

# Lecture 1: The Geometry of Linear Equations

Date: Wednesday, February 4.

After some brief administrative information, I began with an overview of what the course is about, and what we hope to accomplish. I started by writing down a typical pair of linear equations in 2 variables that you might see in high school, and remarked that these are linear equations they all probably already knew how to solve it. Of course, there would be no subject of linear algebra if we never had more than 2 unknowns – in general we have  $m$  equations in  $n$  unknowns, and  $m, n$  may be millions, billions, or even infinity (for PDEs). To solve such large systems one typically needs computers, and computers are such powerful tools for linear algebra that learning computational techniques for their own sake is pointless – you are never going to solve large systems by hand (even  $3 \times 3$  or  $4 \times 4$  is large enough to just use the computer), nor are you likely to ever write your own computer package outside of a class (free software like LAPACK will blow you away). Moreover, computationally speaking, the way we solve problems with thousands of unknowns is really just a better-organized version of the procedure you learned in high school. Instead, while we'll have lots of practice solving example systems, the focus of 18.06 is more abstract: given millions of equations in millions of unknowns, we want to develop techniques to understand the problem, for example by breaking it down into subproblems (matrix factorizations, eigenvectors, ...). Unlike a more abstract course like 18.700, in 18.06 we'll do this by starting with simple examples and gradually inferring more and more general rules (with proofs only at a simple level), rather than starting with abstract axioms (linear maps and vector spaces) and deducing the consequences from there.

I then wrote down an example  $2 \times 2$  problem and considered what Gil Strang (and the textbook) call the “row picture” (viewing it as the intersection of 2 lines) and the “column picture” (viewing it as a linear combination of the columns to get the right hand side). The former gets more and more complicated in higher dimensions: in 3d it is the intersection of 3 planes, in 4d the intersection of 4 hyperplanes, etcetera, whereas the latter is a more “scalable” way to think about it (and will later lead to the concept of subspaces and linear independence). I talked briefly about the singular cases, where the two lines are parallel or where two columns are linearly independent.

We then started looking at Gaussian elimination, starting with a  $3 \times 3$  problem, writing it down as a matrix, and subtracting rows to eliminate variables. I defined concepts like the “pivots”, did back-substitution, and briefly touched on the cases with zeros in the pivot positions – if there is a nonzero in a subsequent row you can just swap rows, otherwise you are out of luck and the procedure breaks down.

## Lecture 2: Elimination, Matrix Operations

Date: Friday, February 6.

I started with some organizational stuff: I reminded them that the first problem set was handed out and is due Wed (no exceptions), and reminded them of the collaboration policy

printed on the course information sheet. I also informed them that I'll be holding an optional Matlab tutorial session on Monday night in 4-237 at 6pm (although for the most part the problem sets will tell them the commands they need to know).

Then, I reviewed Gaussian elimination and back-substitution again, and talked in more detail about the singular case, showing (with an example) how you can get no solutions or infinitely many solutions depending on the right hand side. I then emphasized that we really want to study the structure of the elimination procedure, not just memorize it as a rote algorithm (although the students should certainly be able to do it in their sleep by the end of the term). To do this, we need a way to view the elimination steps algebraically, symbolically, rather than writing it out in sentences ("subtract 3 times row 1 from row 2"), just as ordinary algebra is quite painful without symbols like  $x^2 + x = 3$  (invented in the last few hundred years). I explained that, since what we are doing is some operation on a matrix  $A$  to get an upper-triangular matrix  $U$  (that is easy to solve), what we need is a matrix operation, in particular matrix-matrix multiplication.

Again, what we need with matrix multiplication is not just a way to compute it, but a way to think about it, and there are different ways to think about the same operation that are useful in different circumstances.

I reminded them that matrix-vector multiplication (a special case where one of the matrices has only 1 column) can be viewed in the high school way as rows of the matrix dot-producted with the vector, but it can also be viewed in the "column picture" from last lecture as a linear combination of the columns of the matrix. Similarly, the product of a row vector (which I defined as the transpose of a column vector) with a matrix can be viewed either as dot products of the row vector with the matrix columns *or* as linear combinations of the *row* of the matrix.

I then defined matrix-matrix multiplication. First as the usual computational picture where entry  $(i, j)$  of  $AB$  is the dot product of row  $i$  of  $A$  with column  $j$  of  $B$  (most of the students will have seen this before). Then, I explained that equivalently we can think of it as the product of the matrix  $A$  with each column of  $B$  in sequence, or as the product of each row of  $A$  with the matrix  $B$ . (In homework, they will also do another picture of matrix-matrix multiplication, breaking it down into products of submatrices.) The latter case means that each row of  $AB$  is a linear combination of the rows of  $B$  — this is exactly what we need to express elimination steps.

I then expressed one step of elimination on a  $3 \times 3$  example as multiplying  $A$  on the left by an elimination matrix  $E$ . We derived the elimination matrix by viewing each row as specifying a linear combination of the rows of  $A$ . I then gave an alternate derivation, by pointing out that  $E \cdot I = E$ , so we just need to do the elimination steps on the identity matrix to get the matrix  $E$  — this is a powerful idea that we will return to in the Gauss-Jordan algorithm to get the inverse of a matrix. I emphasized that, in practice, we don't actually form the matrices  $E$ , we just do the elimination operations directly on the matrix — the matrices  $E$  are something that we use conceptually to think about the elimination process and manipulate it algebraically.

## Lecture 3: Elimination Matrices and Matrix Inverses

Date: Monday, February 9.

We started by reviewing elimination matrices from the end of the last lecture, and finishing a small  $3 \times 3$  elimination example, expressing the final upper triangular matrix  $U$  as the product of elimination matrices multiplying the original matrix  $A$  on the left (corresponding to linear combinations of rows). I noted that elimination matrices, and matrix multiplication in general, is not commutative, and showed that we get something different if we multiply the elimination matrices by  $A$  in a different order. I also went back to the original equation  $Ax = b$ , and explained that what we are doing – the reason why elimination makes sense, algebraically – is that we are multiplying both sides of the equation on the left by a sequence of elimination matrices.

To really use the power of an algebraic formulation like this, however, I pointed out that it would be nice to have a way to divide matrices, so that we can move them from one side to the other of an equation. To do this, I defined the inverse of a matrix  $A$  as the matrix that gives the identity when multiplied by  $A$  (on the left or right, although I didn't prove that the left and right inverses are the same). Formally, we can then solve  $Ax = b$  as  $x = A^{-1}b$ . I emphasized that, in practice, you almost never want to compute the inverse of a matrix explicitly – to solve a system of equations, it is significantly less work to just do elimination directly. This is also true computationally, where instead one uses the LU decomposition we will cover in the next lecture. Matrix inverses are primarily a conceptual tool.

I pointed out that the inverse of a matrix need not exist, and indeed it cannot exist if the problem is singular (as the existence of the inverse implies a unique solution to  $Ax = b$  for all  $b$ ).

As a simple example of an inverse, we looked at the inverses of the elimination matrices, where it is easy to see that the inverse is obtained just by flipping the sign underneath the diagonal.

I then presented two ways to actually compute the inverse of a matrix, should you ever need to do so (and also as a way to convince oneself that the inverse exists by a constructive proof).

First, one can view the columns of the inverse matrix as the solution of a sequence of equations with the columns of the identity matrix as the right-hand sides. That is, one can solve  $A \cdot A^{-1} = I$ .

Second, and a bit easier to do by hand, is Gauss Jordan elimination:  $A^{-1}A = I$ , which means that  $A^{-1}$  corresponds to a sequence of row operations on  $A$  to turn  $A$  into the identity. If we do these same row operations on  $I$ , we will compute  $A^{-1}I = A^{-1}$ , the desired result. So, we augment  $A$  by  $I$ , then do a bunch of “downward” elimination steps to turn  $A$  into  $U$ , then rescale the diagonal to get 1's, then do “upward” elimination steps to eliminate above the diagonal and turn  $U$  into  $I$ .

## Recitation thoughts

Date: Week 2.

The two most unfamiliar concepts to the students are probably the case of singular matrices and the formation of elimination matrices. The latter we will explore in much more detail later in the course with null spaces etcetera, but it would probably still be helpful to work through a simple example of elimination in a singular system for the cases with one and multiple solutions. With elimination matrices, the key difficulty seems how to view transformations of the matrix as matrix multiplications, so some practice in converting row operations/permutations into multiplication on the left by matrices, and vice versa, is probably helpful. (On the pset, I asked them to do the same thing for column operations.) (As I mentioned, most of the students seem to have seen matrix multiplication before, but purely as a computational formula to be blindly applied.)

They may also generally need practice in elimination and back-substitution on  $3 \times 3$  systems (and larger, but  $4 \times 4$  is probably too big for recitation); as I said, they will have to be able to do this in their sleep by the end of the term (or really, by exam 1).

A nice problem for recitation might be to explicitly form the elimination matrices for say a  $2 \times 2$  problem corresponding to Gauss-Jordan elimination: form the elimination matrices to turn  $A$  into  $U$ , then form them to rescale the diagonal to 1's, then form the elimination matrices for "upward" elimination to turn  $U$  into  $I$ . Then verify that the product of all of these elimination matrices gives the inverse of  $A$ , by multiplying them together to get  $A^{-1}$ , and then multiplying the result by  $A$  to check that  $A^{-1}A = I$ .

## Lecture 4: $A = LU$ Factorization

Date: Wednesday, February 11

I started by briefly mentioning an omission that may seem surprising to the students: we haven't talked about determinants yet. When linear algebra is first presented in high school or 18.02 for  $2 \times 2$  and  $3 \times 3$  matrices, determinants are presented as a central tool, both to check whether the system is singular and to compute inverse matrices (via Cramer's rule). I told them that we're not going to talk about determinants until halfway through the term, when we will use them as a conceptual tool to understand eigenvalue problems. For solving linear systems, determinant-based techniques are totally useless for anything larger than  $3 \times 3$  matrices. (In fact, the best way to compute determinants, which you almost never do, is to do Gaussian elimination first; computing them directly by generalizing the methods they learned for  $2 \times 2$  and  $3 \times 3$  matrices is exponentially costly.)

I again emphasized that matrix inverses are something that we use to think about problems, but should almost never compute. It is convenient to write down expressions like  $A^{-1}b$ , but we should read that in practice as "the solution to  $Ax = b$ , computed by some means like elimination / back-substitution that does not involve computing  $A^{-1}$ ". If you ever find yourself computing the inverse of a matrix, you are probably missing a better way to solve the problem (unless the matrix inverse itself is the result you were asked to compute, of

course).

It may seem that matrix inverses are useful if you want to solve  $Ax = b$  for lots of right-hand-sides  $b$ : you compute the inverse matrix once, and from then on you just do a matrix-vector multiply. However, there is still a better way, and that is LU factorization.

LU factorization is just another way of expressing what we already know as Gaussian elimination. I showed (both by an example and abstractly), that if we do Gaussian elimination (without row swaps), and move all the elimination matrices to the right-hand side, we end up with an equation  $A = LU$ , where  $U$  is the upper-triangular result of elimination and  $L$  is lower triangular. Furthermore,  $L$  is 1 along the diagonal and the entries below the diagonal are just the multipliers from the elimination steps (with a sign flip), which means that you really get  $L$  "for free" – you don't need to actually multiply together the inverses of the elimination matrices.

Then I showed how, given the LU factorization, you can solve  $Ax = b$  quickly via two triangular systems. In a metaphysical sense, this is the fundamental idea behind Gaussian elimination: we factorize a hard problem ( $A$ ) into two easy problems ( $L$  and  $U$ ).

I then quantified this a bit. For an  $n \times n$  system, using very simple arguments, we showed that the operation count for solving a triangular system grows proportional to  $n^2$ , whereas the operation count for Gaussian elimination (= LU factorization) grows proportional to  $n^3$ . So, you do the LU factorization once (expensive), and from then on any new system involving  $A$  can be solved quickly.

However, I also pointed out a problem. They saw in homework that a  $1000 \times 1000$  system of equation takes about 1 second to solve on a computer. I claimed on the first day of class that people solve million-by-million systems routinely. Since the time grows as  $n^3$ , a million-by-million system would take about 30 years to solve on a PC, and requires 8 terabytes of memory. It is actually possible to solve such a problem in a few hours using 10000 processors, but this is hardly "routine". The real answer is that all extremely large systems have some special structure that allows you to solve them much more quickly – usually they are "sparse" (mostly zero). We'll talk about this more later in the term.

## Lecture 5: Permutations, Dot Products, and Transposes

Date: Friday, February 13.

I started by asking what happens to LU factorization if we have row swaps. I showed by an example that we can then generally get  $PA = LU$  instead of  $A = LU$ , where  $P$  is a permutation matrix. Actually, getting the  $P$  on one side seems rather tricky in general because the permutation is mixed up with the elimination operations (you may not know that you need to do a row swap until after several elimination steps). How to do this properly is a tricky bookkeeping matter that we won't cover in 18.06, and isn't covered in the text. For the most part, in 18.06, when we talk about LU factorization we will assume no permutations are needed for simplicity.

However, permutations are interesting matrices in their own right. We constructed a couple of examples. In these examples, we saw that  $P^{-1} = P^T$ , the transpose: mirror

flipping  $P$  across its diagonal. We also saw that multiplying by  $P$  on the right does a column permutation rather than a row permutation, and in fact it does the column permutation that is the inverse of the row permutation. [This is again because  $P^{-1} = P^T$ , but I didn't show that connection.]

This led into a general discussion of transposes. The whole algebraic reason for transposes is ultimately based on dot products (inner products). First, I explained that dot products can be expressed as  $x^T y$ , which allows us to manipulate them in terms of matrix operations. Next, if we take the dot product of  $x$  with  $Ay$ , i.e.  $x^T(Ay)$ , then it is good to have a way to make  $A$  act on  $x$  instead, and showed that  $x^T(Ay) = (A^T x)^T y$  if we define the transpose in the usual way. I defined it in a few equivalent ways, as swapping rows and columns, as "mirror-flipping" the matrix around its diagonal, as changing the entries from  $a_{ij}$  to  $a_{ji}$ . Ultimately, however,  $x^T(Ay) = (A^T x)^T y$  is the definition of transposition: the transpose is what moves the matrix from one side to the other of an dot product, and this is the whole *raison d'être* of transposition.

I gave a couple of special cases, which we'll be investigating in more detail later in the term. First, symmetric matrices  $A = A^T$ . A simple example of this is  $A = B^T B$  for any square matrix  $B$ . More generally, I showed by explicit calculation with the definitions that  $(AB)^T = B^T A^T$ .

Second, I talked about orthogonal matrices, with  $A^T = A^{-1}$ . I showed that orthogonal matrices preserve dot products and lengths, in that  $x^T y = (Ax)^T (Ay)$  if  $A$  is orthogonal. Then, I pointed out that permutations preserve dot products, and hence permutations are orthogonal matrices, which is why we noticed that  $P^{-1} = P^T$  in our examples. (Strictly speaking, I didn't prove the converse, that preserving inner products means that  $A^T = A^{-1}$ , although this is of course true; 18.06 students aren't likely to quibble over this, though.) As another example, I mentioned in passing that rotations of vectors clearly preserve lengths and angles, and it follows that rotations can be represented by orthogonal matrices.

## Lecture 6: Vector Spaces and Subspaces

Date: Tuesday, February 17.

Today we did vector spaces and subspaces, the column space, and the null space. I told the students that this may seem a little abstract to them at first, but they are essential practical tools in order to think about singular and non-square problems (coming soon), as well as to generalize linear algebra beyond matrices. Just as elimination simplified matrix problems by factorizing them into simpler problems, now we are going to simplify how we think about large systems by breaking the "space" of inputs/outputs into smaller spaces.

By a sequence of examples, I defined vector spaces rather simplistically as sets where you can add, subtract, and multiply by (real) numbers, with the result staying in the set. (Of course, the general definition goes much beyond that, talking about fields etc. as you all know, but that goes outside the scope of 18.06.) And a subspace is a subset of a vector space that is itself a vector space. The key point for defining things this way is that we need these operations to manipulate the space algebraically.

I started with the vector space  $\mathbb{R}^n$  of  $n$ -component column vectors, and gave a couple simple examples of things that are and aren't subspaces (noting that a quick test is that, at the very least, subspaces must contain the origin). I connected it to algebra by looking at a couple of  $2 \times 2$  and  $3 \times 4$  matrices and looking at the space of all vectors  $b = Ax$ , and pointed out that this forms a subspace. I gave a couple of non-column-vector examples like the vector space of matrices, or the vector space of functions.

I then defined this as the column space  $C(A)$  as, equivalently: all linear combinations of the columns of  $A$ , the span of the columns (defining "span"),  $Ax$  for all  $x$ , or all right-hand-sides  $b$  where  $Ax = b$  has a solution. I showed explicitly that this is always a subspace by showing that addition etc. are closed.

I then looked at the set of solutions  $x$  to a  $2 \times 2$  problem  $Ax = b$  (with  $b$  in the column space), and showed that this is **not** a subspace (in general). However, I pointed out that differences of solutions do form a subspace. I then formally defined the null space of  $N(A)$  as the set of all solutions  $x$  to  $Ax = 0$ . I showed that this is a subspace, and that differences of solutions to  $Ax = b$  are in the null space, and that taking any solution  $x$  and adding all possible null space vectors gives all possible solutions to  $Ax = b$ .

In the last couple minutes, I closed with another example of column spaces and null spaces to try to expand their horizons a bit. I reminded them that in 8.01 (freshman physics) and 18.02 (multivariable calculus), they looked at equations of the form  $F = -\text{gradient}U$ , where  $F$  is the force and  $U$  is a potential energy. I then asked what the null space of the gradient is, and after some hesitation a few students mumbled "constant functions". Hence, the equation does not have unique solutions; as they well know, you can add any constant to the potential and get the same force. I then asked what the column space is – that, is, what force fields  $F$  give a solution for the potential? Again, a few students mumbled "conservative fields" and I said more explicitly that the curl of  $F$  must be zero: the column space is the set of curl-free vector fields.

Next lecture, we will focus on how to compute the null space via elimination.

## Questions from Office Hours

Two examples from office hours today show that the notions of a subspace and column space were pretty hard for them to grasp. Maybe I got isolated cases and they probably need a bit more time to digest it, but it might be worth asking them if they understood / repeating it in recitations / putting a few more examples to the next pset?

**Response:** we'll definitely be spending a bunch more class and pset time on subspaces. It's bound to be difficult material for the students because it's one of the first truly new things they've seen in the class (everything else has been just rearrangements of Gaussian elimination).

There are, at least, numerous calculations with nullspaces and column spaces on the next pset, and one problem involving a vector space of matrices. Also, we'll have at least two more lectures devoted to various subspaces of  $\mathbb{R}^n$  (besides today's lecture, which is focused on computing the nullspace).

## Lecture 7: The Nullspace: Solving $Ax = 0$ , Row-reduced Echelon Form

Date: Wednesday, February 18.

I started with a brief review, reminding them of the definitions of the nullspace  $N(A)$ , column space  $C(A)$ , of subspaces, and why the nullspace is important for solving  $Ax = b$  (given one solution, you can add a nullspace vector to get another solution).

I then looked at the very simplest case, the one we have mostly focused on so far, where  $A$  is square and non-singular. Obviously,  $x = 0$  is in the nullspace, and a moment's thought reveals that this is the only vector in the nullspace. You can see this from the fact that  $Ax = b$  always has a unique solution in that case, but if there were a nonzero nullspace vector you would get multiple solutions. Or, more simply,  $Ax = 0$  has a unique solution  $x = A^{-1} \cdot 0 = 0$ .

I next did a simple  $2 \times 2$  case  $A = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$  where you can see the nullspace by inspection. Clearly,  $0$  is in the nullspace. And  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ . And  $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$ . And  $\begin{pmatrix} 2 \\ -2 \end{pmatrix}$ , ... and any multiple of  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ . Graphically, this forms a line in the plane through the origin. We say that the nullspace is any constant times  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ , or equivalently all linear combinations of  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ , or equivalently the span of  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ . This is the kind of solution we are looking for when we ask for the nullspace of a matrix: not just some specific vectors, but the span of some vectors giving the entire subspace.

Then we did a  $3 \times 4$  example (the same example matrix as in Gil's video lecture), this time by elimination to simplify the problem just as we did for  $Ax = b$ . This time, however, we didn't get an upper-triangular matrix, but instead we get a matrix  $U$  in echelon form. I defined the rank of  $A$  as the number of pivots, and defined the "pivot columns" of the matrix as the columns of the pivots, and correspondingly the pivot variables, and the free variables as the remaining variables. The point is that the rank is the number of (non-redundant) equations that we end up with, so we have to pick rank variables to solve for and assign the other ones to arbitrary values. There are actually several choices of which variables we could solve for and which we could leave arbitrary, but making the free variables arbitrary is a nice systematic choice.

The simplest value of the free variables is just to set them to zero, but that gives the zero vector, which we already knew was in  $N(A)$ . The next simplest value is to make one free variable = 1 and the others = 0, and these give the "special solutions". The span of the special solutions is the nullspace.

We then solved the problem a different way, by doing further steps to reduce  $U$  to *row-reduced echelon form*,  $R$ . First with an example, then more generally, then very generally, I showed that we can just read the special solutions right off the  $R$  matrix (the values of the

pivot variables are just  $-1$  times the coefficients in the free-variable columns).

We then did another example with the transpose of our previous  $3 \times 4$  matrix, again doing elimination to get  $R$ . The rank was the same (I remarked that this is no coincidence, we will see this in general later), and we got one special solution.

More generally, for an  $m$ -by- $n$  matrix with rank  $r$ , I pointed out that we would have  $n - r$  free variables and hence  $n - r$  special solutions. Later on, we will say that the dimension of the nullspace is  $n - r$ .

## Lecture 8: Solving $Ax = b$ for Nonsquare $A$

Date: Friday, February 20.

Today, we talked about the “complete” solution to  $Ax = b$ , writing all possible solutions  $x$  as a particular solution plus the nullspace.

The first question is whether a solution even exists. I gave them an example  $3 \times 4$  matrix  $A$  where the third row was the sum of the first two, and in this case it is clear that the third component of  $b$  must be the sum of the first two for a solution to exist. Correspondingly, we showed that exactly this condition falls out of the elimination process: if you do elimination on the augmented matrix of  $(A \ b)$ , then you get a zero row of  $U$  after elimination, and the requirement that you get a corresponding zero row in  $b$  precisely corresponds (in this case) to the third component being the sum of the first two.

More generally, we can state two equivalent conditions for  $Ax = b$  to have a solution. First, if any linear combination of the rows of  $A$  gives zero, the same combination of the rows of  $b$  must give zero. Second,  $b$  must be in the column space  $C(A)$ . It’s clear, from different perspectives, that either of these two conditions is necessary and sufficient for there to be a solution, but it’s not clear what the direct relationship between the two conditions is. Apparently, there is some relationship between the geometry of the rows and the geometry of the columns....this is something we will explore more in subsequent lectures.

Once this is established, we can now go about solving  $Ax = b$ . The solution won’t be unique, because there is a nonzero nullspace in the  $3 \times 4$  example. However, the nullspace tells us the extent of the non-uniqueness: once we find one “particular solution”  $x_p$ , the sum of  $x_p$  and all possible nullspace vectors gives all possible solutions. To find one solution,  $x_p$ , we can just do elimination.

There are lots of possible  $x_p$  solutions, of course. But a simple, systematic way to get it is to set the free variables to zero, and then you get  $r$  equations in the  $r$  pivot variables. We then did this for our example, and got the complete solution by adding the nullspace in.

I then did it a slightly different way, by continuing the elimination process to put  $A$  in row-reduced echelon ( $R$ ) form, and doing the same elimination steps to  $b$  (in our augmented matrix). I then observed that the result of these elimination steps on  $b$  gives exactly the pivot variables in  $x_p$ . I showed this in general, by pointing out that the pivot columns of  $R$  are just the identity, so the equation in the pivot variables is just  $I \cdot x = c$  or  $x = c$ . Thus, if we take the augmented matrix  $(A \ b)$  and reduce it to  $(R \ c)$ , we can just read off the particular solution from  $c$  and the nullspace from the free columns of  $R$  (as in the previous

lecture).

More systematically, for an  $m \times n$  matrix  $A$  with rank  $r$ , we considered several cases. Clearly,  $r \geq 0$  and  $r \leq m, n$ . The most common situation is the “full rank” case where  $r$  is as big as possible. If  $m \leq n$  (a “wide” matrix), full row rank ( $r = m$ ) implies that  $Ax = b$  always has a solution, but that the solution is not unique (unless  $m = n$ ), i.e.  $C(A) = \mathbb{R}^m$ . If  $m \geq n$  (a “tall” matrix), full column rank ( $r = n$ ) implies that  $Ax = b$  may not have a solution (there are  $m - n$  constraints on  $b$ ), but if the solution exists then it is unique since  $N(A) = \{0\}$ . Full row and column rank ( $m = n = r$ ) means that the solution always exists and is unique, and corresponds to square non-singular  $A$  where the inverse matrix exists. I pointed out that the converses are also true: if  $Ax = b$  always has a solution, then  $A$  must have full row rank; if solutions are always unique then  $A$  must have full column rank.

Finally, I did a brief example with infinitely many unknowns, where you solve for a function  $x(t)$ . In particular, I looked at the 18.01 (freshman calculus) problem  $dx/dt = -\sin(t)$ . The solution  $x(t)$  in this case, as they all quickly shouted out, is  $\cos(t) + C$ . I then identified  $\cos(t)$  as a particular solution, and  $C$  as the null space  $N(d/dt)$ . We did another example,  $d^2x/dt^2 = e^{3t}$ , in which case a particular solution is clearly  $e^{3t}/9$ , and the nullspace  $N(d^2/dt^2) = C + Dt$  for any  $C$  and  $D$  ... hence the complete solution is  $e^{3t}/9 + C + Dt$ . I pointed out that if they take 18.03 they will do lots more equations of this sort, where in that case the nullspace is called the “homogeneous” solution, and the arbitrary constants in the nullspace part are used to satisfy additional conditions (initial conditions or boundary conditions). In the same way, the nullspace degree of freedom in  $Ax = b$  can be used to satisfy additional conditions one may want to impose on the solution (e.g. find the  $x$  with minimum  $|x|$ ). In 18.06, of course, we will focus on the case of matrices and column vectors with finitely many unknowns, but I want to occasionally give them glimpses of the fact that these concepts are actually far more general.

## Lecture 9: Independence, Dimension, and Bases

Date: Monday, February 23.

Today I went over linear dependence/independence, bases, and dimensions of vector spaces.

We started with dependence and a few examples, going from the intuitive concept that one vector is dependent on some other vectors if it is in their span, to the general definition that a set of vectors is dependent if they have a linear combination (with at least one nonzero coefficient) that gives zero. I then connected this to the null space: if you make a matrix  $A$  by using a set of vectors as the columns of  $A$ , then the vectors are dependent if  $N(A)$  contains a nonzero vector.

Conversely, a set of vectors is independent if  $N(A) = \{0\}$ , i.e. if the matrix with those vectors as columns has full column rank.

I then defined a basis for a given vector space as a set of linearly independent vectors that spans the space. I gave several examples, e.g. the special solutions form a basis for the nullspace, or 1 and  $t$  are a basis for degree  $\leq 1$  polynomials  $p(t)$ , or the columns of any

invertible  $n$ -by- $n$  matrix are a basis for  $\mathbb{R}^n$ , and in the same examples pointed out that the basis is not unique.... but it always has the same number of vectors.

The number of vectors in a basis was then defined as the “dimension” of a vector space, and is an important invariant quantity that we use to characterize the “size” of a vector space. I talked about a few examples, and how this corresponds to our intuitive notion of “dimension”. Our intuitive notion of dimension actually extends further than dimensions of vector spaces... e.g. we intuitively understand the dimensionality of curved surfaces etcetera, and I briefly mentioned that to handle that case formally you have to use differential geometry, where they talk about the tangent space (which **is** a vector space and its dimensionality is defined just as in 18.06). One can go even further and talk about fractal dimensions, but I said we won’t do that at all in 18.06.

Given a vector space, knowing a basis and its dimension really tell us everything we need to know about it, so the key question is how to come up with these things for any given vector space.

For an  $m \times n$  matrix  $A$  with rank  $r$ , we have already seen that the null space  $N(A)$  has dimension  $n-r$ , and a basis for  $N(A)$  is the special solutions. I then talked very briefly about the column space  $C(A)$ , but I didn’t really have time to finish this case. We’ll finish  $C(A)$ , finding a basis etc., on Wednesday when we talk about all four fundamental subspaces.

## Recitation thoughts

Date: Week 4.

For recitation, there are a few things that they probably need to practice:

1. finding the nullspace  $N(A)$ , i.e. the special solutions
2. finding particular solutions to  $Ax = b$  and hence, combined with  $N(A)$ , the complete solution
3. finding bases and dimensionality of different spaces
4. understanding what are  $N(A)$ ,  $C(A)$ , subspaces, etcetera. (Some of the students still seem to be confused about this.)

It shouldn’t be hard to come up with problems to help them think about this.

A simple example problem would be to find a basis & dimension for the space of 4-component vectors whose components sum to zero. You might want to hint that this is the nullspace of some matrix.... after a little thought, they should realize that this is just the nullspace of  $A = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$  and hence solve it.

Another simple example would be to take the rank-1  $2 \times 2$  matrix  $A = \begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix}$ , and sketch the column space  $C(A)$  and  $N(A)$ . Then ask them to sketch  $C(PA)$  and  $N(PA)$  for a row swap  $P$ , then ask them to sketch  $C(AP)$  and  $N(AP)$  for a column swap (multiplying

by the same  $P$ ) on the right. The key point of this question is: what spaces change when you do (invertible) row and column operations, and what spaces stay the same?

PS. You can mention that exam 1 (next Monday) will cover everything up to and including lecture 8 (Friday), but will **not** include the material of the lectures this week (this is also on the syllabus). Practice problems (and solutions) will be given out on Friday.

## Lecture 10: The Four Fundamental Subspaces

Date: Wednesday, February 25.

On Wednesday, I went over all four subspaces: the nullspace  $N(A)$ , the column space  $C(A)$ , the row space  $C(A^T)$ , and the left nullspace  $N(A^T)$ . For each subspace I gave a way to find a basis and the dimensions for  $A = m$ -by- $n$  matrix of rank  $r$ . In particular, I showed them how to get this information from the elimination process to obtain the row-reduced echelon form  $R = EA$ .

$N(A)$  we had already covered: it has dimension  $n - r$  and one basis is given by the special solutions from  $R$ . The key point was that  $N(A) = N(R)$ , and  $N(R)$  is easy to find.

For  $C(A)$ , I showed by an example or two that  $C(A)$  is different from  $C(R)$  in general. However, there is an easy way to get a basis for  $C(A)$  by looking at  $R$ : I showed them that columns are dependent/independent in  $A$  if and only if the same columns are dependent/independent in  $R$ . By inspection, the pivot columns of  $R$  are independent and form a basis for the other columns of  $R$ . So, the pivot columns of  $A$  form a basis for  $C(A)$ , which has dimension  $r$ .

$C(A)$  is a subset of  $\mathbb{R}^m$ , and  $N(A)$  is a subset of  $\mathbb{R}^n$ , so clearly we are missing some subspaces if we want to partition the whole spaces  $\mathbb{R}^m$  and  $\mathbb{R}^n$ . I showed (or, well, argued informally) that what we are missing is the row space  $C(A^T)$  and the left nullspace  $N(A^T)$ . I drew Gil Strang's favorite diagram, the one in section 3.6 of the text.

For the row space, I argued that the row space of  $A$  is the same as the row space of  $R$  (which is just a linear combination of the rows of  $A$ ). Hence we can just use the pivot rows of  $R$  as a basis for  $C(A^T)$ , unlike the column space. It follows that the row space has dimension  $r$ .

From the fact that the row and column space have the same dimension, it follows that the ranks of  $A$  and  $A^T$  are the same. I emphasized that this is a remarkable fact. It is not at all obvious a priori that doing elimination on  $A$  and  $A^T$  will give the same number of pivots.

For the left nullspace  $N(A^T)$ , I first explained the origin of the name: if  $y$  is in  $N(A^T)$ , then  $y^T A = 0$ , hence "left" nullspace. Since we know that the rank of  $A^T = r$ , it follows that the dimension of  $N(A^T)$  is  $m - r$ .

One way of finding a basis for  $N(A^T)$  is, of course to do elimination on  $A^T$ . However, we can also read  $N(A^T)$  off from not  $R$  but rather the elimination matrix  $E$  that forms  $R = EA$  from  $A$ . (I reminded them that  $E$  is obtained by doing the same elimination steps on  $I$  as on  $A$ .) In particular, if a certain row of  $R$  is all zeros, then the corresponding row of  $E$  is in  $N(A^T)$ . These rows of  $E$  give a basis for the left nullspace.

# Lecture 11: Subspaces and Matrix Operations

Date: Friday, February 27.

I spent the first half of this lecture talking about the subspaces of  $BA$  compared to the subspaces of  $B$  and  $A$ . I spent the second half talking about the exam on Monday.

How subspaces transform under matrix multiplication is a nice little topic that pulls together a number of the things we have learned. It's not explicitly covered in the textbook, although the material is mostly there scattered in one place or another. It's pretty fundamental to understanding even the things we have already done, because  $R$  is related to  $A$  by a matrix multiplication  $R = EA$ , and we need to know how the subspaces of  $R$  relate to the subspaces of  $A$ .

I made a little table of what we already know. The nullspace and the row space are the same for  $R$  and  $A$ , while the left nullspace and the column space are different for  $R$  and  $A$  (in general). A simple way to understand this is to notice that the nullspace and the row space are in  $\mathbb{R}^n$ , and correspond to things that multiply  $A$  on the **right**, while the other two subspaces are in  $\mathbb{R}^m$  and correspond to things that multiply  $A$  on the **left**. Since  $E$  multiplies  $A$  on the **left**, the "right" subspaces aren't changed while the "left" subspaces are changed by a factor of  $E$ .

However, we need to be a bit careful, and to see this I looked at the more general case of the subspaces of  $BA$ .

I showed (very simple proof) that  $C(BA)$  is *contained* in  $C(B)$ , but isn't necessarily equal – it may be smaller. Similarly,  $C((BA)^T)$  is contained in  $C(A^T)$ . Conversely, the nullspace  $N(BA)$  contains  $N(A)$ , but may be larger. And the left nullspace  $N((BA)^T)$  contains  $N(B^T)$ . It followed that the rank of  $BA$  is less than or equal to the ranks of  $A$  and  $B$ .

To show how the rank can decrease, I gave an example of two rank 1  $2 \times 2$  matrices whose product was the zero matrix. When we looked more closely at this case, we saw that I had constructed two matrices such that  $C(A) = N(B)$ , in which case a little thought shows that  $BA$  must be zero.

Again, the key way to think about this is to think about the subspaces flowing from right to left. If  $C(A) = N(B)$ , then what comes "out" of  $A$  on the left feeds "into"  $B$  on the right and gives zero.

The reason the subspaces were the same for  $R = EA$ , or at least the same dimension in the case of the column space and left nullspace, was that  $E$  was invertible.

(This is not the only case where the subspaces will be the same, of course. A very important case that will come later on will be  $A^T A$ , which always has the same rank as  $A$ . I didn't get to this case.)

I then talked a little bit about the exam. Basically, I went over the material on the first page of the practice-exam handout (on the web site), and went over a couple of practice problems.

## Recitation notes

Date: Week 5.

For Tuesday recitations, you'll be handing out the graded exams. But I wouldn't spend a lot of time on exam post-mortem analysis, unless there was a particular mistake that many people in your section made. For Monday recitations, I wouldn't talk about the exam at all unless they ask about it; I doubt they'll want to think about the exam at that point.

Instead, I would push forward with material reviewing the four fundamental subspaces. Some reasonable practice problems might be problems 6 and 9 from section 3.6 in the book, along with any other questions you think are interesting and practical to do in recitation.

As a preview of upcoming stuff, it might be interesting to give them a problem comparing the row space or null space of  $A^T A$  to those of  $A$  for some small  $3 \times 2$  or  $2 \times 3$  matrix  $A$  (they should be the same). (The fact that  $N(A^T A)$  contains  $N(A)$  should be obvious from the discussion on Friday, but the converse is trickier! This will be covered in lecture but it wouldn't hurt for them to see it twice as it's rather important for least-squares analysis.)

## Lecture 12 = Exam 1

Date: Monday, March 2.

## Lecture 13: Graphs and Networks

Date: Wednesday, March 4.

After some brief comments on the exam, I went into a discussion of section 8.2 of the text.

I started by defining graphs and directed graphs, then discussed some of the situations where they arise. Then I said that our first task is to represent this by a matrix, which will then allow us to use the tools of 18.06 to understand graphs. I defined two representations. First, I gave the adjacency matrix (vertices  $\times$  vertices) for undirected graphs, which we will use later in the course to analyze how Google works. Second, I gave the incidence matrix (edges  $\times$  vertices) as in section 8.2.

I showed that, when you do row reduction on the incidence matrix  $A$ , dependent rows correspond to loops in the graph, and row reduction turns the graph into a (spanning) tree with no loops.

I then picked the particular application of the incidence matrix: analyzing circuits (which most of them have seen in some form in their freshman electromagnetism course at least). With each vertex we can associate a voltage, and with each edge a current and a voltage drop; the direction of the edge indicates the sign convention (e.g. which direction is a positive current flowing in).

If  $x$  is a vector of voltages, then  $Ax$  is the vector of voltage drops, and  $N(A)$  is the span of  $(1 \ 1 \ 1 \ 1 \ \cdots)$ , and corresponds physically to the fact that we can add any constant to all

the voltages without changing the voltage drops. More generally, I argued that  $N(A)$  is the span of  $(1 \ 1 \ 1 \ 1 \cdots)$  for any connected graph, and hence the rank of  $A$  is always  $n - 1$  for connected graphs.

$C(A)$  tells us what voltage drops are possible. If row reduction produces a zero row in  $A$ , the same operations on the voltage drops must give zero, and from there I derived “Kirchhoff’s voltage law” (KVL), that the sum of the voltage drops around a loop must be zero. Furthermore, we only need to sum around “independent” loops, given by the zero rows in the row reduction  $A \rightsquigarrow R$ . These “independent loops” are precisely a basis for the left nullspace of  $A$ .

If  $y$  is the vector of currents associated with each edge, then  $A^T y$  is the sum of the currents flowing into each vertex. Kirchhoff’s current law (KCL) is the statement that the total current flowing into each vertex is zero, i.e.  $A^T y = 0$  or  $y$  is in the left nullspace.

Then I put them together, since the voltage drops are connected to the currents by multiplying by a diagonal conductance matrix  $C : y = CAx$ , and hence we are left with the equation  $A^T CAx = 0$ . More generally, if we put in a current source at some vertex, the right-hand-side would be nonzero. I mentioned, as Gil does in the book, that  $A^T CA$  and  $A^T A$  are very important matrices that show up in a lot of different problems.

I didn’t ask them any homework questions on section 8.2. It’s a fun illustration of a lot of the concepts we’ve been doing, but my feeling is that we can’t go much farther without asking the students to understand more graph theory (or circuit theory), neither of which are prerequisites or the subject of this course.

## Lecture 14: Orthogonality and Subspaces

Date: Friday, March 6.

I told them that, if the central theme of the lectures up to now has been elimination, the central theme of the next half-dozen lectures or so will be orthogonality. I started with a little overview, and then afterwards the main topic was orthogonal subspaces, orthogonal complements, and the orthogonality of the four fundamental subspaces.

I started with orthogonal vectors,  $x^T y = 0 = y^T x$ . I pointed out the transposes, and mentioned that we will be seeing transposes of matrices and vectors a lot in the next few lectures. Literally “orthogonal” (from Greek “orthogonios”) means “right angle” in the sense of “correct angle” or “canonical angle”, so I told them I wanted to give a sense of why orthogonal vectors are so nice and important.

A simple example is the Pythagorean theorem for right triangles, and I showed that  $|x + y|^2 = |x|^2 + |y|^2$  in any number of dimensions if  $x^T y = 0$ .

Another example are the coordinate axes: we don’t necessarily have to draw our coordinate axes as orthogonal, but it is a lot nicer that way — orthogonal axes means that changes along one axes can be distinguished easily from changes along other axes. I mentioned that we will return to this in a future lecture with the notion of orthogonal bases (and orthonormal bases). To go from an arbitrary basis to an orthonormal one, we will derive the Gram-Schmidt process, and this will lead us to  $A = QR$ , another of the important matrix

factorizations (like  $A = LU$ ) that we will derive in this course.

Another example is that, if you draw a line, and ask what the closest point on the line is to some point off the line, the "projection", you project orthogonal to the line. This will lead us to orthogonal projections and least squares approximations. The key point will be to solve  $Ax = b$  approximately, when  $b$  is not in  $C(A)$ , by minimizing  $|Ax - b|$ , and this will lead to us using orthogonality.

We will return to all these topics later. The main topic of this lecture, as I told them, was orthogonal subspaces.

Two subspaces  $S$  and  $T$  are orthogonal if every vector in  $S$  is orthogonal to every vector in  $T$ . I gave them a few examples of subspaces that are and aren't orthogonal in 2 and 3 dimensions, and noted that orthogonal subspaces must intersect only at the origin.

The most important example of orthogonal subspaces, in 18.06, are the four fundamental subspaces. I showed explicitly, and via an example, that  $N(A)$  and  $C(A^T)$  are orthogonal, and similarly for  $N(A^T)$  and  $C(A)$ .

Furthermore, we saw that  $N(A)$  consists of **every** vector orthogonal to  $C(A^T)$ , and this led me to define orthogonal complements, denoted by a superscripted perpendicular:  $N(A) = C(A^T)^\perp$ . We did a couple of examples in 2 and 3 dimensions, and saw from an example that  $(S^\perp)^\perp = S$ . I didn't prove this, and mentioned that it is true but quite a bit trickier to prove than showing that  $(A^T)^T = A$ , for example. (I made the mistake of assigning this as a homework problem once before, thinking it would be easy.)

(Aside: it is very tricky to prove  $(S^\perp)^\perp = S$  for infinite-dimensional spaces. In fact, it isn't even true in that case unless you restrict yourself to closed subspaces. I didn't mention this in class, though.)

Using this fact, I explained something that seemed a bit mysterious a few lectures ago.  $Ax = b$  has a solution if and only if  $b$  is in  $C(A)$ . Equivalently (we claimed), it has a solution if, whenever row-reduction on  $A$  gives a zero row, the same operations on  $b$  give a zero row. The connection between these two statements was rather mysterious at the time, but now by rewriting the second statement in equations I showed that it was simply the statement that  $b$  is orthogonal to the left nullspace. We now know that this is equivalent to saying that  $b$  is in  $C(A)$ , because the two spaces are orthogonal complements.

## Lecture 15: Projections

Date: Monday, March 11

The topic of today was orthogonal projections, which we will often just call "projections" in 18.06 (as we won't look at any other kind of projection). The sequence of topics was: projections onto lines, rank-1 matrices, projections onto subspaces, and  $A^T A$ .

I drew a line spanned by a vector " $a$ " and a point  $b$  off the line, and asked how we can find the closest point  $ax$  on the line to  $b$ . Graphically, I observed that  $ax$  must be the point so that the "error"  $e = b - ax$  is perpendicular to the line. (I didn't prove this.) Hence, we will call  $ax$  the "orthogonal projection" of  $b$  onto the line.

Furthermore, we can now write the equation  $a^T e = 0$ , and from that obtain  $ax =$

$(aa^T)/(a^Ta)b = Pb$ , where we have defined the projection matrix  $P$ . I observed that projecting twice must give the same point, and correspondingly showed explicitly that  $P^2 = P$ .

Furthermore, the matrix  $aa^T$  is quite interesting, and more generally I looked at matrices of the form  $uv^T$  for any vectors  $u$  and  $v$ , along with an example. I called this the “outer product” of  $u$  and  $v$ , the opposite of the inner product (dot product)  $u^Tv$  (which only works if  $u$  and  $v$  have the same length). I showed that  $uv^T$  was always a rank-1 matrix, with a column space given by the span of  $u$ , and the row space given by the span of  $v$ , and the null space given by vectors orthogonal to  $v$ . I remarked that, much later in the course, we will learn something called the singular value decomposition that decomposes any rank- $r$  matrix into the sum of  $r$  rank-1 matrices.

I pointed out something else. There are two ways to compute the projection  $Pb$ :

- (1) Compute  $aa^T$ , then divide by  $a^Ta$ , to get  $P$ . Then multiply by  $b$  to get  $Pb$ . This takes about  $2n^2$  multiplication for  $a$  in  $\mathbb{R}^n$ .
- (2) Compute  $(a^Tb)$ , then multiply by  $a$  and divide by  $a^Ta$ . This takes about  $3n$  multiplications (I think I said  $2n$  by mistake, not that it matters).

The difference between these two was just a difference in parenthesization, but it corresponds to a huge difference in the amount of work you have to do. You should almost **never** compute the matrix  $P$  explicitly, unless you are asked to, much like you should never compute the inverse of a matrix if you can avoid it. (This is one of many examples in linear algebra where just parenthesizing your expression differently makes a dramatic difference in the amount of work.)

Then I looked at projection of a point  $b$  onto an arbitrary subspace. To define the subspace, I supposed that we had some matrix  $A$  such that  $C(A)$  is the subspace. For example, the columns of  $A$  could be a basis (although  $A$  could have extra dependent columns too). Just as before, we want the closest point  $Ax$  in the subspace to  $b$ , and I claimed (without proof, just by analogy), that again we want the error  $e = b - Ax$  to be orthogonal to  $C(A)$ .

But since being orthogonal to  $C(A)$  is equivalent to being in  $N(A^T)$ , as we learned in the previous lecture, we can now immediately write that:

$$A^T(b - Ax) = 0,$$

or

$$A^T Ax = A^T b$$

from which we can find the projection  $Ax$ .

If  $A^T A$  is invertible, we can further write down the projection explicitly:

$$Ax = Pb = A(A^T A)^{-1} A^T b,$$

or

$$P = A(A^T A)^{-1} A^T.$$

It is easy to see that the previous rank-1 projection was a special case of this.

Again, I emphasized that you should almost never compute  $P$  (unless you are asked to do so explicitly). Instead, solve  $A^T Ax = A^T b$  for  $x$  by elimination, and then multiply  $Ax$ .

[Aside: Actually, one should almost never do that, either; you should almost always solve such problems using the  $QR$  factorization that we will learn later, to avoid catastrophically increasing the roundoff errors. But that's mainly a subject for another course, namely 18.335.]

Now, however, the key question is what are the properties of  $A^T A$ ? When is  $A^T Ax = A^T b$  solvable, and when is  $A^T A$  invertible?

I then showed that the rank of  $A^T A$  is equal to the rank of  $A$  and  $A^T$ , and hence  $C(A^T A) = C(A^T)$ . I showed this by the same trick as in the book: showing  $N(A^T A) = N(A)$ . If  $Ax = 0$ , then  $A^T Ax = 0$  is obvious. If  $A^T Ax = 0$ , we also have to show  $Ax = 0$ , which is trickier: we use the fact that  $x^T A^T Ax = 0 = |Ax|^2$ , and hence the result follows. This is probably the most complicated proof they've seen in the course so far (they'll prove it in a different way for homework), but the result is very important.

Because  $C(A^T A) = C(A^T)$ , it immediately follows that  $A^T Ax = A^T b$  always has a solution.

Furthermore, we see that  $A^T A$  is invertible if  $A$  has full column rank, i.e. if its columns form a basis for the subspace.

Next time, we will use this to solve  $Ax = b$  approximately, in the least-squares sense.

## Recitation suggestions

Date: Week 6.

I would suggest focusing recitation mainly on orthogonal projections, as this is almost certainly the newest and most confusing topic to the students, and there is a lot of material there (although we will review some of it on Wednesday).

A simple question, to make sure they understand the definitions is this: if  $P$  is the orthogonal projection matrix onto  $C(A)$ , then what is the orthogonal projection matrix onto the orthogonal complement [i.e., onto  $N(A^T)$ ]? (Answer:  $I - P$ ).

Another thing to have them check is that, assuming  $A$  has full column rank so that  $A^T A$  is invertible, that  $P^2 = P$ .

I would also have them compute orthogonal projections for a couple of examples: compute the projection of some  $b$  onto  $C(A)$  for a  $3 \times 2$  matrix with full column rank, and then a  $3 \times 3$  matrix with the same  $C(A)$  but not full column rank (i.e. throw in an extra column dependent on the first two).

## Lecture 16: Least Squares Approximations

Date: Wednesday, March 11

Today, we went over least-squares approximations and least-squares fits.

I began with a review of the previous lecture. To find the closest point  $A\hat{x}$  to  $b$  in  $C(A)$ , we project orthogonally so that the error  $e = b - A\hat{x}$  is orthogonal to  $C(A)$ . We then arrived

at the “normal equations”  $A^T A \hat{x} = A^T b$  for  $\hat{x}$ , and hence obtain the projection  $A \hat{x}$ . Note the notational switch: instead of “ $x$ ”, I told them we will follow the book and call the projection solution  $\hat{x}$  (not to be confused with the hat notation for a unit vector in 8.01 or 18.02). Furthermore, we found that  $C(A^T A) = C(A^T)$ , so the normal equations are always solvable. If  $A$  is full column rank,  $A^T A$  is invertible and we can explicitly write  $A \hat{x} = Pb$  with the projection matrix  $P = A(A^T A)^{-1} A^T$ .

With this brief review, I then looked at a couple of little warm-up questions.

First, if  $A$  is an invertible  $n \times n$  matrix,  $C(A)$  is  $\mathbb{R}^n$  and projection should do nothing. We can check this explicitly, since in this case we can write  $P = A(A^T A)^{-1} A^T = AA^{-1}(A^T)^{-1} A^T = I$ .

Second, I repeated a very good question that one student asked after Monday’s lecture. What happens if you have another matrix  $B$  with the same column space  $C(B) = C(A)$ ? Then projection with  $B$  should do the same thing. That is, if we solve  $B^T B \hat{y} = B^T b$ , we must get the same projection  $B \hat{y} = A \hat{x}$ . Note that  $\hat{y}$  is not equal to  $\hat{x}$  in general. Furthermore, this points out a nice little shortcut: if  $A$  does not have full column rank, we can replace  $A$  by a matrix  $B$  whose columns are a basis for  $C(A)$  (throwing out the dependent columns of  $A$ ), and then we get a smaller problem ( $B^T B$  is a smaller matrix) with the same solution for the projection. We can also get  $P$  explicitly that way, since  $B^T B$  is invertible even though  $A^T A$  is not.

With these warm-ups in mind, I then looked at the problem of solving  $Ax = b$  approximately when  $b$  is not in  $C(A)$ . In this case, there is no exact solution  $x$ , so we want to find the solution of the closest right-hand-side that does have a solution. But this just means replacing  $b$  with  $Pb$ , and solving  $A \hat{x} = Pb$  means that we solve the normal equations  $A^T A \hat{x} = A^T b$ . We then consider  $\hat{x}$  to be an approximate solution for  $Ax = b$ : the least-squares solution (although the reason for that name is not obvious yet).

To understand this solution better, I then claimed that  $\hat{x}$  minimizes  $\|Ax - b\|^2$  over all possible  $x$ . That is, it is the solution with minimum error or “residual”  $b - Ax$ . Following the book, there are three “proofs” of this.

Geometrically or “graphically”,  $Ax$  must be in  $C(A)$ , and the closest point in  $C(A)$  should be the orthogonal projection of  $b$ . More explicitly (the book’s “algebraic” proof), in orthogonally projecting  $b$  we decomposed  $b$  as  $b = A \hat{x} + e$ , where  $e$  is orthogonal to  $C(A)$ . Thus, for any  $x$ , we can substitute to write  $\|Ax - b\|^2 = \|Ax - A \hat{x} - e\|^2 = \|Ax - A \hat{x}\|^2 + \|e\|^2$  (by the “Pythagorean theorem” for the orthogonal vectors  $Ax - A \hat{x}$  and  $e$ , which we proved last Friday for any orthogonal vectors). Since  $\|e\|^2$  is fixed, the minimum is achieved when  $\|Ax - A \hat{x}\|^2 = 0$ , i.e. when  $x = \hat{x}$ .

For homework, they will prove it by calculus: they will write the equations of zero partial derivatives of  $\|Ax - b\|^2$  with respect to  $x$ , and obtain the normal equations. (The book does this, but only for the  $2 \times 2$  case.)

Hence the name “least-squares” solution:  $\hat{x}$  minimizes the sum of the squares of the components of the error  $e = b - Ax$ .

We then applied this to fitting (or “regression”). Supposing we had  $m$  data points  $(t_i, b_i)$ , and wanted to fit to a line  $C + Dt$  for unknown coefficients  $C$  and  $D$ , how would we do it?

First, we turned this into an equation  $Ax = b$  that would be satisfied for  $x = (C, D)$  if the line went through all of the points. But this is impossible in general if  $m > 2$ . Since we can't fit it exactly, a reasonable thing is to minimize the sum of the squares of the errors  $C + Dt_i - b_i$ , but this is exactly the least squares solution above to  $Ax = b$ . I explicitly wrote down the normal equations  $A^T A \hat{x} = A^T b$ , and we see that it is a rather simple  $2 \times 2$  matrix.

I then tried another case, where we fit to a parabola rather than a line, and showed how to turn this into a  $3 \times 3$  problem via the normal equations.

In homework, they will use the same principle to fit to higher and higher degree polynomials, and they will see that by doing so the fit doesn't necessarily improve in a sensible way! Over-fitting, with too many parameters, is always risky.

I then tried fitting to a decaying exponential  $Ce^{-Dt}$ , which is very important for lots of problems (e.g. population decline, particle decay, light attenuation etc.). In this case,  $D$  does not appear linearly in the problem so we can't write down the fitting as a matrix, at least not directly. But if we take the log of the fitting problem, we can turn it back into a straight-line fit for the unknowns  $\log C$  and  $D$ . This is what is done in practice (although strictly speaking it is minimizing a different measure of the error — the error on a log scale — than a least-square fit of  $Ce^{-Dt}$  directly).

Finally, I pointed out that there are other fitting problems where we can't solve this way. e.g. if we tried to fit to  $C \sin(Dt + E \cos(Ft))$ , we would find that no matter what we do the fit parameters come in nonlinearly. This is a nonlinear regression problem, and you have to solve it by nonlinear optimization techniques that we won't cover in 18.06. I did mention, however, that even nonlinear optimization methods often still use ideas from 18.06: a typical scheme is to linearize the problem by a Taylor expansion, solve the linear problem to find an approximate solution, then linearize again, and repeat...much like Newton's method. Linear equations are still an important building block for nonlinear problems!

Next time, we will look at  $QR$  factorization and orthonormal bases, which leads to an even better way (at least, better for the computer if not by hand) to solve equations than the normal equations. In Matlab, if you type "`x = A \ b`" to solve  $Ax = b$ , it will notice if  $b$  is not in the column space and will automatically do the least-squares solution by the best method.

## Lecture 17: Orthonormal Bases, Gram-Schmidt, and $A = QR$

Date: Friday, March 13.

On Friday I covered orthonormal bases, Gram-Schmidt, and  $QR$  factorization.

I started by defining an orthonormal basis  $q_1, \dots, q_n$  as a basis of orthogonal vectors of length 1. I motivated this by talking about Cartesian coordinates (in  $\mathbb{R}^2$ ) and pointed out how nice it is to have bases that are orthogonal, so that different directions are not mixed up. More generally, orthonormal bases are generally going to give us the "nicest" way to describe a vector space and have many useful properties in 18.06.

If we form a matrix  $Q$  whose columns are the orthonormal basis vectors, I showed that  $Q^T Q = I$ . I emphasized that it is **not** generally the case that  $Q Q^T = I$ , unless  $Q$  also happens to be square. I did a little  $3 \times 2$  example of this.

A square matrix with orthonormal columns is an “orthogonal matrix”, in which case we immediately see that  $Q^T = Q^{-1}$ . We actually saw orthogonal matrices before, when we talked about transposes. So, I only briefly reminded them of what we saw before: two examples of orthogonal matrices are permutations and rotation matrices, and orthogonal matrices have the property that they preserve lengths and dot products. e.g.  $\|Qx\| = \|x\|$  for any  $x$  if  $Q$  is orthogonal.

Back to the non-square case, we looked at the orthogonal projection matrix  $P$  onto  $C(Q)$ , and saw that  $P = Q Q^T$ . Equivalently, we solve  $\hat{x} = Q^T b$  and the projection of  $b$  is then  $Q \hat{x}$ . (This is another way to see why  $Q Q^T$  is not generally  $I$ : if  $Q$  is not square, its column space is a subspace of  $\mathbb{R}^m$ , and the projection matrix  $P$  must not be the identity.)

I pointed out that the projection being  $Q Q^T$  is actually something they’ve probably seen before. If they have a vector and want the component in the  $x$  direction, that is just the dot product of the vector with the unit vector in the  $x$  direction. Similarly, the dot product  $q_i^T b$  is simply the component of  $b$  in the  $q_i$  direction, which is the same thing as  $\hat{x}_i$ , and hence  $\hat{x} = Q^T b$ . This is a very useful fact that we will return to on Monday.

Now, I supposed we had an arbitrary matrix  $A$ , with full column rank, whose columns  $a_1, \dots, a_n$  are not necessarily orthonormal. How then do obtain a “nicer” basis, an orthonormal basis  $q_1, \dots, q_n$  for the same space  $C(A)$ ? The answer (or at least, an answer), of course, is Gram-Schmidt.

To illustrate Gram-Schmidt, I started with an example for  $n = 2$ . I did it both abstractly, for  $a_1$  and  $a_2$ , and for two example vectors in  $\mathbb{R}^3$ . Furthermore, I wrote down the process in matrix form, and showed that we were multiplying the matrix  $A$  by a sequence of upper-triangular  $n \times n$  invertible matrices, much like elimination matrices. Inverting these, I showed that we had written  $A = QR$ , where  $Q$  has orthonormal columns and  $R$  is upper triangular and invertible (or “right” triangular).

I also generally outlined the Gram-Schmidt process for  $n$  vectors—at each step, we subtract off the projections onto the previous vectors—and argued that since we always subtract multiples of earlier columns from later columns, the  $R$  matrix stays upper-triangular.

ERRATUM: In class, when I wrote down the general Gram-Schmidt process, I think I screwed up. I subtracted the projections onto the previous  $a_i$ ’s (the previous columns), when I should have subtracted the projections onto the previous  $q_i$ ’s. You can’t subtract the projections onto the previous  $a_i$ ’s one-by-one, you would have to subtract the projection onto their span, which is the general matrix projection operator. You can subtract the projections onto the  $q_i$ ’s one-by-one because they are orthogonal. I will mention this in class Monday; please also mention it in recitation.

I showed that  $R$  was upper-triangular in another way:  $R = Q^T A$ , and by inspection of this matrix we saw that the lower triangular corresponded to dot products  $q_i^T a_j$  for  $i > j$ , which were zero by construction (since Gram-Schmidt makes later columns orthogonal to the span of the earlier columns). There is actually a way to read the  $R$  matrix off the Gram-

Schmidt process, but I didn't show them this, and suggested that they can just compute  $R = Q^T A$  at the end.

Finally, I showed them how, given  $A = QR$ , you can use this to find the least-squares solution to  $Ax = b$ : Instead of solving  $A^T A \hat{x} = A^T b$ , I derived that the problem reduces to the triangular system of equations  $R\hat{x} = Q^T b$ . (They have to solve this in one of their homework problems.) On a computer, this is how you actually solve least-squares problems: you first do the  $QR$  factorization of  $A$ , then solve  $R\hat{x} = Q^T b$  — the reason is not because it is less work (it isn't) but rather because this technique has much less rounding errors. I told them that, when they are solving problems by hand starting with  $A$ , they should just solve the normal equations  $A^T A \hat{x} = A^T b$  as this is less work (and rounding error isn't an issue in homework problems where the solutions are usually integers or small fractions).

We won't cover modified Gram-Schmidt, which is purely a numerical optimization.

## Recitation suggestions

Date: Week 7.

There is a lot that was covered since Wednesday: least-square solutions and fitting, Gram-Schmidt, and  $QR$  factorization ... and on Monday we will do orthonormal bases of functions (Fourier series, and also Lagrange polynomials derived by Gram-Schmidt).

First, you should definitely cover Gram-Schmidt for a matrix with 3 or more columns, since I screwed this up in lecture. A useful exercise might be to do Gram-Schmidt on a  $3 \times 3$  matrix the wrong way and the right way and understanding the difference. Just to be clear, the wrong way, which I wrote down too hastily in lecture, is:

$$\begin{aligned} q_1 &= a_1 / \|a_1\| \\ q_2 &= \left[ a_2 - \frac{(a_1 a_1^T) a_2}{a_1^T a_1} \right] / \|\cdots\| \\ q_3 &= \left[ a_3 - \frac{(a_1 a_1^T) a_3}{a_1^T a_1} - \frac{(a_2 a_2^T) a_3}{a_2^T a_2} \right] / \|\cdots\| \end{aligned}$$

$q_1$  and  $q_2$  are correct, but  $q_3$  is wrong. The right way is:

$$\begin{aligned} q_1 &= a_1 / \|a_1\| \\ q_2 &= [a_2 - q_1 q_1^T a_2] / \|\cdots\| \\ q_3 &= [a_3 - q_1 q_1^T a_3 - q_2 q_2^T a_3] / \|\cdots\| \end{aligned}$$

A simple check to make sure that they grasp the fitting problem is to ask them to come up with a fitting problem and a corresponding matrix equation  $Ax = b$  where the solution is not unique. (The simplest example is probably fitting a line to a single point. You'll be amazed at how difficult some of the students will find this problem to be, however — I was, when I asked it on an exam in a previous term.)

Regarding orthogonal functions, I want them to understand a few things after Monday's lecture:

- (1) we can define a dot product of two functions by an integral
- (2) with this definition, we can now find orthogonal functions ... if necessary, by Gram-Schmidt
- (3) there exist orthonormal bases for the set of “all” functions (not getting into details of exactly what space of functions we are considering) ... to get the expansion coefficient for a particular function, we just take the dot product with that function
- (4) all of this is just like what we did for column vectors, with a new definition of  $a^T b$ .

One consequence of all this, which is in a homework problem, that we can solve least-squares and projection problems for functions.

A simple example problem would be to give them a function like  $f(x) = x^5$  on  $x$  in  $[-1, 1]$ , and ask them what is its closest approximation (in the least-squares sense) in the space of polynomials of degree 2 or less. You can use the fact that, in class on Monday, I plan to derive the first three Legendre polynomials by Gram-Schmidt on the functions 1,  $x$ , and  $x^2$ .

## Lecture 18: Fourier Series and Orthogonal Polynomials

Date: Monday, March 16.

Today, there were two handouts. These are a supplement to section 8.5 of the book.

The point of today’s lecture was to try to expand their minds towards the realization that what we’ve been doing — specifically, orthonormal bases, Gram-Schmidt, and orthogonal projection/least-squares — are not limited to column vectors, but can work for any vector space as long as we define a dot product. In particular, probably the most important other vector spaces are vector spaces of functions. Lots of problems in science, engineering, economics, etc. have functions as the unknowns (e.g. the concentration of some chemical as a function of time, the electric field in space, the velocity distribution in some fluid, the price of a stock option over time). Just as orthonormal bases are the nicest way to describe subspaces of column vectors, they are essential to understand and solve problems where the unknowns are functions.

I started with the Legendre handout, and looked at functions  $f(x)$  on the interval  $[-1, 1]$ , and defined the dot product of  $f(x)$  and  $g(x)$  as the integral of  $f(x)g(x)$ . The length of a function is then the square root of its dot product with itself, just as for column vectors.

With these definitions, I then considered the two functions  $f(x) = 1$  and  $f(x) = x$ , and showed that they are orthogonal, and hence when we normalize them to  $q_1$  and  $q_2$ , as in the notes, they form an orthonormal basis. Their span is all polynomials of degree  $\leq 1$ , so they are an orthonormal basis for all polynomials of degree  $\leq 1$ .

If we want an basis for all polynomials of degree  $\leq 2$ , we need to add in some quadratic polynomial, e.g.  $x^2$ .  $q_1$ ,  $q_2$ , and  $x^2$  are linearly independent and form a basis, but  $x^2$  is not orthogonal to 1 so it is not an orthonormal basis. We can make it an orthonormal basis,

however, by Gram-Schmidt: subtracting the projections of  $x^2$  onto 1 and  $x$ . I did this, as in the notes, and arrived at a function proportional to  $3x^2 - 1$ .

We can continue this process, if we want, to higher degree. What we are doing is just Gram-Schmidt on the functions  $1, x, x^2, \dots$ . The results (a few of which are shown on the handout, although I didn't go through the algebra in class) are known as the Legendre polynomials (modulo some normalization conventions). They are an orthonormal basis for polynomials of any degree, including infinite series (e.g. convergent Taylor series). And we can do anything with them that we did with orthonormal bases before.

So, for example, I went through projections onto a subspace, e.g. projections onto polynomials of degree  $\leq 2$ . And correspondingly I went through “least-square” fits of functions – e.g. if you want to find the polynomial  $p(x)$  of degree  $\leq 2$  that is closest to a function  $b(x)$  in the sense of minimizing  $\|p(x) - b(x)\|$ , you just project onto the first three Legendre polynomials.

The Legendre polynomials can be obtained by  $1, x, x^2, \dots$  by Gram-Schmidt. I then switched to a different set of functions, which are orthogonal without us having to do anything: the Fourier sine basis functions (see second handout).

In particular, I switched to looking at functions  $f(x)$  on  $[0, 1]$ , with the same inner product (but now integrating from 0 to 1), and considered the functions  $\sin(n\pi x)$ . I sketched them for the first few values of  $n$ . These functions, by a “lucky accident” (not really), are already orthogonal: I went through a quick 18.01 exercise to show that if you integrate  $\sin(n\pi x) \sin(m\pi x)$  you get zero unless  $m = n$ , in which case you get  $1/2$ . This means, if we multiply them by  $\sqrt{2}$ , they form an orthonormal basis.

Moreover, since it is an orthonormal basis, if we have a function  $f(x)$  that is written in terms of that basis, we can get the basis coefficients just by a dot product as in the notes.

But an orthonormal basis for what? What is in their span? The remarkable claim of Fourier in 1807, which took over 150 years to rigorously analyze, is that their span includes “all” functions. It eventually turned out that you have to exclude functions whose integral blows up, and you have to ignore isolated points of discontinuity, but for practical purposes Fourier's claim turns out to be essentially true. This is especially amazing when you consider that the sine functions are smooth and are zero at the endpoints – how can they possibly span nonsmooth functions or functions that are nonzero at the ends? I showed them the two examples from the notes that illustrate precisely this.

The plots in the handout are the Fourier sine series truncated to a few terms — we know that this is just the orthogonal projection of the function onto the subspace spanned by those terms, which is exactly the same thing as the least-square fit of the function by those few terms. Looking at the plots, you can see that the truncated Fourier series oscillates around the real function, just as you would expect a fit to do in order to minimize the squared error. You can also see that, the worse the discontinuity the Fourier series is trying to match, the slower the convergence and the bigger the ripples (Gibb's phenomena).

I mentioned, but did not show, that there are generalizations of this to cosine series, and to series of both sines and cosines (covered in the book), and even to the general Fourier transform defined not on a small interval but for functions on the whole real line. I also

mentioned the application to signal processing — an MP3 music file basically takes the sound waveform as a function of time, decomposes it into cosines, then throws out frequency components that are too weak to hear in order to compress the data to a smaller file.

## Lecture 19: Properties of Determinants

Date: Wednesday, March 18.

Today we covered section 5.1 of the text, going through properties 1-10 of determinants, and defining determinants by properties 1-3 as in the text. I'll be gone Friday, but Friday's lecture will be covered by Gil Strang.

Gil, the lecture is at 11am in 54-100. Yours will be the only other lecture exclusively about determinants in 18.06 this term. The only thing I absolutely need you to cover is the  $n!$  permutation formula for determinants, since I will need this fact for the characteristic polynomial in eigenvalue problems — explain that it is the natural descendent of the ad-bc type formulas they already know for  $2 \times 2$  and  $3 \times 3$  matrices, but also that it grows as  $O(n!)$  so it is not useful for large  $n$  (compared to the product-of-pivots formula). Beyond that, anything you want to talk about is fine. Volumes and Jacobians would be good. Anything you want to say about Cramer's rule and cofactors would be fine, although I've been emphasizing that these formulas aren't very useful beyond  $3 \times 3$  compared to Gaussian elimination because they require exponentially more operations. A nice practically-important case with a cute formula is the determinant of a tridiagonal matrix, although it's probably hard for them to appreciate why it is important.

Detailed summary of today's lecture:

I started with the explicit formula for the inverse of a  $2 \times 2$  matrix  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ . I checked by explicit calculation that it was the inverse. I asked for a show of hands, and nearly all of the students had seen it before. In this formula appears the inverse of  $ad - bc$ , which obviously must be nonzero for the inverse to exist. I said that this was the determinant of the matrix,  $\det A = ad - bc$ .

Again, I took a show of hands, and they had all seen this before, and they've also seen determinants of  $3 \times 3$  matrices (which they use for cross products in 18.02 and 8.01/8.02). Many of them had also seen the formula for the inverse of a  $3 \times 3$  matrix. I said that the generalization of these formulas was something called "Cramer's rule" for the determinant. And that this is one application of determinants: checking whether  $A$  is singular (if  $\det A = 0$ ) and finding the inverse.

But the reason we haven't been doing this already in 18.06, as I mentioned to them a few weeks ago, is that these formulas become truly horrible for larger matrices, and in fact require exponentially (or actually,  $n!$ ) work compared to the  $n^3$  work required for Gaussian elimination. There is a better way to compute determinants that we'll see later in the lecture, but it involves doing elimination first. Because of this, determinants are hardly ever computed directly for large matrices; however, they are useful to think about problems sometimes, and in particular they will be conceptually very useful for eigenproblems, which

will be the main topic of the course after exam 2. Basically, they are useful conceptually because they give an explicit formula for when the matrix is singular – even if we don’t use this formula directly, we will use it to think about singularity and its consequences.

I also reminded them that determinants are used in Jacobian factors in 18.02, and we’ll see (probably Friday) that determinants can be used to compute volumes. There are a few other cases where they are used too; I mentioned that I recently learned that the log of the determinant is used in quantum path integral calculations. However, determinants no longer occupy the central role in linear algebra that they enjoyed a century ago.

The purpose of today’s lecture is to define the determinant of any square matrix, not just  $2 \times 2$  or  $3 \times 3$ . The way we will do this is somewhat indirect: we will define it not by an explicit formula (one exists, but it is very hard to apply or reason from), but from its properties. In particular, we can define the determinant by 3 properties. Then we will derive lots of other properties from these, and also ways to calculate determinants.

I then went through the three defining properties ( $\det I = 1$ , row swaps flip the sign, and linearity with respect to individual rows).

After this, I went through the other properties listed in the book chapter 5.1, one by one, and numbered them 1 to 10 exactly as in the book.

I won’t list them all here, but I’ll touch on a couple of highlights that I emphasized.

One major result is that the determinant is the product of the pivots, multiplied by  $-1$  for every row swap during elimination. This is the most efficient way to compute determinants in general: do elimination first, then multiply the pivots. It also shows why determinants are not useful in general for solving systems of equations: in order to compute the determinant, you have to do elimination, and once you have done that you already know how to solve the problem and whether it is singular.

Another major result is that  $\det(AB) = (\det A)(\det B)$ . I didn’t actually prove this, I just told them that the proof is the most subtle of all the properties, and mentioned briefly that there is a proof in the book by showing that  $\det(AB)/\det(B)$  has the properties 1-3 of  $\det(A)$ . However, I used it to show various things, e.g. that  $\det(A^T) = \det(A)$  (and hence every row identity has a corresponding column identity), that the determinant of an orthogonal matrix is  $\pm 1$ , and  $\det(A^{-1}) = 1/\det(A)$ .

## Lecture 20: Formulas for Determinants; Jacobians

Date: Friday, March 20.

Taught by Professor Gilbert Strang.

## Lecture 21: Exam Review

Date: Monday, March 30.

## Lecture 22 = Exam 2

Date: Wednesday, April 1.

## Lecture 23: Eigenvalues and Eigenvectors

Date: Friday, April 3.

Friday, we started a major new topic, one that will occupy us for most of the rest of the term: eigenvectors and eigenvalues. The most important point for the students to understand is that, once we have an eigenvector, we can replace the entire matrix with a single number. Anything we can do with a number (multiply, powers, division, exponentiation), we can then do with the matrix, as long as we are working with eigenvectors. And since the eigenvectors usually form a basis, we can express any vector in terms of eigenvectors.

I started with a few general words explaining that our objective will be to replace the whole matrix with a single number, and discussing the importance of eigenvectors and eigenvalues in many fields, from physics to engineering to statistics. We will see many of these applications in more detail later in the term.

Then, I started with a simple  $2 \times 2$  example matrix,  $A = \begin{pmatrix} 5 & -1 \\ -2 & 4 \end{pmatrix}$ . Multiplying it by the vector  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ , we “miraculously” get the same vector multiplied by 6. This is an eigenvector of  $A$ , and 6 is the eigenvalue. If we multiply by  $A$  twenty times, we just multiply the vector by  $6^{20}$ . If we multiply by  $A^{-1}$ , we just divide by 6. If we multiply by  $\sin(A)$  or  $\exp(A)$  .... well, we don’t know quite what that means yet, but if it means anything, surely we should just multiply  $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$  by  $\sin(6)$  or  $\exp(6)$ .

More generally, I wrote the equation  $Ax = \lambda x$  as defining any eigenvector  $x$  and eigenvalue  $\lambda$ , where we require  $x$  to be nonzero. Our first task is to find these things, to see how many such solutions can exist and whether our initial  $2 \times 2$  example was just a lucky coincidence.

I then showed that  $x$  must be in the nullspace  $N(A - \lambda I)$ , and hence the eigenvectors form a subspace. (This is pretty obvious, looking at  $Ax = \lambda x$  – if we multiply  $x$  by a constant, for example, it still satisfies the same equation.) Furthermore,  $A - \lambda I$  is singular, and thus  $\det(A - \lambda I) = 0$ .

I then worked out  $\det(A - \lambda I)$  for the  $2 \times 2$  example, and we found that it was a quadratic equation, with two roots: the eigenvalue 6, which we already knew, and another eigenvalue 3. To find an eigenvector corresponding to  $\lambda = 3$ , we just need the nullspace of  $A - 3I$ . Nullspaces are something we already know how to compute; I quickly went through the calculation and found the eigenvector  $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ .

More generally, I pointed out that  $\det(A - \lambda I)$  must be a polynomial of degree  $n$  in  $\lambda$  for any  $n \times n$  matrix  $A$ , a fact we can deduce from the permutation formula for the determinant.

This is the characteristic polynomial. Usually, this will have  $n$  roots (possibly complex), with the rare exceptional cases of repeated roots that we will put off until later in the course.

So, our general procedure is now as follows:

- (1) Find the roots of  $\det(A - \lambda I) = 0$ , the eigenvalues  $\lambda_i$ .
- (2) For each eigenvalue, find a basis of  $N(A - \lambda I)$ , giving the corresponding eigenvectors  $x_i$ . (Usually, the nullspace will be one-dimensional, except in cases of repeated roots.)

I went through this in more detail for the general  $2 \times 2$  case  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ , in which case we have the nice formula for the characteristic polynomial:  $\lambda^2 - \text{trace}(A)\lambda + \det(A)$ , where the trace is the sum of the diagonal entries. In the  $2 \times 2$  case, it is also easy to see that the trace is the sum of the eigenvalues, and the determinant is the product. I stated (without proof) that this is true in general, and they are a nice little check on your calculations.

In the  $2 \times 2$  example, the two eigenvectors  $x_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$  and  $x_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$  are linearly independent and form a basis for  $\mathbb{R}^2$ . Hence, we can write any vector  $x$  as a linear combination of the eigenvectors. I gave a simple example of  $2x_1 + x_2 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$ , and pointed out that  $A^{123}$  times this is just  $6^{123}2x_1 + 3^{123}x_2$ . Note also that this is approximately  $6^{123}2x_1$ : multiplying by  $A$  repeatedly tends towards an eigenvector of the largest eigenvalue, a fact we will return to on Monday.

Our general procedure will be to find  $n$  eigenvalues and  $n$  eigenvectors for an  $n \times n$  matrix  $A$ . Using these as a basis, we can then write any vector in terms of the eigenvectors, and hence treat any operation involving  $A$  as just individual numbers multiplying each eigenvector.

The only thing that might upset this is a case where we somehow don't have  $n$  independent eigenvectors. This is possible, but rare. Such a matrix is called "defective", and we will not deal with defective matrices until later in the term.

Several important cases are **never** defective. An  $n \times n$  matrix  $A$  is not defective if either:

- (i) it has  $n$  distinct eigenvalues (usually true: repeated roots are unlikely for a random polynomial),
- (ii)  $A$  is real-symmetric, or
- (iii)  $A$  is orthogonal.

(ii) and (iii) have generalizations to complex matrices that we will discuss later. One of the complications of eigenvalue problems, as I mentioned in class, is that even real matrices may lead to complex eigenvalues, so you may have to deal with complex numbers whether you like it or not. We will defer complex matrices until later, however. (Real-symmetric matrices never have complex eigenvalues, something we will return to later.)

Finally, I talked about an important practical matter. Steps (1) and (2) to compute the eigenvalues and eigenvectors, above, are really something of a fraud. They are quite

useful for  $2 \times 2$  matrices. For  $3 \times 3$ , they are marginally useful, although you may have to solve a cubic equation. For larger matrices, they are useless...worse than useless, they are horribly inefficient and numerically unstable methods that are likely to give you garbage in practice. Good methods to compute eigenvalues weren't discovered until the 1960s, and have almost nothing to do with the process we use in 18.06. Problem 9 of the current problem set is actually the core of a "real" method to compute eigenvalues (the QR algorithm), but we won't go any further than this; it takes over a week in a graduate course (18.335) to derive just one modern method for how eigenvalues are really computed. As a result, the focus of this part of the course will not be on how eigenvalues are computed, but what their properties are and what you can do with them once you have found them (probably using black-box software like Matlab).

## Recitation suggestions:

Date: Week 9.

I would suggest focusing mainly on the new concept of eigenvalues and eigenvectors.

Remind them of the definition  $Ax = \lambda x$  of eigenvectors and eigenvalues, and start with a few simple questions:

If you know the eigenvalues of  $A$  are  $\lambda_1, \dots, \lambda_n$  and corresponding eigenvectors are  $x_1, \dots, x_n$ , then what are eigenvalues and eigenvectors of:

$A^2$ ?  $A^{-1}$  (assuming  $A$  is nonsingular)?  $A + A^3$ ?  $A + 3 \cdot I$ ?

Assuming the eigenvalues are real numbers, why must  $A^2 + I$  be nonsingular? What are the eigenvalues of  $1/(I + A^2)$ ?

Perhaps give them a simple  $2 \times 2$  matrix  $A$  with integer eigenvalues (say  $\begin{pmatrix} 2 & -4 \\ -1 & -1 \end{pmatrix}$ ), have them compute the eigenvectors and eigenvalues, and then compute (for example)  $1/(A^4 + I)$  multiplied by some vector  $x$  (not an eigenvector: get them to expand  $x$  in the basis of the eigenvectors by solving a  $2 \times 2$  linear equation for the coefficients). Like in the problem set (problem 7), let them leave the answer as a sum of eigenvectors multiplied by some coefficients (don't make them sum up the nasty fractions).

A nice article giving ways to come up with small integer-valued matrices with integer eigenvalues and eigenvectors is:

K. J. Heuvers, Symmetric matrices with prescribed eigenvalues and eigenvectors, *Mathematics Magazine* **55**, p.106-111 (1982).

Also, if you google for "eigenvalue example" (no quotes), you will find lots of small matrices with simple integer eigenvalues and eigenvectors. For example,  $\begin{pmatrix} 13 & 5 \\ 2 & 4 \end{pmatrix}$  or  $\begin{pmatrix} 13 & -4 \\ -4 & 7 \end{pmatrix}$  or  $\begin{pmatrix} 2 & -4 \\ -1 & 1 \end{pmatrix}$ .

## Lecture 24: Similar Matrices, Diagonalization, and Powers of $A$

Date: Monday, April 6, 2009.

On Monday, we continued with eigenvectors and eigenvalues, covering eigenvector bases, similar matrices, diagonalization, and matrix powers, with an example of the Fibonacci sequence.

I started with a brief review of the definitions and  $2 \times 2$  example from last time. The key point is that, once we have an eigenvector, the matrix acts just like a simple number. If we have a vector that is not an eigenvector, we write it as a linear combination of the eigenvectors and then the matrix acts just like a number on each term. For an  $n \times n$  matrix, we will almost always have  $n$  independent eigenvectors and corresponding eigenvalues, except in rare cases where the matrix is defective (which we will put off until later in the course).

So, our general strategy, given then eigenvectors and eigenvalues is:

- (1) Expand an arbitrary vector  $y$  in the basis of the eigenvectors  $x_i$ :

$$y = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

for some coefficients  $c_i$ .

- (2) Any operation involving the matrix  $A$ , acting on  $y$ , can be replace with that operation involving the eigenvalues acting on each  $x_i$ . The simplest case is

$$Ay = c_1\lambda_1x_1 + \cdots + c_n\lambda_nx_n$$

That is, each coefficient  $c_i$  gets multiplied by the corresponding  $\lambda_i$ .

What we want to do now is to write this process out in matrix form, which will lead us to new insight into the problem.

I wrote step (1) as  $y = Sc$ , where  $S$  is the matrix whose columns are the eigenvectors  $x_1, \dots, x_n$ , and  $c$  is the column vector with entries  $c_i$ . If the eigenvectors form a basis ( $A$  is not defective), then  $S$  is invertible, so  $c = S^{-1}y$ . (Note, as usual, that in practice it is usually easier to solve  $y = Sc$  by elimination than to compute  $S^{-1}$ , although the latter is useful for formal manipulation.)

I then wrote step (2) as

$$Ay = S\Lambda c = S\Lambda S^{-1}y$$

where  $\Lambda$  is the diagonal matrix of the eigenvalues. Since this must be true for any  $y$ , we must have

$$A = S\Lambda S^{-1}$$

or

$$\Lambda = S^{-1}AS$$

This is one of the important matrix factorizations in the course, like  $A = LU$  or  $A = QR$ . This is called the “diagonalization” of  $A$ .

It is a special case of a more general concept, called similar matrices. If we take any invertible matrix  $M$ , and write  $B = M^{-1}AM$ , then I showed that  $A$  and  $B$  have the same eigenvalues, and the eigenvectors are related by a factor of  $M$ . A common reason to use similar matrices is to show that  $A$  is similar to a “simpler” matrix, for which it is easier to find the eigenvalues – they are doing that in the problem set 6, with problems 9 and 10.

Diagonalization is simply saying that  $A$  is similar to a diagonal matrix  $\Lambda$ , if  $M = S$ .  $A$  is “diagonalizable”, which is true whenever  $A$  is not defective.

If we multiply by  $A^2$ , we know that we just multiply the eigenvectors by  $\lambda^2$ . We can see this from the diagonalization, too:

$$A^2 = (S\Lambda S^{-1})(S\Lambda S^{-1}) = S\Lambda^2 S^{-1},$$

where  $\Lambda^2$  is just the diagonal matrix of the eigenvalues squared. Thus, we see that the eigenvalues of  $A^2$  are  $\lambda_i^2$  with the same eigenvectors  $x_i$ . We can also see this directly by multiplying  $A^2$  by  $x_i$ . More generally, if we look at  $A^k$  for any power  $k$ , I showed that we get:

$$A^k = S\Lambda^k S^{-1}.$$

Not only do eigenvectors simplify the computation of things like matrix powers, they also give us new insight. To see this, I looked at the Fibonacci sequence  $0, 1, 1, 2, 3, 5, 8, \dots$ . Similar to the treatment in the book, I showed that we can treat this as repeatedly multiplying by a  $2 \times 2$  matrix  $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ . By finding the eigenvalues and eigenvectors of  $A$ , and expanding the initial condition in the eigenvectors, I found an explicit formula for the  $k$ -th term in the Fibonacci sequence. More interestingly, I showed that the ratio of successive terms approaches the golden ratio  $(1 + \sqrt{5})/2$ .

On Wednesday, I think I’ll do a bit more with linear recurrence relations to make sure that concept sinks in, and then we’ll cover Markov matrices, which are another interesting application of matrix powers.

## Lecture 25: Markov Matrices

Date: Wednesday, April 8.

On Wednesday, I spent a bit more time covering linear recurrence relations, for fun did an extension to recursive linear filters (which I told them will not be on the exams), and then started the topic of Markov matrices (which I finished on Friday).

To start with, I reminded them of the Fibonacci example we did on Monday, and told them that a more general name for this sort of thing is a “linear recurrence relation:” a sequence  $f_0, f_1, \dots$  where each term is a linear function of a finite number of the preceding terms. Because they are linear, we can always describe them by matrices and analyze them with eigenvalues and eigenvectors, as in the Fibonacci example.

I did another example, a “Tribonacci” sequence  $0, 1, 1, 2, 4, 7, 13, \dots$  where each term is the sum of the three preceding terms. In this case, we need to keep track of three terms at a time, so I showed how this is described by powers of the  $3 \times 3$  matrix  $A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ .

Computing the eigenvalues of this means that we have to solve a cubic equation, which is possible but annoying, so I just told them the eigenvalues (computed in Matlab):

$$\begin{aligned} &1.83929\dots \\ &-0.41964\dots + 0.60629\dots i \\ &-0.41964\dots - 0.60629\dots i \end{aligned}$$

In this case, two of them are complex. We will deal with complex numbers and matrices and eigenvalues in more detail in a few lectures, but the important thing is that if we multiply by  $A$  repeatedly, the dominant term will always be the one corresponding to the eigenvalue  $\lambda$  with the biggest absolute value  $|\lambda|$ . The absolute value of a complex number is the square root of the sum of the squares of its real and imaginary parts, which in this case gives  $0.73735\dots < 1.83929\dots$ , so the solutions of the Tribonacci sequence will eventually be parallel to the  $1.83929\dots$  eigenvector.

In principle, we could find the eigenvectors just by the null space of  $A - \lambda I$  for each  $\lambda$ , but I showed them a trick that only works for recurrence relations.  $Ax = \lambda x$  for an eigenvector, but  $Ax$  must also give the next step in the recurrence, which means that the components of  $x$  are shifted down by one. We can quickly see, as a consequence, that the components of the eigenvector  $x$  must be  $(\lambda^2, \lambda, 1)$ , scaling the last term to be 1 (for a 3-term recurrence) or  $(\lambda, 1)$  for a 2-term recurrence like Fibonacci.

Hence the ratios of subsequent terms in the Tribonacci sequence goes to  $1.83929\dots$  (the “Tribonacci constant”, which can be expressed in terms of square and cube roots but the formula is rather horrible: see <http://mathworld.wolfram.com/TribonacciConstant.html>)

I also pointed out an important fact (which I emphasized by the magic words, “typical exam question”): whether a recurrence relation blows up or not is determined by whether there are any eigenvalues with absolute value  $> 1$ . If the absolute values are all  $< 1$ , the terms must decay towards zero. If an absolute value equals 1, that term will neither grow or decay (but may oscillate).

For fun, I then did a little extension of this idea of linear recurrences. In a linear recurrence, we have  $\vec{u}_{k+1} = A\vec{u}_k$  for some matrix  $A$ . In signal processing, one adds to this some “input” vectors  $\vec{v}_k$  which “kick” the recurrence at each step:

$$\vec{u}_{k+1} = A\vec{u}_k + \vec{v}_k$$

This is a “recursive filter”, where the  $\vec{v}_k$  are the inputs and the  $\vec{u}_{k+1}$  are the outputs. For example, the inputs might be samples of some music, and the outputs might be filtered to remove noise or amplify the bass or something like that. If we start with  $\vec{u}_0$  and repeat this

equation multiple times, I showed that we obtain:

$$\vec{u}_k = A^k \vec{u}_0 + \sum_{n=0}^{k-1} A^n \vec{v}_{k-1-n}$$

Just as for recurrences, the outputs will not blow up (for bounded  $|v|$ ) if  $A$  has eigenvalues with magnitude  $\leq 1$ . In signal processing, the eigenvalues are called the “poles” of the filter, and some of the students learned them under this name in 6.01 (and they say that the filter is stable if the poles are inside the unit circle, i.e.  $|\lambda| \leq 1$ )

We can go one step further. Typically, signal processing, one is interested in analyzing particular types of signals, such as sine waves at a particular frequency. In particular, one looks at signals of the form

$$\vec{v}_k = z^k \vec{v}$$

for some constant  $\vec{v}$  and some number  $z$  (complex for sinusoids). Plugging this in above, we get:

$$\sum_{n=0}^{k-1} A^n \vec{v}_{k-1-n} = z^{k-1} \left[ \sum_{n=0}^{k-1} A^n z^{-n} \right] \vec{v}$$

where the expression in square brackets is a geometric series. I reminded them of the formula for a geometric series of numbers, and claimed (without proof) that we can use essentially the same formula for matrices:

$$z^{k-1} \left[ \sum_{n=0}^{k-1} A^n z^{-n} \right] = z^{k-1} (I - z^{-1} A)^{-1} (I - A^k z^{-k}) = (zI - A)^{-1} (z^k - A^k)$$

The only time this expression fails is when  $zI - A$  is singular, which occurs precisely when  $z$  is an eigenvalue of  $A$ . Such a point is called a pole. (In that case, if we go back to the original summation we find that it grows linearly instead of exponentially.)

Anyway, that business about recursive filters (which I spent a bit more time on than I intended to) aside, we then returned to another application of matrix powers: Markov matrices.

I defined Markov matrices, both in general and using a rather contrived example of population movement between the East and West coasts. I showed that such a process again corresponds to multiplying repeatedly by  $A$ , except that here  $A$  is Markov. Again, we will study this by decomposing the initial condition into eigenvectors of  $A$ , and then multiply each eigenvector by the corresponding eigenvalue repeatedly.

Key questions are: First, does this process diverge? i.e. is there an eigenvalue with  $|\lambda| > 1$ ? And second, is there a steady state, a solution that doesn't change, corresponding to  $\lambda = 1$ .

I claimed (deferring the proof to lecture 26) that there is always an eigenvalue  $\lambda = 1$ , and the other eigenvalues have  $|\lambda| \leq 1$ . Usually, the other eigenvalues have magnitude  $< 1$ . In this case, those eigenvectors decay away, and the system reaches an equilibrium, or steady

state, always converging towards an eigenvector of  $\lambda = 1$ . So, if we ask you for the steady state, as in homework, you don't have to solve for any eigenvalues: you know ahead of time that any Markov matrix has a  $\lambda = 1$  solution, and just have to find the corresponding eigenvector.

I did this in my little  $2 \times 2$  example, finding a steady-state eigenvector, and hence the steady-state populations of the two coasts in my model.

That concluded Wednesday's lecture. On Friday, we finished our analysis of Markov matrices by proving their key properties, and then moved on to linear systems of ODEs.

## Lecture 26: Differential Equations

Date: Friday, April 10. On Friday, we finished covering Markov matrices, and covered the first half of the material on linear systems of ODEs.

For Markov matrices, the main point was to prove the properties I had claimed in the previous lecture, that there is always a  $\lambda = 1$  eigenvector and that the other eigenvectors have  $|\lambda| \leq 1$ . I pointed out that this is an instance of a more general trend with eigenvectors: often, we will look at the structure of the matrix, and use this to tell us something about the eigenvectors and eigenvalues even if we cannot solve the problem by hand.

The basic strategy, here, is simply to write down the properties of Markov matrices as equations rather than as words; once we have expressed them in the language of linear algebra, then we can manipulate these expressions algebraically to quickly get the result we want.

In particular, the key property is that the entries of the columns of the matrix sum to 1. This can be written in equation form as  $[1 \ 1 \ 1 \ \cdots \ 1 \ 1]A = [1 \ 1 \ 1 \ \cdots \ 1 \ 1]$ . This is an extremely useful equation that tells us many things at once.

First, if we transpose it, we find that  $[1 \ 1 \ 1 \ \cdots \ 1 \ 1]^T$  is an eigenvector of  $A^T$  with eigenvalue 1. But we saw already on the pset that  $A$  and  $A^T$  have the same eigenvalues, although they may have different eigenvectors. So  $A$  must have an eigenvalue  $\lambda = 1$ .

Second,  $[1 \ 1 \ 1 \ \cdots \ 1 \ 1]\vec{u}$  for any vector  $\vec{u}$  is just the sum of the components (e.g. the "total population" in our simple Markov example). But since  $\vec{u}_k = A^k \vec{u}_0$ , by multiplying both sides by  $[1 \ 1 \ 1 \ \cdots \ 1 \ 1]$  we immediately find that the sum of the components of  $\vec{u}_k$  equals the sum of the components of  $\vec{u}_0$ : the "total population" (or the total probability, or whatever our Markov process measures) is conserved.

If there was an eigenvalue with magnitude  $> 1$ , that would correspond to exponentially growing processes, which is incompatible with the fact that the total population must be conserved. Well, actually, at this point we have to use the second property of Markov processes: that the components of  $A$  are non-negative. That means that if we start out with positive populations, they can't grow exponentially: multiplying them by  $A$  can never make them negative, and if the sum is bounded then the (positive) components are bounded. (If we allowed negative values of  $A$ , we could conceivably have an exponentially growing vector that still had a constant sum of its components, by using very large positive and negative quantities.)

Now, there is still a problem: if we have some other eigenvalue with  $|\lambda| = 1$ , but  $\lambda \neq 1$ , then we could have a process that oscillates forever without settling into a steady state. (I gave an example using the  $2 \times 2$  permutation matrix  $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ ) But, I quoted another theorem (without proof): if all of the entries of  $A$  are strictly positive (no zeros), then all of the eigenvalues have  $|\lambda| < 1$  except for one eigenvalue  $\lambda = 1$ . In this case, we always must obtain a steady state, and always the same steady state (up to multiplication by any constant).

Finally, I showed one other trick. Normally, to multiply by  $A$  repeatedly, we would expand the initial state in the basis of the eigenvectors, and then multiply each term by the corresponding  $\lambda$  repeatedly. But this is somewhat painful, because it would seem to require us to compute all of the eigenvectors in general. Instead, we can use a simple trick: whatever the steady state is, we know that it must have the same sum of components as the initial vector, because the sum of the components is conserved. So, we can just take the  $\lambda = 1$  eigenvector (computed via the nullspace of  $A - I$ ) and scale it to have the sum of the components matching the initial vector. This is why we can interpret the steady-state components as percentages, and usually write them by normalizing them to sum to 1.

The above occupied a little over half of the lecture. With the remaining half, I talked about linear systems of ODEs, a topic that we will return to on Monday.

I started with the simple ODE  $dx/dt = ax$ , where  $a$  and  $x(t)$  are numbers, with the initial condition  $x(0) = 0$ . It is pretty easy to see that the solution must be  $x(0)\exp(at)$ .

Next, we looked at a system of ODEs, which I quickly converted to a matrix form  $dx/dt = Ax$ , for a vector  $x(t)$ . If only  $A$  were a number, we would know how to solve this....but for an eigenvector, it is just a number! In particular, I showed that if  $x(t)$  is proportional to an eigenvector, then the proportionality constant must be  $\exp(\lambda t)$ .

So, the general strategy is simply this: expand the initial condition  $x(0)$  in the basis of the eigenvectors of  $A$ , then get  $x(t)$  by multiplying each term by  $\exp(\lambda t)$  for each corresponding  $\lambda$ .

I then compared and contrasted this with the linear recurrence equation  $u_{k+1} = Au_k$ . In that case, we expand the initial condition  $u_0$  in terms of the eigenvectors of  $A$ , and multiply each term by  $\lambda^k$  for the corresponding  $\lambda$ .

In both cases, the idea is to expand the solution in the eigenvectors, for which  $A$  acts like a number and the solution is simple. In the recurrence case, the recurrence  $u_k$  blows up if  $|\lambda| > 1$ . In the ODE case, the solution  $x(t)$  blows up if the real part of  $\lambda > 0$ . (In a couple of lectures, we will deal in more detail with the case of complex  $\lambda$ .) So, for example, a recurrence with eigenvalues of  $-0.1$  and  $-7$  blows up, while an ODE system with the same  $A$  will decay exponentially.

In the next lecture, we will look at ODEs again, this time from the perspective of matrix exponentials. I'll also do a couple of examples on converting systems of ODEs into a single higher-order ODE or vice versa.

## Recitation suggestions

Date: Week 10.

I would suggest focusing mainly on the system-of-ODEs case, as that will be the least familiar to most of the students, especially since we've spent several lectures on matrix powers but only half a lecture so far on ODEs.

A nice problem might be to start with a system of two ODEs, and then convert it into a linear recurrence relation by using a finite-difference approximation to the derivative, and then analyze the stability of both cases by looking at the matrix eigenvalues. That is, start with:

$$\frac{du}{dt} = Au$$

and then approximate it by  $u(k\Delta t) = u_k$  and:

$$\frac{u_{k+1} - u_k}{\Delta t} = Au_k$$

for a small  $\Delta t$ , giving the recurrence:

$$u_{k+1} = (A\Delta t + I)u_k$$

For example, use  $A = \begin{pmatrix} -5 & 2 \\ 4 & -3 \end{pmatrix}$ . Two questions

- (1) does the exact solution  $u(t)$  blow up for large  $t$ , or ...? (It goes to zero: the eigenvalues of  $A$  are  $-7$  and  $-1$ , both  $< 0$ )
- (2) How does the approximation  $u_k$  behave for large  $k$ ?  
answer: it blows up if  $\Delta t > 2/7$ , as this gives an eigenvalue  $< -1$ ; otherwise, if  $0 < \Delta t < 2/7$ , it decays exponentially.

The basic thing is that they should understand that in both cases you write things in terms of eigenvalues and eigenvectors, but in one case you use  $\exp(\lambda t)$  and in the other you use  $\lambda$  to a power (and the matrices are different in the two cases so the eigenvalues are different....but related).

I would also look in the book, which has a number of problems on ODEs and matrix exponentials that you can use for practice. I would stay away from problems requiring complex numbers for now.

## Lecture 27: Symmetric Matrices

Date: Monday, April 13.

On Monday, we mostly finished our coverage of ODE systems, although I will return to this topic when we talk about complex matrices in order to explain the role of complex eigenvalues, and started talking about symmetric matrices.

I talked a bit more about stability and steady-state solutions, and gave another example in which there were eigenvalues of 0 and  $-3$  and the solution tended to a constant.

I also talked about the equivalence between a system of  $n$  1st-order ODEs and 1  $n$ -th order ODE, and showed (with a couple of simple 2nd-order examples) how to convert from one to the other and back.

Finally, I talked about matrix exponentials. Like matrix inverses, it is rarely necessary to compute a matrix exponential explicitly, but they are very useful in algebraically manipulating and understanding the solution to a system of ODEs.

If we have one ODE  $dx/dt = ax$ , where  $a$  is a scalar, we already saw that  $x(t) = \exp(at)x(0)$ .

If we have a system of ODEs,  $dx/dt = Ax$ , I would like to write the solution in the same way:  $x(t) = \exp(At)x(0)$ . But what is the  $\exp(At)$  – how do we take the exponential of a matrix?

I gave three equivalent definitions, one by one, showing their equivalence:

- (1)  $\exp(At)$  is whatever matrix we multiply by  $x(0)$  to get  $x(t)$ . Such a matrix must exist, because the solution is a linear function of the initial conditions. This definition, while unambiguous, is a bit unsatisfying.
- (2) For  $A$  diagonalizable, I wrote down explicitly how we solve the system (first, expand  $x(0)$  in the basis of the eigenvectors, then multiply each eigenvector by  $\exp(\lambda t)$ ) in equations, and saw that we got:

$$x(t) = S \exp(\Lambda t) S^{-1} x(0)$$

where  $S$  is the matrix whose columns are the eigenvectors, and I defined  $\exp(\Lambda t)$  as the matrix with  $\exp(\lambda_i t)$  on the diagonal for  $i = 1, 2, 3, \dots$ . By comparison with definition (1), we conclude that

$$\exp(At) = S \exp(\Lambda t) S^{-1}$$

I also checked explicitly that my definition of  $\exp(\Lambda t)$  was consistent with (1).

- (3) By analogy with the Taylor series for  $\exp(t)$ , I defined  $\exp(At)$  as  $\sum_{k=0}^{\infty} (At)^k / k!$ . By differentiating with  $t$ , I showed explicitly that this was equivalent to (1). By plugging in  $A = S\Lambda S^{-1}$ , I showed explicitly that this was equivalent to (2).

In some sense, the most important point is (2): once you have an eigenvector, you can do anything with the matrix that you can do with a number (including exponentiation).

At this point, I told them, we are mostly done with ODEs. Of course, there a lot of other topics to cover, which are covered in 18.03, such as nonlinear equations or "inhomogeneous" equations  $dx/dt = Ax + (\text{something})$ . But we won't cover these in 18.06. The most glaring omission at this point is explanation of complex eigenvalues, which will come in a couple of lectures.

First, however, we will cover an important case in which the eigenvalues are **never** complex. Moreover, like we did with Markov matrices, we will learn of another way to use

the structure of the matrix  $A$  to reason about the properties of eigenvectors and eigenvalues, even for  $A$  that are too big to solve by hand. This case is (real) symmetric matrices,  $A = A^T$ .

I claimed (without proof for now) three important properties of a real-symmetric matrix: its eigenvalues are real, its eigenvectors are orthogonal (or can be chosen orthogonal), and it is always diagonalizable.

I gave a little  $2 \times 2$  example  $\begin{pmatrix} 0 & -2 \\ -2 & 3 \end{pmatrix}$ . Indeed, its eigenvalues were real ( $-1$  and  $4$ ) and its eigenvectors were orthogonal ( $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$  and  $\begin{pmatrix} -1 \\ 2 \end{pmatrix}$ ).

If we have an orthogonal basis of eigenvectors, we can normalize them to make an orthonormal basis. This means that our matrix  $S$  of eigenvectors becomes an orthogonal matrix  $Q$ , and we have the factorization:

$$A = Q\Lambda Q^T$$

This is nice for all the reasons that orthonormal bases are nice. In particular, when we want to expand a vector (e.g. our initial condition in an ODE) in the basis of the eigenvectors, we just need to do an orthogonal projection, i.e. a dot product with the corresponding eigenvector divided by its length squared.

I stopped there. On Wednesday, we will derive the first two properties of real-symmetric eigenproblems (I won't prove diagonalizability ...the proof involves induction on subspaces and is too subtle for 18.06), and talk about the additional condition of positive-definiteness.

At this point, I'm about half a lecture behind the syllabus, but I think I will catch up on Wednesday.

## Lecture 28: Positive Definite Matrices

Date: Wednesday, April 15.

On Wednesday, I proved that real-symmetric matrices have real eigenvalues, that eigenvectors with distinct eigenvalues are orthogonal, and that independent eigenvectors with the same eigenvalue can be chosen orthogonal by Gram-Schmidt (since the eigenvectors for a given eigenvalue form a subspace), leading to the  $A = Q\Lambda Q^T$  diagonalization.

In doing so, foreshadowing Lecture 29, I had to review a little bit about complex numbers. In particular, I defined the complex conjugate  $\bar{z}$  and showed that  $\bar{z}z = |z|^2$  is a positive quantity for nonzero  $z$ , the magnitude. I introduced taking dot products of complex vectors by the conjugate transpose, since without the conjugation I pointed out that  $(1, i)^T(1, i) = 0$ .

One of the most common ways to get a real-symmetric matrix is by  $A = B^T B$ , and I pointed out that in this case the eigenvalues will be nonnegative in addition to real. If  $B$  has full column rank,  $A$  will have strictly positive eigenvalues and is called positive definite.

In particular, I defined positive-definite matrices in two equivalent ways:

- (1) all eigenvalues are positive (for diagonalizable  $A$ )

(2)  $y^T A y > 0$  for all nonzero  $y$ .

I showed that these two are equivalent (for diagonalizable  $A$ ), and that  $A = B^T B$  implies (2) if  $B$  has full column rank. I also claimed, without proof, another test for positive-definiteness:

(3) All pivots of  $A$  are  $> 0$ .

Finally, I briefly discussed the Taylor expansion of a multivariate function  $f(\vec{x})$ , and showed that the second-order term is a second-derivative matrix, and that this matrix must be positive-definite at a point of zero gradient for the point to be a minimum.

## Lecture 29: Complex Matrices

Date: Friday, April 17.

In this lecture, I more generally defined  $x^H$  and  $A^H$  as the conjugate transpose (“adjoint”), and went through the list of topics in the course so far, from elimination to least-squares to eigenvalues, and argued that all was well for complex vectors and matrices as long as all transposed are replaced by adjoints. (If they are not, I pointed out several cases where disasters arise.) I defined Hermitian and unitary matrices as the generalizations of real-symmetric and orthogonal matrices.

Although fast Fourier transforms were listed on the syllabus, and Prof. Strange (very) briefly covers this, I decided to forgo the topic. As I explained to the students, I don’t feel that in half a lecture FFTs can really be presented in a form that they can actually make use of, particularly as they don’t know the applications of FFTs. Also, I’ve spent a lot of time on FFTs (I co-wrote the `fft` function in Matlab), and I’ve concluded that matrix notation is actually not the best way to express this subject; it obscures all of the most interesting parts. So, it is best left as a topic for other courses.

However, I did say that one of the places complex numbers arise is to express oscillating quantities, and quickly derived the famous Euler identity  $e^{i\theta} = \cos(\theta) + i\sin(\theta)$  from the Taylor expansions, and I pointed out that Fourier series become much nicer if expressed in terms of complex exponentials.

The most important case in 18.06 where complex numbers arise is as complex eigenvalues. I went through an example of  $d\vec{u}/dt = A\vec{u}$  with  $A = \begin{pmatrix} 0 & 2 \\ -2 & 0 \end{pmatrix}$ , which has purely imaginary eigenvalues, and showed that we got oscillating solutions. I first found the eigenvalues and eigenvectors, and pointed out that they must come in complex-conjugate pairs for a real matrix  $A$ . Then I derived  $u(t)$  for some initial condition, and showed that a real initial condition again gives a sum of complex-conjugate pairs in the solution and hence a purely real result. Finally, I expressed the final result in terms of sines and cosines.

At the very end, I briefly mentioned what would happen if you got a complex eigenvalue, say  $3 + 2i$  and  $3 - 2i$ . In this case, you would get the product of an exponential growth  $e^{3t}$  multiplied by an oscillating quantity.

## Recitation suggestions

I would suggest coming up with a  $2 \times 2$  real matrix  $A$  with complex eigenvalues, and then solving an ODE  $d\vec{u}/dt = A\vec{u}$  and/or a recurrence  $u_k = A^k u_0$ , for a real initial condition. Be sure to write the final result as a purely real quantity in terms of sines and cosines.

(Perhaps some review of polar forms of complex numbers would be useful.)

A nice problem relating to Lecture 28 is to consider unitary matrices  $A$  and to have them show (with some hints) that the eigenvalues  $\lambda$  have unit magnitude  $|\lambda| = 1$ . You can also prove that the eigenvectors are orthogonal (or can be chosen orthogonal for multiple eigenvalues), similar to the proof for real-symmetric matrices in class, but that is probably too much work for recitation. Especially since the students are probably still terrible at proofs.

## Lecture 30: Defective Matrices: Jordan Forms and Generalized Eigenvectors

Date: Wednesday, April 22.

I covered defective matrices, and my coverage was quite a bit different from the textbook and lectures from previous terms (in which the Jordan form appeared briefly, but without much explanation of where it came from or how it is used). I focused on the case of double roots of the characteristic polynomial, since this is by far the most common type of defective matrix. Almost everything I said is covered in the handout:

<http://web.mit.edu/18.06/www/Spring09/generalized-eigenvectors.pdf>

## Lecture 31: Singular Value Decomposition

Date: Friday, April 24.

This lecture was about the singular value decomposition (SVD). My approach again differed a bit from the book, but not too much. Although the book calls the SVD the “culmination” of the eigenvector approach, I pointed out that the emphasis of the SVD is a bit different. While most of our efforts on eigenvalues focused on their implications for computations, e.g. of linear recurrences or solutions to ODEs, the SVD is mainly focused on how to understand a matrix and its subspaces.

I started with the real-symmetric case, using a  $3 \times 3$  example  $B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}$ , and

pointed out that the eigenvectors give an ideal orthonormal basis for the four subspaces, and moreover an orthonormal basis whose orthogonality is preserved under multiplication by  $B$  (unlike for an arbitrary orthonormal basis we might construct by Gram-Schmidt). Furthermore, I showed that it allows us to write  $A$  as a sum of rank-1 projections onto

orthogonal subspaces, weighted by the eigenvalues. This allows us to break  $B$  up into its most basic pieces (rank-1 matrices) and know the relative contribution of each piece. The most important pieces of  $B$  correspond to the projections onto the spaces for the biggest eigenvalues.

We'd like to do the same thing for any matrix  $A$ , even non-symmetric, defective, or even non-square matrices. I gave an example of a  $3 \times 2$  matrix  $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$ .

Since we know how to handle real-symmetric matrices, we can make real-symmetric matrices out of  $A$  by computing  $A^T A$  and  $A A^T$ . Then, hopefully, we'll be able to work backwards to say something about  $A$ .

I computed the eigenvectors and eigenvalues of  $A^T A$  and  $A A^T$  (the latter giving me the matrix  $B$  I had already looked at, saving some time), and showed that there was a funny "coincidence:" the nonzero eigenvalues are the same.

I then proved that this is not a coincidence:  $A^T A$  and  $A A^T$  must always have the same nonzero eigenvalues. The nonzero eigenvalues must be positive (as we saw in our positive-definite lecture), so I called them  $\sigma_i^2$  for  $i = 1, \dots, r$  where  $r$  is the rank (recalling that  $A$ ,  $A^T$ ,  $A^T A$ , and  $A A^T$  all have the same rank, which we proved a long time ago). The proof is quite easy: if  $A^T A$  has an eigenvector  $v$ , multiplying both sides of the eigenequation by  $A$  shows immediately that  $Av$  is an eigenvector of  $A A^T$  with the same eigenvalue.  $Av$  cannot be zero because  $N(A^T A) = N(A)$ , and the eigenvalue was nonzero.

In fact, one can go a little bit farther and show that  $Av = \sigma u$ , and  $A^T u = \sigma v$ , if  $v$  and  $u$  are eigenvectors of  $A^T A$  and  $A A^T$ , respectively.

At this point, we were basically done. By writing any vector  $x$  in the basis of the eigenvectors  $v$ , we can immediately derive the form of the SVD:  $A = U \Sigma V^T$ .

(Here, I had to make a correction. I had accidentally swapped the letters  $u$  and  $v$ , so I derived the SVD as  $V \Sigma U^T$ . There is nothing wrong with this, of course, but I pointed out that everyone else in the world labels them in the opposite order.)

I emphasized, as Gil Strang does, that  $U$  and  $V$  give (in some sense) the best possible basis for the four subspaces: an orthonormal basis that turns into another orthonormal basis under multiplication by  $A$ , weighted by the singular values  $\sigma$ . It expresses  $A$  as a sum of rank-1 matrices, weighted by the singular values, which tells you the biggest orthogonal contributions to  $A$ , the most important "parts" of the matrix in some sense.

In particular, I talked a little about problem 5 on this week's homework, where you interpret an image as a huge matrix, take the SVD. If you then take the contributions just from the 50 biggest singular values, for example, you should see that it captures most of the "important information" in the matrix: the image is still recognizable. As another example, I mentioned that later in the course we will talk a bit about principal components analysis, where you can take a huge pile of data and use the SVD to extract the most important statistically uncorrelated contributions to the variation in the data. In general, taking the  $k$  largest singular values gives the "best" rank- $k$  approximation for  $A$  in some sense (which I didn't precisely define), which is why it tells you the most important information in the

matrix.

At the very end, I also defined the pseudoinverse  $A^+ = V\Sigma^+U^T$ , where  $\Sigma^+$  is like  $\Sigma$  but with  $\sigma_i$  replaced by  $1/\sigma_i$ . I claimed, without proof, that for an invertible matrix  $A^+ = A^{-1}$ , while for other matrices  $A^+$  gives precisely the least-square approximation. The main interest in the pseudo-inverse, however, is for nearly singular matrices. Sometimes you'll have singular values that are very tiny, say  $10^{-12}$ , which you suspect probably come from rounding errors or measurement errors of some sort. If you just naively compute  $A^{-1}$  or  $A^+$ , these tiny singular values will flip into huge numbers, say  $10^{12}$ , which will greatly amplify errors in your solution. So, what you do instead is to take the SVD of your matrix, and just throw out any singular values that are too small (set them to zero), and then when you do least-squares or compute  $A^+$  the solution will behave more nicely.

(I think I actually forgot to define the terms “singular vector” and “singular value”. I just called them  $u$ ,  $v$ , and  $\sigma$ .)

## Remarks

Hi gang, we are rapidly approaching exam 3. Starting with lecture 32 on Monday and also following exam 3, I'll be doing several lectures on interesting applications of linear algebra and also to give glimpses of more advanced subjects. It is unlikely that this material will end up on the final exam, although it should hopefully serve to reinforce the basics that the students have already learned. I also don't think we can have any more problem sets, since the last permissible due date for assignments is Friday May 8. So, after exam 3, you will have a bit more freedom in your recitation sections to review whatever past material you think your students need more practice with, or even introduce them to one or two new things for fun if you wish (although you should try not to let your recitations turn into lectures). For next week's recitation, though, you should probably focus on helping them to prepare for exam 3.

This week's problem set falls due on the Wed before the exam. So, Liang should try to make sure that the pset solutions are posted immediately on Wednesday night. I don't know if you will have time to grade them by Thursday, but you should make an effort to at least be available on Thursday to discuss the pset if the students need. However, I should note that my current plan is to have very limited coverage of defective matrices and SVDs on exam 3, probably restricted to true/false questions or fill-in-the blank at most.

## Lecture 32: ???

Date: Monday, April 27.

## Lecture 33: Exam Review

Date: Wednesday, April 29.

## Lecture 34: Exam 3

Date: Friday, May 1.

## Lecture 35: Linear Operators on Functions

Date: Monday, May 4.

Today's lecture, as I told them at the beginning, was one of three lectures in the course where I want to expand their minds to realize that linear algebra isn't just about matrices and column vectors. We had a lecture on vector space, where we saw that vectors could be things like functions. We had a lecture on orthogonal function where we saw that dot products and bases and Gram-Schmidt could apply to functions. And today's lecture shows that eigenproblems can be for functions too.

(I wrote up some notes on this topic in a previous term:

<http://web.mit.edu/18.06/www/Fall07/operators.pdf>.)

I started by reminding of one of the problems we looked at in lecture 32. There, we had the wave equation for oscillations of a stretched string, and I had given them the following equation for the different frequency modes  $y(x)$ :

$$-\frac{d^2}{dx^2}y(x) = \frac{\rho\omega^2}{T}y(x),$$

where  $\rho$  was the density,  $T$  was the tension, and  $\omega$  was the (angular) frequency. We then discretized this onto a grid, and replaced derivatives by finite differences, to obtain an eigenproblem  $Ay = \lambda y$ , where  $A = D^T D$  was the second-difference matrix (discrete Laplacian), made out of the first-difference matrices  $D$ . By the form of  $A$ , we could immediately see that the eigenvalues  $\lambda$  were real and positive, and hence the frequencies  $\omega$  are real, and the eigenvectors are orthogonal (the so-called "normal modes" of oscillation). In hindsight, this is rather unsatisfying: properties like the frequencies being real are fundamental physical results, and shouldn't depend on us discretizing the problem. They should follow from the original PDE.

I then observed that the PDE already looks sort of like an eigenproblem: something times  $y$  equals a number times  $y$ . We want to study this as an eigenproblem for the same reason we study eigenproblems involving matrices: knowing the eigenvectors turns a horrible problem involving a matrix...or a derivative...into a simple problem involving numbers. And, just like we could use the form of the matrix to tell us a lot about the solutions, we can do the same thing for PDEs.

To be more precise, I defined linear operators  $L$ :  $L$  operating on a vector should give another vector (possibly in a different vector space). And linearity means that  $L(ay_1 + by_2) = aLy_1 + bLy_2$  for any constants  $a, b$  and vectors  $y_1, y_2$ . If  $y$  is in the vector space  $\mathbb{R}^n$ , then  $L$  is just a matrix (with  $n$  columns). If  $y$  is a function, then examples of linear operators are derivatives, integrals, multiplication by constants or other functions, or combinations of these things.

In the wave-equation case, I defined  $y(x)$  as a function on  $[0, \pi]$  where  $y(0) = y(\pi) = 0$ , which is a vector space, with the dot product  $f(x) \cdot g(x) = \int_0^\pi f(x)g(x)dx$ . Our linear operator is  $A = -\frac{d^2}{dx^2}$ . We now have an eigenproblem  $Ay = \lambda y$ , where  $y$  is the eigenvector or “eigenfunction”,  $\lambda$  is the eigenvalue, and  $A$  is the linear operator.

In this case, it is pretty easy to solve for the eigenfunctions and eigenvalues. The only functions whose second derivatives are proportional to themselves are sines and cosines, but to be zero at the endpoints we must stick with sines only. Hence, our eigenfunctions are  $y_n(x) = \sin(nx)$ , with eigenvalues  $\lambda_n = n^2$ .

Notice that these functions are precisely the ones we worked with for Fourier sine series, and we already know (from an explicit integration) that they are orthogonal. Notice also that the eigenvalues are real and positive. Kind of like the solutions to a real-symmetric positive-definite eigenproblem! In fact, that is precisely what this is.

But what does it mean for a second derivative to be “symmetric?” To understand this, we need to define “transposes” of operators, not just of matrices.

As I’ve emphasized many times in the course, the only reason to ever do a transpose is if a dot product is lurking nearby. A transpose has the property of moving a matrix from one side to the other in a dot product:  $x \cdot (Ay) = (A^T x) \cdot y$ . In fact, this is “really” the definition of a transpose; the ordinary rule of swapping rows and columns follows as a consequence. A transpose is whatever we have to do to an operator to make it act to the left in an inner product. This allows us to define transposes (or “adjoints”) in general.

For the case here, I considered the problem of a derivative. If we have  $f(x) \cdot [\frac{d}{dx}g(x)]$ , what do we have to do to make the derivative act on  $f(x)$  instead? Writing out the dot product, we can immediately see that what one has to do is to integrate by parts. With the help of the boundary conditions, we found that the “transpose” of  $\frac{d}{dx}$  is exactly  $-d/dx$ .

This means that  $A = -\frac{d^2}{dx^2} = (\frac{d}{dx})^T(\frac{d}{dx})$ . It immediately follows that  $A$  is symmetric and positive definite (or at least positive semidefinite, but a little more effort shows that the nullspace of  $\frac{d}{dx}$  is only the 0 function). And hence  $A$  has positive real eigenvalues and orthogonal eigenfunctions.

The proofs of the properties of  $A$  are exactly the same as the proofs for matrices; if we think back, we never used the fact that they were matrices, only that transposes moved things from one side to the other in a dot product. Just to emphasize this, I reviewed the proof that the eigenvectors are orthogonal for distinct eigenvalues.

This also means that we didn’t actually have to do any integrals to know that the Fourier basis functions are orthogonal; it follows immediately from the fact that they are eigenfunctions of a Hermitian operator.

I then did a couple of examples. I solved Poisson’s equation  $-\frac{d^2}{dx^2}\phi(x) = \rho(x)$  by expanding  $\rho(x)$  in the eigenfunctions (i.e. a Fourier sine series) and then dividing each term by the eigenvalue, exactly as we might solve  $Ax = b$  if we had  $b$  in the eigenvector basis. I also solved the heat equation  $+\frac{d^2}{dx^2}T(x) = -\frac{dT}{dt}$  by expanding the initial condition  $T|_{t=0}$  in the eigenfunction (sine series) basis and then multiplying each term by  $e^{-\lambda t}$ , exactly as if we were solving  $\frac{du}{dt} = -Au$ . Thus, eigenfunctions allow us to do things like  $(\frac{d^2}{dx^2})^{-1}$  or  $\exp(t\frac{d^2}{dx^2})$  that otherwise would seem very mysterious, since we can replace the differential operator

with simple numbers.

Moreover, by looking at things like the symmetry of the PDE operator, we can say a lot about the solutions of equations that we cannot solve by hand. The example I gave was the (1d) Schrodinger operator  $-\frac{d^2}{dx^2} + V(x)$ , for some real function  $V(x)$ . Because  $-\frac{d^2}{dx^2}$  and  $V(x)$  are both symmetric, their sum is symmetric, and we can immediately conclude that the eigenvalues are real, etcetera, even if we can't actually solve it.

Finally, I closed with the following observation suggested to me recently by Gil Strang.  $\frac{d}{dx} \sin(nx) = n \cos(nx)$ . That is the operator  $d/dx$  acting on one orthogonal basis (sines) gives us a number times another orthogonal basis (cosines). This is precisely an SVD of  $\frac{d}{dx}$ . In fact, we have to use an SVD for  $\frac{d}{dx}$ , because in some sense  $d/dx$  isn't "square": the derivative of a function that is zero at its endpoints is no longer zero at its endpoints in general, so the input and output are in different vector spaces.

I told them that the formal study of this subject is called "functional analysis," and there are one or two courses on the subject offered at MIT if they are interested to learn more.

## Recitation remarks

As I mentioned after we finished exam grading, this is your "free week". Cover whatever topic in linear algebra you most enjoy that you think the students will appreciate. Try to give them some desk-work, perhaps in the second half of recitation, so that it is not completely a lecture format, however.

## Lecture 36: Google PageRank; Principal Components Analysis

Date: Wednesday, May 6.

I started this lecture by talking about principal components analysis (PCA): you have a big matrix  $A$ , and you want to understand the principal sources of variation between the columns of  $A$ , so you subtract the mean of the columns from each column and then take the SVD of the remaining matrix.

I illustrated this using one of my favorite SVD examples, the "BMLwalker" demo from

<http://www.biomotionlab.ca/>

which I projected onscreen. In this study, each column of  $A$  represents motion capture data for one person walking on a treadmill, and you do PCA to understand the principal sources of variation between people. The cool thing is that they have an animation, where you can turn a knob to add or subtract the top four singular vectors from the average walking person. These singular vectors turn out to be correlated with things like male/female or happy/sad, so you can see the "eigenmale" and the "eigensad" walking person. It's really striking – I urge you to visit the site if you haven't seen it before.

I mentioned that, since the singular vectors are eigenvectors of  $A^T A$  and  $A A^T$ , this can be understood from a statistical perspective because those are the covariance matrices. The fact that the eigenvectors diagonalize these mean that the singular vectors are statistically uncorrelated contributions to the variance, and the singular value is the variance. But since this is 18.06, not a statistics course, I didn't go into more detail.

I also mentioned that people use this for many other kinds of data, too, for example in facial recognition (google "eigenfaces"), although.

I then talked about using eigenvectors to analyze another huge dataset: Google's PageRank algorithm (or at least, the early published algorithm). First, I showed that the web can be represented by an adjacency matrix. Then, I talked about two derivations of PageRank. One is to consider a person randomly clicking on links, with some probability of jumping to a randomly chosen page, giving a Markov process. The steady-state eigenvector is the PageRank, proportional to the probability of being on a given page. Another way of deriving it is a self-referential definition: the PageRank of a page is the average of the PageRanks of pages that link to it, weighted inversely by the number of other links on those pages. I did a little example of a "web" with four pages: MIT, 18.06, my home page, and Harvard, and calculated the ranks of each page.

I ended with a little preview of the next lecture. How does Google solve for the steady-state eigenvector of a 25-billion-squared matrix (and one that is changing every day)? Even storing the matrix is out of the question. However, since the adjacency matrix is mostly zeros (most pages only have a few links), you only need to store the nonzero entries, and I showed that one can multiply by the  $n \times n$  Markov matrix in  $O(n)$  operations. Because of this, one need only multiply a random vector by the matrix repeatedly, and it will converge to the PageRank (steady-state) vector. This is called the "power method" for the biggest-magnitude eigenvalue and eigenvector.

## Lecture 37: Sparse Matrices and Iterative Methods

Date: Friday, May 8.

Today I talked about sparse matrices and iterative methods. I looked at two very simple methods: the power method (and inverse iteration) and steepest-descent. I told them they should never use these methods because there are much better methods available....but the better methods are much more complicated, and the simple methods give some of the core ideas of the fancier methods.

I started by talking about sparse matrices. Matrix problems up to about  $10^4 \times 10^4$  can be solved directly by methods like Gaussian elimination, but much larger than that becomes very difficult—even to store the matrix. Much larger matrices are common, e.g. Google's web data, or even a 3d PDE solution with a  $100 \times 100 \times 100$  grid gives a matrix of order  $10^6$ . These kinds of problems are solved routinely, but this is only possible because the matrices have some special structure.

Most commonly, very large matrices are sparse: mostly zero. If an  $n \times n$  matrix  $A$  has, say, at most 10 nonzero elements per row, then we can store it by storing roughly  $10n$

numbers, and multiply  $Ax$  with only  $10n$  multiplications instead of  $n^2$ . Examples of this include adjacency matrices of graphs like the web, or matrices arising from finite-difference approximation like we saw a few lectures ago.

The problem is that the algorithms we have learned in the class, like Gaussian elimination,  $LU$ , and  $QR$ , don't generally preserve sparsity. Even if you start out with a sparse matrix, you may end up with a dense matrix. We have to come up with algorithms that only ever multiply  $A$  times vectors. In fact, in many circumstance you may not even be allowed to look at the entries of  $A$ , you may only have a black box that takes  $x$  as input and gives you  $Ax$  – this happens because  $A$  may have some special structure other than sparsity that allows you to store it implicitly and multiply vectors by it quickly (e.g. using a fast Fourier transform).

This brings us to iterative methods. The basic idea behind iterative methods is to start with a guess for the solution  $x$  of  $Ax = b$  or  $Ax = \lambda x$ , and then have some step to improve your guess to a better guess, but only using  $A$  via matrix-vector products. The initial guess need not be very good, and typically is random numbers or zeros; the key thing is to come up with a step that improves the guess as rapidly as possible, so that we don't need to repeat the step many times to get an answer. This gives us an approximate answer in general, but the key thing is that we can get the answer to any desired accuracy, say 16 decimal places.

The simplest example of this is the “power method.” Multiply any random  $x$  by  $A$  many times, and it should converge to an eigenvector of the eigenvalue with the biggest magnitude. At each step, we probably want to divide  $x$  by its length to prevent it from blowing up. To get the eigenvalue at the end, we only need to compare  $x$  to  $Ax$ : each component should be multiplied by  $\lambda$ . But since it is not an exact eigenvector, each component will be multiplied by a slightly different factor, so we want some kind of average. In particular, a common choice is  $\lambda \approx x^T Ax / x^T x$ , the called the “Rayleigh quotient”. This is essentially the average growth factor for each component of  $x$ , weighted by the component squared.

I had Matlab running on the projector, so I did this process for a randomly chosen  $10 \times 10$  positive-definite matrix  $A$ , and showed them that after only a few steps we indeed got the biggest-magnitude eigenvalue and the corresponding eigenvector to many decimal places of accuracy.

To find the minimum-magnitude eigenvalue, we could instead multiply by  $A^{-1}$  repeatedly, a process called “inverse iteration”. This raises the question, however, of how to compute  $A^{-1}$ , or rather (since we should know by now that you should almost never compute matrix inverses explicitly) of how to solve linear systems  $Ax = b$  by an iterative method.

For simplicity, I restricted myself to positive-definite real-symmetric matrices  $A$ , and reminded them that solving  $Ax = b$  is then equivalent to minimizing the function

$$f(x) = \frac{1}{2}x^T Ax - x^T b.$$

At the minimum,  $\nabla f = Ax - b = 0$ , so  $Ax = b$  at the minimum.

Then, all we need to do, starting from a guess  $x$ , is to go “downhill” in this function in order to improve the guess. There are many, many algorithms in the field of numerical

optimization that can be applied to this problem, but I explained that I would show them only the very simplest techniques.

We started with the technique of line-searches. Instead of solving the whole problem at once, we would begin by minimizing  $f(x)$  only along a single line. That is, we solve

$$\min_{\alpha} f(x + \alpha d)$$

for some direction  $d$ . If we then repeat this for many directions  $d$ , we will hopefully approach the minimum over all  $x$ . The key questions are how to do the line minimizations, and how to choose the directions  $d$ .

The line minimizations are actually just an 18.01 exercise. After a little algebra, we found that the minimum is attained for:

$$\alpha = d^T r / d^T A d,$$

where  $r$  is the “residual” vector  $r = b - Ax$ .

The simplest choice for the direction vector is just the downhill direction:

$$d = -\nabla f = r.$$

This is called the “steepest descent” algorithm.

I then implemented the steepest-descent algorithm in Matlab (it is only a few lines), and tried it on a  $10 \times 10$  example. Unfortunately, it converges rather slowly. I wrapped it in a loop and plotted the norm of the residual vs. iteration, and we saw that it converged, but only after thousands and thousands of iterations.

So, you should never use steepest-descent. Fortunately, there is a much better algorithm that is only a slight modification to steepest descent, called the “conjugate gradient” algorithm. Essentially, one chooses the line-search directions  $d$  to be the gradient directions, but orthogonalized via Gram-Schmidt (cleverly implemented) so that  $d_i^T A d_j = 0$  for  $i \neq j$ . What this does is to make the line searches independent: after minimizing over 10 directions orthogonalized in this way, one has found the minimum over the *whole* 10-dimensional space spanned by those direction vectors. I didn’t prove this fact, or explicitly give the conjugate-gradient algorithm other than the above equations, I just talked about it qualitatively a little.

More generally, if  $A$  is not positive-definite or not symmetric, there are a whole host of more complicated iterative algorithms. There are also much better iterative schemes than the power method in general, typically keeping track of several vectors at a time so that they can handle closely-spaced eigenvalues. With dense-matrix algorithms, you are unlikely to ever need to write your own, and will always just use canned routines like those in Matlab. With iterative algorithms, unfortunately, the subject is still a matter of ongoing research and one typically needs a bit deeper understanding in order to use them effectively, and you typically need to try several approaches to see which one works best on your particular problem.

(And then there is the whole arcane art of preconditioning, but I didn’t get into that.)

## Recitation Suggestions

For recitation this week, I would just review for the final exam. The final exam will cover the material of exams 1-3, so the practice exams are a good source for the list of topics and for practice questions. You can also use previous final exams, etcetera. Use your sense of what topics the students have the most difficulty with. I would especially make sure to review exam 2's material: exam 3 is fresh in their minds, while exam 1's material was easier for them, so exam 2's material may be a danger spot.

## Lecture 38: Numerical Linear Algebra

Date: Monday, May 11.

Hi gang, this was the final lecture with new material (although it won't be on the exam).

Today's topic was numerical linear algebra: basically, how does our understanding of linear algebra change when you implement it in the real world on a computer.

To start with, I described a very fortunate situation: you should never have to write your own dense (non-sparse) linear-algebra code, or even your own matrix multiplication. The state of the art in linear algebra is encapsulated in a high-quality free-software library called LAPACK, which handles everything from LU factorization to the SVD. Programs like Matlab, GNU Octave, Scilab, Scipy, NumPy, etcetera, are built around LAPACK, which does most of the low-level linear algebra in the world. For performance, LAPACK in turn depends on something called the BLAS (Basic Linear Algebra Subroutines), which are standard routines for matrix multiplications, dot products, matrix-vector products, and similar operations. The idea is that computer vendors and other people will rewrite the BLAS from time to time to optimize it for new machines, and you just plug the latest optimized BLAS into LAPACK and you get a high-performance library. There is even a nice free implementation of BLAS called ATLAS which optimizes itself for your machine if you let it run overnight.

Not only will LAPACK be much more robust and full-featured than anything you are likely to write yourself, it will also be much faster. You might think that writing a BLAS would be trivial – a matrix multiply is just three loops, and can be written in 6 lines of code. But even though BLAS libraries like ATLAS are doing exactly the same arithmetic operations as those three loops, they are probably 10 times faster (at least), because they contain lots of tricks to deal efficiently with the slowest and most complicated part of your computer: the memory. So, in ATLAS, those 6 lines of code for a matrix multiply actually become around 150,000 lines of code.

The other issue that arises in real implementations of linear algebra, one which you cannot completely sweep under the rug, is error propagation.

For example, suppose you are solving  $Ax = b$  but  $b$  is not known exactly; instead, you only know  $b + \Delta b$  for some unknown error  $\Delta b$ . Then you are really solving  $A(x + \Delta x) = b + \Delta b$ , and the question is: how do errors in  $b$  propagate into errors in  $x$ ? This equation can be

solved rather quickly to obtain:

$$\Delta x = A^{-1} \Delta b$$

but that's still not quite useful. We don't actually know what  $\Delta b$  is, but we might know how big it is: we might have an estimate for  $\|\Delta b\|$ , and from that we would like to get an estimate of  $\|\Delta x\|$ .

Actually, that's not quite right either. An absolute error like  $\|\Delta b\|$  is not meaningful; if we multiply everything by 10 we haven't really changed the accuracy, but  $\|\Delta b\|$  is 10 times bigger. Instead, (like many things in science and engineering and economics) we need to nondimensionalize our error: we look at the relative errors  $\|\Delta b\|/\|b\|$  and  $\|\Delta x\|/\|x\|$  instead. This is a meaningful question, independent of overall scale factors and units: if we have some relative error in  $b$  (say a 1% error), what relative error does that give in  $x$  (1%? 10%? 0.1%?)?

Taking the norm of both sides of the above equation and performing some trivial algebraic manipulations, I quickly arrived at the result:

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|\Delta b\|}{\|b\|}$$

where I defined the induced matrix norm

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

which is a nice number depending only on  $A$ , measuring the “size” of the matrix  $A$ . By plugging in an eigenvector, I then showed that  $\|A\| \geq \max |\lambda|$ , the eigenvalue with the maximum magnitude. So, we can see that the matrix norm has something to do with the eigenvalues. I then stated, without proof, that in fact  $\|A\| = \max |\lambda|$  if  $A$  is real-symmetric, and more generally  $\|A\|$  is equal to the maximum singular value of  $A$ .

Since the eigenvalues and singular values of  $A^{-1}$  are the inverses of those of  $A$ , we can immediately see that  $\|A^{-1}\|$  is equal to the inverse of the minimum singular value of  $A$ .

Hence, the quantity  $\|A\| \|A^{-1}\|$  is a nice dimensionless number: the ratio of the biggest singular value to the smallest singular value of  $A$  (or the biggest/smallest  $|\lambda|$  if  $A$  is symmetric). We call this the *condition number* of  $A$ , and sometimes denote it by  $\kappa(A)$ .

Thus, we have:

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\Delta b\|}{\|b\|}$$

That is, the relative error in  $b$  is multiplied by the condition number to get the maximum relative error in  $x$ .

The best possible condition number is 1, when  $A$  is a multiple of the identity matrix: in this case we are just rescaling  $b$ , which doesn't change the relative error.

As  $A$  approaches a singular matrix, and one of the eigenvalues goes to zero, the condition number blows up. A matrix with a large condition number is called “ill-conditioned”, and it is near-singular. In this case, a small error in  $b$  can turn into a large error in  $x$ .

More generally, the condition number shows up in lots of places. For example, it turns out to be very important for the convergence rate of iterative solvers for  $Ax = b$ . The steepest descent algorithm requires a number of iterations proportional to the condition number. The conjugate-gradient algorithm requires a number of iterations proportional to the square root of the condition number or better, and there are additional tricks called preconditioning to try to improve the condition number of  $A$ .

Even if you think you know  $b$  and  $A$  exactly, errors will still show up in practice, because the computer represents numbers in scientific notation with a finite number of significant digits, about 15 decimal digits of precision, called “floating-point arithmetic”. Every time you perform a floating-point operation, the result is rounded to this many digits, e.g.  $2/3$  is rounded to roughly  $0.666 \cdots 667$ . As you perform a big computation like LU factorization or even a dot product, these errors accumulate and grow, so even if you start with 15 accurate decimal places you may end up with far bigger errors at the end. It turns out that these errors also grow by a factor of the condition number when solving  $Ax = b$ .

At this point, I stopped, and the students filled out their course evaluations for the remaining 15 minutes.