficient matrix. We replace the output vector by the closest vector to it that is in the image of the coefficient matrix. The vector that we subtract in the process is our guess for the error vector. Another way of putting it is that we are trying to choose the error vector to be as small as possible. Here, we define "small" as meaning that the length (the square root of the sum of squares of coordinates) is as small as possible.
- Finding the closest vector geometrically means "dropping a perpendicular" which can be operationalized using matrix multiplications (we will not go into the computational details here).

3. Ordinary least squares regression: the general setting

3.1. **Linearity in parameters is what is crucial.** The example we discussed above involves the case of a linear function of one variable (in the affine linear sense, so that there may be a nonzero intercept).

Describing such a function required two parameters, one a slope parameter m and one an intercept parameter c. Moreover, the example was also linear in the parameters. Explicitly, it has the form:

$$f(x) = (1)c + (x)m$$

Thus, a particular input x_i with output y_i gives an equation of the form:

$$1c + x_i m = y_i$$

The corresponding row of the augmented matrix is thus:

$$\left[\begin{array}{cc|c} 1 & x_i & y_i \end{array}\right]$$

The corresponding row of the coefficient matrix is:

$$\begin{bmatrix} 1 & x_i \end{bmatrix}$$

Now, we could consider some more complicated examples. For instance, consider a model with a functional form:

$$f(x) = A\cos x + B\sin x$$

The function f described here is not a linear function at all. However, it is linear in the parameters A and B, and those are what we are trying to determine using input-output pairs. Each input-output pair thus gives a linear equation. Explicitly, each piece of information of the form $f(x_i) = y_i$ gives rise to a linear equation of the form:

$$(\cos x_i)A + (\sin x_i)B = y_i$$

The corresponding row of the augmented matrix is:

$$\left[\begin{array}{cc|c} \cos x_i & \sin x_i & y_i \end{array}\right]$$

The corresponding row of the coefficient matrix is:

$$\begin{bmatrix} \cos x_i & \sin x_i \end{bmatrix}$$

We can thus apply the linear regression methods to this situation. However, this is a relatively unimportant case, so we will turn instead to another functional form that is linear in the parameters and is extremely important: the *polynomial*. But first, let's formulate the general picture.

3.2. **The full general setting: functions of one variable with multiple parameters.** Consider a function of one variable of the form:

$$f(x) = \beta_1 g_1(x) + \beta_2 g_2(x) + \cdots + \beta_m g_m(x)$$

where the functions $g_1, g_2, \ldots, g_m$ are all known functions, and the coefficients $\beta_1, \beta_2, \ldots, \beta_m$ are the parameters of the model. These are the parameters that we are trying to determine.

Each input-output pair (x_i, y_i) gives rise to an equation of the form:

$$\beta_1 g_1(x_i) + \beta_2 g_2(x_i) + \cdots + \beta_m g_m(x_i) = y_i$$

The corresponding row of the augmented matrix is:

$$\left[\begin{array}{cccc|c} g_1(x_i) & g_2(x_i) & \ldots & g_m(x_i) & y_i \end{array}\right]$$

The corresponding row of the coefficient matrix is:

$$\begin{bmatrix} g_1(x_i) & g_2(x_i) & \ldots & g_m(x_i) \end{bmatrix}$$

Suppose we are given a collection of input-output pairs (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n). The linear system that we want to solve is:

$$\begin{bmatrix} g_1(x_1) & g_2(x_1) & \dots & g_m(x_1) \\ g_1(x_2) & g_2(x_2) & \dots & g_m(x_2) \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ g_1(x_n) & g_2(x_n) & \dots & g_m(x_n) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdot \\ \cdot \\ \beta_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

Let's use vector notation:

$$\vec{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdot \\ \cdot \\ \beta_m \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}, X = \begin{bmatrix} g_1(x_1) & g_2(x_1) & \dots & g_m(x_1) \\ g_1(x_2) & g_2(x_2) & \dots & g_m(x_2) \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ g_1(x_n) & g_2(x_n) & \dots & g_m(x_n) \end{bmatrix}$$

We are essentially trying to solve the linear system:

$$X\vec{\beta} = \vec{y}$$

where the matrix X is determined completely by the *input* parts of the input-output pairs, and the vector $\vec{y}$ is the outputs.

If we were trying to solve this directly, we would try to make sure that the matrix X has full column rank m, i.e., we have enough input-output pairs to be able to uniquely determine all the parameters. If it does, then that's good enough. Obviously, a necessary condition for full column rank is that $n \geq m$. If the inputs are chosen wisely, this is also sufficient (to be more specific, we'd need to know the functions g_i).

Once we introduce error, though, the model changes. We're actually trying to solve the system:

$$X\vec{\beta} = \vec{z}$$

where $\vec{z}$ is the closest vector to $\vec{y}$ in the range of X. In other words:

- We first perform the orthogonal projection of $\vec{y}$ on the subspace of $\mathbb{R}^n$ that is the image of X. Note that this is a m-dimensional subspace of $\mathbb{R}^n$.
- If $\vec{z}$ is this orthogonal projection, we then solve the equation $X\vec{\beta} = \vec{z}$. Note that full column rank guarantees a unique solution.
- This solution is our best estimate for the parameter values.

3.3. **Functions of more than one variable.** We could have the same setup for a functional form that uses more than one variable:

$$f(x_1, x_2, \dots, x_r) = \beta_1 g_1(x_1, x_2, \dots, x_r) + \beta_2 g_2(x_1, x_2, \dots, x_r) + \dots + \beta_m g_m(x_1, x_2, \dots, x_r)$$

Everything in the description proceeds as with the single variable case. In some sense, the original input variables do not matter. What gets "seen" as far as linear regression is concerned is the coefficient matrix. Note that we now have to use *double subscripts*, one subscript to refer to which input we are talking about, and one subscript to refer to the coordinate of the input that we are referring to. Explicitly, the input-output pair information is:

$$\begin{aligned} f(x_{1,1}, x_{1,2}, \dots, x_{1,r}) &= y_1 \\ f(x_{2,1}, x_{2,2}, \dots, x_{2,r}) &= y_2 \\ \cdot &= \cdot \\ \cdot &= \cdot \\ \cdot &= \cdot \\ f(x_{n,1}, x_{n,2}, \dots, x_{n,r}) &= y_n \end{aligned}$$

The design matrix becomes:

$$X = \begin{bmatrix} g_1(x_{1,1}, x_{1,2}, \dots, x_{1,r}) & g_2(x_{1,1}, x_{1,2}, \dots, x_{1,r}) & \dots & g_m(x_{1,1}, x_{1,2}, \dots, x_{1,r}) \\ g_1(x_{2,1}, x_{2,2}, \dots, x_{2,r}) & g_2(x_{2,1}, x_{2,2}, \dots, x_{2,r}) & \dots & g_m(x_{2,1}, x_{2,2}, \dots, x_{2,r}) \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ g_1(x_{n,1}, x_{n,2}, \dots, x_{n,r}) & g_2(x_{n,1}, x_{n,2}, \dots, x_{n,r}) & \dots & g_m(x_{n,1}, x_{n,2}, \dots, x_{n,r}) \end{bmatrix}$$

3.4. **Terminology used.** We use the following terminology:

- The vector $\vec{\beta}$ is termed the *parameter vector*. Its coordinates β_i are termed the *parameters*, *effects*, or *regression coefficients*.
- The coefficient matrix X is termed the *design matrix*. The entries of the matrix are sometimes termed the *regressors* (there are some complications here worth a separate discussion, that we cannot afford right now).
- The vector $\vec{\varepsilon}$ is termed the *error term*, *disturbance*, or *noise*.

4. Polynomial regression: a special case of linear regression

4.1. **Wait, why are polynomials a special case of linear things?** An obvious point of confusion is how *polynomial* regression can be a special case of *linear* regression. After all, polynomial functions are *more* general than linear functions.

Answer: the *polynomial* refers to the nature of the function in terms of its input. The term *linear* refers to the nature of the function in terms of its parameters. Specifically, polynomial regression is used for situations where our model predicts a polynomial of bounded degree with unknown coefficients, and the goal of regression is to determine these coefficients.

4.2. **The setup.** Consider the general case of a polynomial of degree $\leq m$. Note that the number of arbitrary coefficients here is $m+1$. We will number the subscripts starting from 0, so the notation will differ somewhat from earlier. Explicitly, the general functional form is:

$$f(x) := \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m$$

Suppose we are given n distinct input-output pairs (x_i, y_i). The coefficient matrix X for the linear system (called the design matrix in linear regression terminology) has the form:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix}$$

X is a $n \times (m+1)$ matrix.

The vector $\vec{\beta}$ has $m+1$ coordinates and is:

$$\vec{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \cdot \\ \cdot \\ \cdot \\ \beta_m \end{bmatrix}$$

In the no-error case, we are trying to solve the equation:

$$X\vec{\beta} = \vec{y}$$

In the case with an error vector $\vec{\varepsilon}$, we are trying to solve the equation:

$$X\vec{\beta} = \vec{y} - \vec{\varepsilon}$$

where we are trying to keep $\vec{\varepsilon}$ as small as possible.

The matrix X defines a linear transformation $\mathbb{R}^{m+1} \to \mathbb{R}^n$.

The matrix X is called the *Vandermonde matrix* for the given collection of inputs x_i. Note that X is determined by the inputs alone, and is independent of the outputs.

4.3. **The no-error case.** In the case that there is no error, we are trying to solve:

$$X\vec{\beta} = \vec{y}$$

It turns out from some easy-to-verify-but-hard-to-think-of facts in polynomial theory that if all the values x_i are distinct, then the rank of X is $\min\{m+1, n\}$. In particular, the bare minimum number of inputs that we need is $m+1$: with $m+1$ distinct inputs, we get a square matrix X that has full rank. We can thus uniquely recover $\vec{\beta}$ from $\vec{y}$. Note, however, that $m+1$ inputs do not allow for independent confirmation of the hypothesis that f is a polynomial of degree $\leq m$, because any $m+1$ input-output pairs will fit a polynomial. To obtain independent confirmation, we need $m+2$ or more input-output pairs. Note that if we take $m+2$ or more input-output pairs, the coefficient matrix has full column rank $m+1$, so we still get a unique solution. However, since the row rank is not full, the possibility of inconsistency has now been opened up. If the system is still consistent, that is independence in favor of the hypothesis that f is indeed a polynomial of degree $\leq m$.

4.4. **The case of error.** The goal is to solve the system:

$$X\vec{\beta} = \vec{y} - \vec{\varepsilon}$$

where we are trying to keep $\vec{\varepsilon}$ as small as possible. We consider the image of X, and attempt to find the vector in that image that is closest to $\vec{y}$. In other words, we determine the orthogonal projection of $\vec{y}$ on the image of X. If we call that vector $\vec{z}$, we must then solve the equation:

$$X\vec{\beta} = \vec{z}$$

Note also that $\vec{\varepsilon} = \vec{y} - \vec{z}$.

Geometrically, the image of X is a $(m+1)$-dimensional subspace of $\mathbb{R}^n$.

5. Against overfitting

5.1. **Combining measurement error with uncertainty about the model.** We consider overfitting in connection with two related concerns: *measurement error* and *the fear that we have picked a completely wrong model.* We have dealt with both issues separately. It's now time to connect the dots.

- We have dealt with the issue of measurement error using the idea of linear regression: we collect a lot more points than the bare minimum needed to ascertain the values of the parameters, and then use ordinary least squares regression to determine the parameter values. Note that we need a lot more data points than the actual number of parameters once we are combating measurement errors.
- We have dealt with the concern that the model itself is completely wrong in the hypothesis testing discussion: The idea is to pick one or more input over and above the minimum number of inputs needed to determine parameters, and see if the system is still consistent. For instance, for the original example of a linear function of one variable with possibly nonzero intercept, we just need two points to find the line, but we need three or more points to obtain evidence in favor of the hypothesis of linearity.

Note that we are solving two very different types of problems with roughly the same solution. The problem of uncertainty about whether the model is correct is solved by using extra data points, but subject to a very stringent test: we demand that the extra data points keep the system consistent. The problem of measurement error is solved by using extra data points, but mostly just to obtain a better estimates.

What if both problems occur together? In other words, we need to contend with measurement error and with the problem that our model may be just wrong. In the presence of both measurement error and uncertainty, we need not just a few more data points, we need a *lot more* data points. Instead of a stringent test of fitting the line or curve, we impose a requirement that the total error in the fit is small. By "total error" here we mean the length of the error vector $\vec{\varepsilon}$. There is an alternative related way of measuring goodness of fit called R^2 (also called the *coefficient of determination*).

5.2. **Trying to fit a generic polynomial.** The preceding discussion of polynomial regression assumed that we are trying to fit using a polynomial with a fixed bound on its degree. Let's consider a somewhat more generic setting.

We are given a function f and n input-output pairs for f. We have some reason to believe that f may be polynomial.

If we didn't have measurement error, the process would be simple: try fitting f with a constant polynomial, a polynomial of degree ≤ 1, a polynomial of degree ≤ 2, and so on, and continue until we get a perfect fit. If we manage to get a perfect fit using a polynomial of degree m where $m + 1$ is less than n (the number of data points we have), that is strong evidence in favor of the hypothesis that f is polynomial, because the remaining $n - (m + 1)$ outputs agreeing with the predictions on sheer chance would be unlikely. If we think in terms of solving linear systems, this is equivalent to requiring that the appropriate augmented entries just happen to be 0 even though the entries were chosen randomly. The larger the gap $n - (m + 1)$, the stronger the evidence in favor of the hypothesis. If, on the other hand, we only achieve perfect fit at $n = m + 1$, that's no evidence in favor of the polynomial hypothesis.

Suppose we do have measurement error. Let's say we have 1000 data points. In that case, we can always fit a polynomial of degree 999 perfectly without measurement error. But that's cheating. What would be more convincing would be a polynomial of small degree (such as 3) that, even if not a perfect fit, gives a low error. The idea, therefore, would be to start with constant polynomials, polynomials of degree ≤ 1, polynomials of degree ≤ 2, polynomials of degree ≤ 3. Each time we increase the degree of the polynomial, we add in a new parameter, allowing for more wiggle room to fit the data points. So, our fit keeps getting better and better. But, *it's not clear whether the improved fit is fitting the signal or the noise.*

The general idea in this case is to see each time we allow one more parameter whether that additional parameter adds significant explanatory value, i.e., whether it pushes down the measurement error considerably. In general, the fewer parameters we have (which in this case means "the smaller the degree of the polynomial we are using") the more impressive a good fit is. This is also related to Occam's Razor.

5.3. **Interpretation in terms of the innards of linear regression.** Let's consider the above in terms of how linear regression is actually executed. As above, consider that we have 1000 input-output pairs for f. The output vector for f is a vector in $\mathbb{R}^{1000}$.

Trying to fit this with a constant function involves finding the orthogonal projection on a one-dimensional subspace of $\mathbb{R}^{1000}$. Note that the specific description of the one-dimensional subspace actually depends on the input values.

Trying to fit this with a linear function involves finding the orthogonal projection on a two-dimensional subspace containing that one-dimensional subspace. Trying to fit with a polynomial of degree ≤ 2 involves finding the orthogonal projection on a three-dimensional subspace that contains the two-dimensional subspace. In general, trying to fit a polynomial of degree $\leq m$ involves projecting onto a subspace of dimension $m + 1$. And the subspace for smaller degree is contained in the subspace for bigger degree.

Since the subspaces are growing, the distance from the vector $\vec{y}$ to the subspace is decreasing, because the subspace is spreading out more and more over the place. What we want is a situation where $\vec{y}$ already gets close enough to the subspace that going up in dimension does not bring one much closer.

5.4. **The general caution against overfitting.** A revealing quote is by mathematician and computer scientist John von Neumann: "With four parameters I can fit an elephant. And with five I can make him wiggle his trunk." Another is by prediction guru Nate Silver: "The wide array of statistical methods available to researchers enables them to be no less fanciful and no more scientific than a child finding animal patterns in clouds." In other words, don't be overimpressed by impressive fits if they use lots of parameters.

The concern behind *data mining* is a discrete version of the same concern.

6. The computational procedure for ordinary least squares regression

6.1. **The transpose of a matrix.** For a $n \times m$ matrix A, we denote by A^T the $m \times n$ matrix that is obtained by interchanging the role of rows and columns in A. Explicitly, the $(ij)^{th}$ entry of A^T equals the $(ji)^{th}$ entry of A, where $1 \leq i \leq m$ and $1 \leq j \leq n$. A^T is pronounced A-*transpose* and is called the *transpose* of A. Note that the letter T is a fixed letter (chosen because it is the first letter of the word "transpose") and should not be mistaken for a variable in the exponent.

The transpose of a matrix is a useful construction because taking dot products involves an interplay of rows and columns.

Note also that in the above scenario, A describes a linear transformation from $\mathbb{R}^m$ to $\mathbb{R}^n$, whereas A^T describes a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$. However, although it might be tempting to believe this, these linear transformations are not merely inverses of one another. The relationship between them is far more subtle and deep. There are some special kinds of square matrices, called *orthogonal matrices*, for which the transpose coincides with the inverse.

6.2. **The normal equation.** Recall that we are trying to solve:

$$X\vec{\beta} = \vec{y} - \vec{\varepsilon}$$

where we are attempting to pick $\vec{\varepsilon}$ to be as small as possible. In our typical setup, X is a $n \times m$ matrix, where n is the number of input-output pairs and m is the number of parameters. $\vec{\beta} \in \mathbb{R}^m$, whereas $\vec{y}$ and $\vec{\varepsilon}$ are both vectors in $\mathbb{R}^n$.

As discussed previously, this is achieved when $\vec{\varepsilon}$ is perpendicular to the image of the linear transformation given by X. In fact, $\vec{\varepsilon}$ is the *unique* choice of vector satisfying *both* these conditions:

1. The system $X\vec{\beta} = \vec{y} - \vec{\varepsilon}$ is consistent (i.e., $\vec{y} - \vec{\varepsilon}$ is in the image of the linear transformation with matrix X).
2. The vector $\vec{\varepsilon}$ is perpendicular to all the columns of the matrix X. In other words, $X^T\vec{\varepsilon} = \vec{0}$. Here, X^T is a $m \times n$ matrix, and the $\vec{0}$ appearing on the right is now a vector in $\mathbb{R}^m$.

Thus, if we multiply both sides of the equation in Condition (1) by X^T, we obtain:

$$X^T(X\vec{\beta}) = X^T(\vec{y} - \vec{\varepsilon})$$

This simplifies to:

$$X^TX\vec{\beta} = X^T\vec{y} - X^T\vec{\varepsilon}$$

By Condition (2), the term $X^T\vec{\varepsilon}$ becomes $\vec{0}$, and we are left with:

$$(X^TX)\vec{\beta} = X^T\vec{y}$$

Note that X^TX is a $m \times m$ square matrix, and both $\vec{\beta}$ and $X^T\vec{y}$ are vectors in $\mathbb{R}^m$. In other words, the system of equations that we are now solving for $\vec{\beta}$ has an equal number of variables and equations.

One small note. When we multiply by a matrix, we potentially lose information. In this case, we may be worried that solving the new system might give many solutions that are not solutions to the old system. However, it turns out that there is no loss of information in this case: the solution vectors $\vec{\beta}$ to the above are precisely the same as the solution vectors for $X\vec{\beta} = \vec{y} - \vec{\varepsilon}$ for the correctly chosen $\vec{\varepsilon}$.

Recall that our eventual goal is to find the vector $\vec{\beta}$, not the vector $\vec{\varepsilon}$. Our ability to obtain a new form of the equation that does not feature $\vec{\varepsilon}$ is therefore a feature, not a bug. The equation above is called the *normal equation*. We can attempt to solve this equation for the vector $\vec{\beta}$ given our knowledge of X and $\vec{y}$, without intermediately attempting to discover $\vec{\varepsilon}$.

In the case that the matrix X^TX is invertible, we can write the general solution as:

$$\vec{\beta} = (X^TX)^{-1})X^T\vec{y}$$

Note that the invertibility of X^TX is equivalent to X having full column rank (the statement is obvious in only one direction, so you'll have to take me on faith). Note, however, that unless the inverse of X^TX

is pre-computed, it is generally easier to solve the normal equation directly using Gauss-Jordan elimination rather than first compute the inverse and then multiply.

Finally, note that we *can* find the error vector $\vec{\varepsilon}$ once we know $\vec{\beta}$, as follows:

$$\vec{\varepsilon} = \vec{y} - X\vec{\beta}$$

6.3. **Multiple outputs for the same input.** We might sometimes be in a situation where we are performing a linear regression where the same input appears with two different outputs in different input-output pairs. The idea here is that we have attempted measuring the output for a given input multiple times, and came up with different values each time. We need to be a little more careful regarding the validity of general statements about rank, but otherwise, the procedure stays intact.

6.4. **A baby case: the line through three points.** We consider a very special case of linear regression: the case where we have three input-output pairs (x_1, y_1), (x_2, y_2), (x_3, y_3) and we are trying to fit $y = f(x)$ for a linear function $f(x) = \beta_0 + \beta_1 x$. We take β_0 as the first variable and β_1 as the second variable (note: in our earlier discussion, we had used $mx + c$ as the general description, but we're now using better notation). In the no-error case, we would be trying to solve:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

In the case that we have measurement error, we wish to project $\vec{y}$ onto the two-dimensional subspace that is the image of the (linear transformation given by the) coefficient matrix. The projection is the vector $\vec{y} - \vec{\varepsilon}$ and the difference vector $\vec{\varepsilon}$ is perpendicular to the image.

The image of the linear transformation given by the coefficient matrix is a plane in $\mathbb{R}^3$ with basis given by the column vectors of the matrix, i.e., the vectors:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

In the language we used in the general description, we have:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}, \vec{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, m = 2, n = 3$$

The transposed matrix X^T is given as:

$$X^T = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \end{bmatrix}$$

The product matrix is a 2×2 matrix given as follows:

$$X^T X = \begin{bmatrix} 3 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & x_1^2 + x_2^2 + x_3^2 \end{bmatrix}$$

We also have:

$$X^T \vec{y} = \begin{bmatrix} y_1 + y_2 + y_3 \\ x_1 y_1 + x_2 y_2 + x_3 y_3 \end{bmatrix}$$

Thus, the system that we are trying to solve for β_0 and β_1 is:

$$\begin{bmatrix} 3 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & x_1^2 + x_2^2 + x_3^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} y_1 + y_2 + y_3 \\ x_1 y_1 + x_2 y_2 + x_3 y_3 \end{bmatrix}$$

The general version of this is described in Section 5.4 of the text, Example 5 (Page 243-244 in the 4th Edition).