Math 2500: Linear Algebra

Math 2500: Linear Algebra

Contents

Preface

This is a set of class notes designed to guide an inquiry-based course on linear algebra. This is not a complete resource! Rather, this text is meant to accompany the book Introduction to Linear Algebra by Gilbert Strang. Industrious students might also use it for a self-study course.

An important feature of this text is the integration of the Sage Mathematical Software System. Students in this course will learn to use Sage to perform long tedious computations (which linear algebra has in spades) and create visualizations.

Acknowledgements

I thank the Sage community, in particular William Stein, for creating such a wonderful tool and making it open-source.

I also thank Rob Beezer for creating the MathBook project which makes this particular book project available on the web.

Chapter 1

Vectors, the Dot Product, and Matrices

This first chapter introduces the basic objects of linear algebra. You will meet vectors, the dot product, and matrices.

Vectors are a generalization of the concept of number. Where real numbers can help us model the geometry of points on a line, vectors will allow us to model the geometry of a plane, or (three-dimensional) space, or even "spaces" with higher dimensions. We begin by learning about the algebra of vectors, and making connections to the geometry.

The dot product is a funny kind of multiplication. It plays an important role in mathematics because it captures all of the basics of measurement. We shall learn how to use the dot product to measure lengths and angles. By its definition, the dot product is connected with the concept of a linear equation, so it will make frequent appearances in our work.

Matrices are another way to generalize the concept of number. (In fact, they generalize the concept of vector.) We start here by learning about the algebra of matrices. The whole rest of this course will focus on matrices, their uses, and their properties.

A running theme for this course is the use of the Sage mathematical software system. In order to get started, the fourth section of this course is dedicated to getting started with Sage using the SageMathCloud (SMC). You will make an account and run through an introductory workshop with SMC. Also, throughout this workbook you will find little embedded pieces of Sage code. These are implemented using the Sage SingleCell server. Most of these Sage cells are mutable. You can change the content in them and re-evaluate your new Sage code. I encourage you to play with these—it is a good way to learn the basics of Sage.

The fifth and final section of the chapter is a short assignment designed to consolidate learning. You will get a chance to practice your skills and to think more deeply about the concepts you have learned.

1.1 Vectors

The Assignment

- Read section 1.1 of *Strang* (pages 1-7).
- Read the following and complete the exercises below.

Learning Objectives

Before class, a student should be able to:

- Add vectors.
- Multiply a vector by a scalar.
- Compute linear combinations.
- Draw pictures which correspond to the above operations.

At some point, a student should be able to:

- Solve linear combination equations involving unknown coefficients.
- Solve linear combination equations involving unknown vectors.

Some Discussion

Algebraically, a **vector** is a stack of numbers in a set of parentheses or brackets, like this

$$\begin{pmatrix} 2 \\ 7 \\ 9 \end{pmatrix}, \text{ or } \begin{bmatrix} 2 \\ 7 \\ 9 \end{bmatrix}, \text{ or } \begin{pmatrix} 2 & 7 & 9 \end{pmatrix}.$$

The individual numbers are called the **components** or **entries** or **coordinates** of the vector. For example, 7 is the second component of the vectors above.

The first two vectors above are called **column** vectors because they are stacked vertically. The third is called a **row** vector because it is arranged horizontally. For this class, we will always use column vectors, but to save space, we might sometimes write them as row vectors. It is up to you to make the switch. (We will see later how this matters!)

Vectors can take lots of different sizes. The vectors above are all 3-vectors. Here is a 2-vector:

$$\begin{pmatrix} 71 \\ -12 \end{pmatrix}.$$

Here is a 4-vector:

$$\begin{pmatrix} \pi \\ 0 \\ -\pi \\ 1 \end{pmatrix}.$$

The main value in using vectors lies in their standard interpretations. Let's focus on 3-vectors for now. The vector $\left(\begin{smallmatrix} a \\ b \\ c \end{smallmatrix}\right)$ can represent

- A point in space described in the standard three-dimensional rectangular coordinate system with x coordinate equal to a, y-coordinate equal to b and z coordinate equal to c.
- An arrow in space which points from the origin $(0, 0, 0)$ to the point (a, b, c).
- An arrow in space which points from some point (x, y, z) to the point $(x + a, y + b, z + c)$.

Operations on Vectors

There are two operations on vectors which are of utmost importance for linear algebra. (In fact, if your problem has these operations in it, there is a chance you are doing linear algebra already.)

Scalar Multiplication Given a number $\lambda \in \mathbb{R}$ and a vector $v = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$, we form the new vector

$$\lambda v = \begin{pmatrix} \lambda a \\ \lambda b \\ \lambda c \end{pmatrix}.$$

Addition Given a vector $v = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ and a vector $w = \begin{pmatrix} d \\ e \\ f \end{pmatrix}$ of the same size, we form their **sum**

$$v + w = \begin{pmatrix} a+d \\ b+e \\ c+f \end{pmatrix}$$

These operations have "obvious" generalizations to vectors of different sizes. Because things go entry-by-entry, these are often called **coordinate-wise** operations.

Combining these two operations gives us the notion of a *linear combination*. If λ and μ are numbers and v and w are vectors of a common size, then the vector

$$\lambda v + \mu w$$

is a linear combination of v and w.

Sage Instructions

Basic Constructions A vector in Sage is constructed by applying the `vector` command to a list. Lists are entered in square brackets with entries separated by commas, so the typical way to create a vector looks like this:

```
u = vector([1,1,2])
```

Notice that nothing was displayed. Sage just put the vector u into memory. We can ask for it by calling it.

```
u
```

```
(1, 1, 2)
```

Sage defaults to displaying vectors horizontally, which is different from how we normally write them by hand. This is okay. You will get used to it quickly.

Sage knows how to add, multiply by scalars, and form linear combinations, and the notation for it is just as easy as you would expect.

```
v = vector([-1,-1,2])
u + v
```

```
(0, 0, 4)
```

```
pi * u
```

```
(pi, pi, 2*pi)
```

```
3*u + 4*v
```

```
(-1, -1, 14)
```

If you ask Sage to plot a vector, you get this kind of picture:

```
plot(v)
```

And in two dimensions something similar...

```
a = vector([-1,1])
plot(a)
```

If you find that you want a vector to have its tail someplace that is not the origin, use the `arrow` command.

```
plot(arrow([1,1],[2,3], color='red',
           arrowsize=2, width=2),
     figsize=5, aspect_ratio=1)
```

Note that Sage cut off some of this plot! Also, I used some options just to show them off. The `arrow` command works in three dimensions, too.

Interactive Demonstrations This is a Sage "interact." You can use this to explore the idea of linear combinations of 2-vectors.

```
@interact(layout= {'top':[['a','c','e'],['b','d','f'],['l','m']]})
def two_dim_plot(a=input_box(1,width=10),
     b=input_box(2,width=10),c=input_box(2,width=10),
     d=input_box(1,width=10),
                 l=input_box(1,width=10), m=input_box(1,width=10),
                      e=input_box(2,width=10),f=input_box(2,width=10)):
    two_dim = arrow([0,0], [a,b], color ='red') +
         arrow([0,0],[c,d],color='blue')
    two_dim+= arrow([0,0], [l*a,l*b], color='red') +
         arrow([m*c,m*d],[l*a+m*c,l*b+m*d],color='red')
    two_dim+= arrow([0,0], [m*c,m*d], color='blue') +
         arrow([l*a,l*b],[l*a+m*c,l*b+m*d],color='blue')
    two_dim+= point([e,f],size=20,color='black',zorder=2)+
         arrow([l*a,l*b],[l*a+m*c,l*b+m*d],color='blue')
    two_dim+= text('v␣=␣(a,b)', [a-.1,b+.1], color='red') + text('w=(c,d)',
         [c+.1,d-.1],color='purple')
    two_dim+= text('l*v␣+␣m*w', [l*a+m*c+.1, l*b+m*d+.1],color='purple') +
         text('P=(e,f)', [e+.1,f+.1],color='black')
    two_dim+= arrow([0,0],[l*a+m*c,l*b+m*d],color='purple', arrowsize=1,
         width=1)
    two_dim.show(axes=True)
```

This is a different Sage interact. You can use this one to explore linear combinations of 2-vectors.

```
@interact(layout= {'top':[['a','c','e'],['b','d','f'],['l','m']]})
def two_dim_plot(a=input_box(1,width=10),
     b=input_box(2,width=10),c=input_box(2,width=10),
     d=input_box(1,width=10),
                 l=input_box(1,width=10), m=input_box(1,width=10),
                      e=input_box(2,width=10),f=input_box(2,width=10)):
    two_dim = arrow([0,0], [a,b], color ='red') +
         arrow([0,0],[c,d],color='blue')
    two_dim+= arrow([0,0], [l*a,l*b], color='red') +
         arrow([m*c,m*d],[l*a+m*c,l*b+m*d],color='red')
    two_dim+= arrow([0,0], [m*c,m*d], color='blue') +
         arrow([l*a,l*b],[l*a+m*c,l*b+m*d],color='blue')
    two_dim+= point([e,f],size=20,color='black',zorder=2)+
         arrow([l*a,l*b],[l*a+m*c,l*b+m*d],color='blue')
    two_dim+= text('v␣=␣(a,b)', [a-.1,b+.1], color='red') + text('w=(c,d)',
         [c+.1,d-.1],color='purple')
    two_dim+= text('l*v␣+␣m*w', [l*a+m*c+.1, l*b+m*d+.1],color='purple') +
         text('P=(e,f)', [e+.1,f+.1],color='black')
    two_dim+= arrow([0,0],[l*a+m*c,l*b+m*d],color='purple', arrowsize=1,
         width=1)
    two_dim.show(axes=True)
```

Task 1.1. Find an example of numbers λ and μ so that

$$\lambda \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \mu \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

or describe why no such example can exist.

Task 1.2. Find a vector $b = \left(\begin{smallmatrix} b_1 \\ b_2 \end{smallmatrix}\right)$ so that

$$\begin{pmatrix} 2 \\ 7 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 10 \\ -3 \end{pmatrix}$$

or describe why no such example can exist.

Task 1.3. Find a vector $b = \left(\begin{smallmatrix} b_1 \\ b_2 \end{smallmatrix}\right)$ so that this equation has at least one solution λ

$$\begin{pmatrix} 1 \\ -2 \end{pmatrix} + \lambda \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

or describe why no such example can exist.

Task 1.4. Give examples of numbers a and b such that

$$a \begin{pmatrix} 2 \\ 1 \end{pmatrix} + b \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \\ 5 \end{pmatrix}$$

or explain why no such numbers exist.

In the situations like the last exercise, the pair of numbers a, b is called a **solution** to the equation.

Task 1.5. Give an example of a vector $X = \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$ so that the equation

$$a \begin{pmatrix} 2 \\ 1 \end{pmatrix} + bX = \begin{pmatrix} 7 \\ 5 \end{pmatrix}$$

has no solution (a, b), or explain why no such example exists.

Task 1.6. Give an example of a number λ so that

$$\lambda \begin{pmatrix} 7 \\ -1 \\ 2 \end{pmatrix} + 3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 49 \\ -7 \\ 20 \end{pmatrix}$$

or explain why no such number exists.

Task 1.7. Give an example of numbers λ and μ which are a solution to the equation

$$\lambda \begin{pmatrix} 7 \\ -1 \\ 2 \end{pmatrix} + \mu \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 49 \\ -7 \\ 20 \end{pmatrix}$$

or explain why no such solution exists.

Task 1.8. Give an example of a vector $w = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ so that the equation

$$a \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

has no solution (a, b), or explain why no such vector exists.

Task 1.9. Give an example of a vector $w = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ so that the equation

$$a \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

has exactly one solution (a, b), or explain why no such vector exists.

Task 1.10. Give an example of a vector $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ such that the equation

$$a \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + c \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

has no solutions (a, b, c), or explain why no such vector exists.

Task 1.11. Give an example of a vector $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ such that the equation

$$a \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + c \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

has exactly one solution, or explain why no such vector exists.

Task 1.12. Give an example of a vector $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ such that the equation

$$a \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} + b \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + c \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ 7 \end{pmatrix}$$

has no solutions, or explain why no such vector exists.

1.2 The Dot Product

The Assignment

- Read section 1.2 of *Strang*
- Read the following and complete the exercises below.

Learning Objectives

Before class, a student should be able to

- Compute the dot product of two given vectors.
- Compute the length of a given vector.
- Normalize a given vector.
- Recognize that $u \cdot v = 0$ is the same as "u and v are orthogonal."
- Compute the angle between two given vectors using the cosine formula.

At some point, a student should be able to

- Interpret the statements $u \cdot v < 0$ and $u \cdot v > 0$ geometrically.
- Pass back and forth between linear equations and equations involving dot products.
- Make pictures of level sets of the dot product operation.

Discussion

The dot product is a wonderful tool for encoding the geometry of Euclidean space, but it can be a bit mysterious at first. As Strang shows, it somehow holds all of the information you need to measure lengths and angles.

What does this weird thing have to do with linear algebra? A dot product with a "variable vector" is a way of writing a linear equation. For example,

$$\begin{pmatrix} 7 \\ 3 \\ -2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 7x + 3y - 2z.$$

Sometimes this will allow us to connect linear algebra to geometry, and use geometric thinking to answer algebraic questions.

Sage and the dot product

Basic Commands

Sage has a built-in dot product command `u.dot_product(v)`. This will return the dot product of the vectors u and v.

```
u = vector([1,1,1]); v = vector([-1,0,1])
u.dot_product(v)
```

```
0
```

It also has a built-in command for computing lengths. Here sage uses the synonym "norm": `u.norm()`. Of course, you can also call this like a function instead of like a method: `norm(u)`.

```
u.norm(), v.norm()
```

```
(sqrt(3), sqrt(2))
```

There is no built-in command for angles. You just have to compute them using the cosine formula, like below. (I will break up the computation, but it is easy to do it all with one line.)

```
num = u.dot_product(v)
den = u.norm() * v.norm()
angle = arccos(num / den)
angle
```

```
1/2*pi
```

Of course, Sage's `arccos` command returns a result in *radians*. To switch to degrees, you must convert.

```
angle*180/pi
```

```
90
```

Often, it is helpful to normalize a vector. You can do this with the `normalized` method like this:

```
u.normalized()
```

```
(1/3*sqrt(3), 1/3*sqrt(3), 1/3*sqrt(3))
```

Sage Interacts

Interacts would go here...

Exercises about the Dot Product

Task 1.13. What shape is the set of solutions $\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$ to the equation

$$\begin{pmatrix} 3 \\ 7 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = 5?$$

That is, if we look at all possible vectors $\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$ which make the equation true, what shape does this make in the plane?

What happens if we change the vector $\left(\begin{smallmatrix} 3 \\ 7 \end{smallmatrix}\right)$ to some other vector? What happens if we change the number 5 to some other number?

What happens if instead of 2-vectors, we use 3-vectors?

Task 1.14. Find an example of two 2-vectors v and w so that $\left(\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right) \cdot v = 0$ and $\left(\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right) \cdot w = 0$, or explain why such an example is not possible.

Task 1.15. Let $v = \left(\begin{smallmatrix} 3 \\ -1 \end{smallmatrix}\right)$. Find an example of a pair of vectors u and w such that $v \cdot u < 0$ and $v \cdot w < 0$ and $w \cdot u = 0$, or explain why no such pair of vectors can exist.

Task 1.16. Find an example of three 2-vectors u, v, and w so that $u \cdot v < 0$ and $u \cdot w < 0$ and $v \cdot w < 0$, or explain why no such example exists.

Task 1.17. Find an example of a number c so that

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = c$$

has the vector $\left(\begin{smallmatrix} 4 \\ 7 \end{smallmatrix}\right)$ as a solution, or explain why no such number exists.

Task 1.18. Let $v = \left(\begin{smallmatrix}2\\1\end{smallmatrix}\right)$ and $w = \left(\begin{smallmatrix}-3\\4\end{smallmatrix}\right)$. Find an example of a number c so that

$$v \cdot \begin{pmatrix}1\\-1\end{pmatrix} = c \quad \text{and} \quad w \cdot \begin{pmatrix}1\\-1\end{pmatrix} = c,$$

or explain why this is not possible.

Task 1.19. Let $P = \left(\begin{smallmatrix}-3\\4\end{smallmatrix}\right)$. Find an example of numbers c and d so that

$$\begin{pmatrix}2\\-1\end{pmatrix} \cdot P = c \quad \text{and} \quad \begin{pmatrix}1\\-1\end{pmatrix} \cdot P = d,$$

or explain why no such example is possible.

Now we move to three dimensions!

Task 1.20. Let $V = \left(\begin{smallmatrix}1\\1\\1\end{smallmatrix}\right)$. Find a unit vector of the form $X = \left(\begin{smallmatrix}x\\y\\0\end{smallmatrix}\right)$ so that $V \cdot X = \sqrt{2}$, or explain why no such vector exists.

Task 1.21. Find an example of numbers c, d, and e so that there is no solution vector $X = \left(\begin{smallmatrix}x\\y\\z\end{smallmatrix}\right)$ which simultaneously satisfies the three equations

$$\begin{pmatrix}1\\1\\1\end{pmatrix} \cdot X = c, \qquad \begin{pmatrix}2\\2\\2\end{pmatrix} \cdot X = d, \qquad \begin{pmatrix}0\\0\\1\end{pmatrix} \cdot X = e,$$

or explain why no such numbers exist.

Task 1.22. Find an example of numbers c, d, and e so that there is no solution vector $X = \left(\begin{smallmatrix}x\\y\\z\end{smallmatrix}\right)$ which simultaneously satisfies the three equations

$$\begin{pmatrix}1\\1\\1\end{pmatrix} \cdot X = c, \qquad \begin{pmatrix}0\\1\\1\end{pmatrix} \cdot X = d, \qquad \begin{pmatrix}0\\0\\1\end{pmatrix} \cdot X = e,$$

or explain why no such numbers exist.

1.3 Matrices

The Assignment

- Read *Strang* section 1.3 (pages 22-27).
- Read the following and complete the exercises below.

Learning Goals

Before class starts, a student should be able to:

- multiply a matrix times a vector
  - as a linear combination of columns
  - as a set of dot products, row times column
- translate back and forth between our three representations
  - a system of linear equations,
  - a linear combination of vectors equation, and
  - a matrix equation $Ax = b$.
- Correctly identify the rows and columns of a matrix
- describe what is meant by a lower triangular matrix

At some point, as student should be comfortable with these concepts, which get a very brief informal introduction in this section:

- linear dependence and linear independence
- the inverse of a matrix

Discussion

A **matrix** is a two-dimensional array of numbers like this:

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}.$$

Sometimes it helps to think of a matrix as a collection of its **rows** which are read across:

$$M = \begin{pmatrix} \longrightarrow \\ \longrightarrow \end{pmatrix}$$

and sometimes it helps to think of a matrix as a collection of its **columns** which are read down:

$$M = \begin{pmatrix} \downarrow & \downarrow \end{pmatrix}.$$

It is often more clear to describe a matrix by giving the sizes of its rows and columns. An m by n matrix is one having m rows and n columns. It is really easy to get these reversed, so be careful. For example, this is a 2×3 matrix, because it has two rows and three columns:

$$B = \begin{pmatrix} 1 & 1 & 2 \\ 3 & 5 & 8 \end{pmatrix}$$

A matrix is called a **square** matrix when the number of rows and the number of columns is equal. The matrix A that I wrote down above is square because it is a 2×2 matrix.

Multiplying Matrices and Vectors

It is possible to multiply a matrix by a vector like this:

$$\begin{pmatrix} 1 & 1 & 2 \\ 3 & 5 & 8 \end{pmatrix} \begin{pmatrix} 13 \\ 21 \\ 34 \end{pmatrix} = \begin{pmatrix} 102 \\ 416 \end{pmatrix}$$

For this to work, it is absolutely crucial that the sizes match up properly. If the matrix is m by n, then the vector must have size n. In the above example $m = 2$ and $n = 3$.

Later, we shall see that the word "multiplication" is not really the best choice here. It is better to think of the matrix as "acting on" the vector and turning it into a new vector. For now, the word multiplication will serve.

How exactly does one define this matrix–vector multiplication?

Linear Combination of Columns Approach The first way to perform the matrix-vector multiplication is to think of the vector as holding some coefficients for forming a linear combination of the columns of the matrix. In our example, it looks like this:

$$\begin{pmatrix} 1 & 1 & 2 \\ 3 & 5 & 8 \end{pmatrix} \begin{pmatrix} 13 \\ 21 \\ 34 \end{pmatrix} = 13 \begin{pmatrix} 1 \\ 3 \end{pmatrix} + 21 \begin{pmatrix} 1 \\ 5 \end{pmatrix} + 34 \begin{pmatrix} 2 \\ 8 \end{pmatrix} = \begin{pmatrix} 102 \\ 416 \end{pmatrix}$$

Dot Products with the Rows Approach The second way is to think of the matrix as a bundle of vectors lying along the rows of the matrix, and use the dot product. In our example above, this means that we consider the vectors

$$r_1 = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}, \quad r_2 = \begin{pmatrix} 3 \\ 5 \\ 8 \end{pmatrix}, \text{ and } v = \begin{pmatrix} 13 \\ 21 \\ 34 \end{pmatrix}$$

(notice I've rewritten the rows as columns) and then perform this kind of operation:

$$\begin{pmatrix} 1 & 1 & 2 \\ 3 & 5 & 8 \end{pmatrix} \begin{pmatrix} 13 \\ 21 \\ 34 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} v = \begin{pmatrix} r_1 \cdot v \\ r_2 \cdot v \end{pmatrix} = \begin{pmatrix} 102 \\ 416 \end{pmatrix}.$$

Two important remarks:

- Note that these operations only make sense if the sizes match up properly.
- Note that the two versions of the operation give you the same results.

Matrix Equations

There are many situations in linear algebra that can be rewritten in the form of an equation that looks like this:

$$Av = b$$

where A is a matrix, and v and b are vectors. The interesting case is when we know A and b, but we want to find the unknown v. We will call this a **matrix-vector equation**.

Let's consider the case where you are given some **square** matrix A. Sometimes one can find another matrix B so that no matter what vector b is chosen in the matrix-vector equation above, the solution vector takes the form $v = Bb$. When this happens, we say that A is **invertible** and call B the **inverse** of A. It is common to use the notation A^{-1} in place of B. This is a wonderful situation to be in! Eventually, we will want to figure out some test for when a given matrix is invertible, and find some ways to compute the inverse.

A Note about Vectors

This reading also has a brief introduction to the idea of a set of vectors being **linearly depedent** or **linearly independent**. Strang is coy about the precise definition, so here it is:

A set of vectors $v_1, v_2, \ldots, v_n$ is called **linearly depdendent** when there is some choice of numbers $a_1, a_2, \ldots, a_n$ which are not all zero so that the linear combination

$$a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = 0$$

A set of vectors which is not linearly dependent is called **linearly independent**.

This is a little funny the first time you read it. Note that for any set of vectors, you can make a linear combination of those vectors come out as 0. Simply choose all of the coefficients to be zero. But that is so easy to do we call it **trivial**. What the definition is asking is that we find a *nontrival linear combination of the vectors to make zero.*

Sage and Matrices

Sage has many useful commands for working with linear algebra, and given the central role played by matrices in this subject, there are lots of things Sage can do with matrices. We'll focus here on just basic construction and matrix–vector multiplication.

The matrix construction command

The command to construct a matrix is pretty straightforward. One types `matrix(r, c, [list of entries` where `r` is the number of rows and `c` is the number of columns. The entries should be read across the rows starting with the top row.

```
A = matrix(2,3, [1,2,3,5,8,13]); A
```

```
[ 1  2  3]
[ 5  8 13]
```

If you wish, you can structure that list of entries to be a list of lists, where each sublist is a row in your matrix.

```
B = matrix(2,3, [[1,2,3], [5,8,13]]); B
```

```
[ 1  2   3]
[ 5  8  13]
```

Every once in a while, it might matter to you what kinds of numbers you put into the matrix. Sage will let you specify them by putting in an optional argument like this: `matrix(number type, r, c, [list of entries])`

```
C = matrix(ZZ, 2, 2, [2,1,1,1])
C # the best matrix
```

```
[2 1]
[1 1]
```

The notation `ZZ` means "integers." There are other sets of numbers here:

- `QQ` the rational numbers (with exact arithmetic)
- `RR` the real numbers (with computer precision arithmetic)
- `CC` the complex numbers
- `AA` the set of all algebraic numbers, that is, all of the numbers that are roots of some polynomial with integer coefficients

You can find out what kind of entries a matrix thinks it has by calling the `.parent()` method on it.

```
A.parent()
# this should say something about the integers
```

```
Full MatrixSpace of 2 by 3 dense matrices over Integer Ring
```

```
D = matrix(QQ, 3,3, [[1,0,1],[2/3, 1, 0],[0,0,9/5]])
# this should say something about the rationals
D.parent()
```

```
[  1   0   1]
[2/3   1   0]
[  0   0 9/5]
Full MatrixSpace of 3 by 3 dense matrices over Rational Field
```

Building a matrix from rows or columns

It is possible to build a matrix by bundling together a bunch of vectors, too. Let's start with an example made using rows.

```
v1 = vector([2,1]); v2= vector([3,4])
# construct E with rows v1 and v2, then display
E = matrix([ v1, v2]); E
```

```
[2 1]
[3 4]
```

Sage prefers rows. I wish it were the other way, but I am sure there is a good reason it prefers rows. If you want to make a matrix whose columns are the vectors `v1` and `v2`, you can use the `transpose` method. We'll talk more about the operation of transpose later, but it basically "switches rows for columns and vice versa."

```
F = matrix([v1,v2]).transpose(); F
```

```
[2 3]
[1 4]
```

Matrix action on vectors

Of course, Sage knows how to perform the action of a matrix on a vector.

```
C; v1
```

```
[2 1]
[1 1]
(2, 1)
```

```
C*v1
```

```
(5, 3)
```

And if you get the sizes wrong, it will return an error.

```
A; v1
```

```
[ 1  2  3]
[ 5  8 13]
(2, 1)
```

```
A*v1
```

```
Error in lines 1-1
...
TypeError: unsupported operand parent(s) for '*': 'Full MatrixSpace of 
    2 by 3 dense matrices over Integer Ring' and 'Ambient free module 
    of rank 2 over the principal ideal domain Integer Ring'
```

If you really need it, Sage can tell you about inverses.

```
A.is_invertible()
```

```
False
```

```
C.is_invertible()
```

```
True
```

```
C.inverse()
```

```
[ 1 -1]
[-1  2]
```

Exercises

Task 1.23. Make an example of a matrix $\left(\begin{smallmatrix} 1 & \bullet \\ -1 & \bullet \end{smallmatrix}\right)$ so that the equation

$$\begin{pmatrix} 1 & \bullet \\ -1 & \bullet \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

has exactly one solution, or explain why this is not possible.

Interpret this as a statement about 2-vectors and draw the picture which corresponds.

Task 1.24. Make an example of a matrix $\left(\begin{smallmatrix} 4 & 8 & \bullet \\ 3 & 6 & \bullet \\ 1 & 2 & \bullet \end{smallmatrix}\right)$ so that the equation

$$\begin{pmatrix} 4 & 8 & \bullet \\ 3 & 6 & \bullet \\ 1 & 2 & \bullet \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 8 \\ 6 \\ 2 \end{pmatrix}$$

has exactly one solution, or explain why this is not possible.

Interpret this as a statment about 3-vectors and draw the picture which corresponds.

Task 1.25. Make an example of a matrix $\left(\begin{smallmatrix} 2 & -1 \\ \bullet & \bullet \end{smallmatrix}\right)$ so that the equation

$$\begin{pmatrix} 2 & -1 \\ \bullet & \bullet \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 7 \\ 3 \end{pmatrix}$$

has exactly one solution, or explain why this is not possible.

Interpret this as a statement about a pair of lines in the plane and draw the picture which corresponds.

Task 1.26. Make an example of a matrix $\left(\begin{smallmatrix}1&0&1\\1&1&3\\\bullet&\bullet&\bullet\end{smallmatrix}\right)$ so that the equation

$$\begin{pmatrix}1&0&1\\1&1&3\\\bullet&\bullet&\bullet\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}1\\1\\1\end{pmatrix}$$

has no solutions, or explain why this is not possible.

Interpret this as a statement about a planes in space and draw the picture which corresponds.

Task 1.27. Find a triple of numbers x, y, and z so that the linear combination

$$x\begin{pmatrix}1\\2\\3\end{pmatrix}+y\begin{pmatrix}4\\5\\6\end{pmatrix}+z\begin{pmatrix}7\\8\\9\end{pmatrix}$$

yields the zero vector, or explain why this is not possible.

Rewrite the above as an equation which involves a matrix.

Plot the three vectors and describe the geometry of the situation.

Task 1.28. The vectors

$$r_1=\begin{pmatrix}1\\4\\7\end{pmatrix},\qquad r_2=\begin{pmatrix}2\\5\\8\end{pmatrix},\qquad\text{and}\qquad r_3=\begin{pmatrix}3\\6\\9\end{pmatrix}$$

are linearly dependent because they lie in a common plane (through the origin). Find a normal vector to this plane.

Since the vectors are linearly dependent, there must be (infinitely) many choices of scalars x, y, and z so that $xr_1+yr_2+zr_3=0$. Find two sets of such numbers.

Task 1.29. Consider the equation

$$\begin{pmatrix}2&1\\1&1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}b_1\\b_2\end{pmatrix}.$$

We are interested in being able to solve this for x and y for any given choice of the numbers b_1 and b_2. Figure out a way to do this by writing x and y in terms of b_1 and b_2.

Rewrite your solution in the form

$$\begin{pmatrix}x\\y\end{pmatrix}=b_1\begin{pmatrix}\bullet\\\bullet\end{pmatrix}+b_2\begin{pmatrix}\bullet\\\bullet\end{pmatrix}.$$

How is this related to the inverse of the matrix $A=\left(\begin{smallmatrix}2&1\\1&1\end{smallmatrix}\right)$?

Task 1.30. Find an example of a number c and a vector $\left(\begin{smallmatrix}b_1\\b_2\end{smallmatrix}\right)$ so that the equation

$$\begin{pmatrix}3&51\\c&17\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}b_1\\b_2\end{pmatrix}$$

does not have a solution, or explain why no such example exists.

Explain your solution in terms of

- lines in the plane,
- 2-vectors and linear combinations, and
- invertibility of a matrix.

1.4 Getting Started with Sage

The Assignment

It it is time to get comfortable with the basics of using Sage. Begin by setting up an account with SageMathCloud.

- Be sure that you have login access to computers in student labs across campus. Before class, drop by a campus lab and make sure your credentials work properly and you can sign in to a machine. For our tutorial session we will use one of the labs on the first floor of Wright Hall, so one of those would make a good choice.
- Go to the course web site and follow the links to the Sage Intro workshop.
- Complete the first two steps of the workshop set up process:
    1. Start: where you make an account at SageMathCloud.
    2. Get: where you create your first "project" and populate it with files for a tutorial.

Learning Objectives

At the end of this assignment, a student should have an account at SageMathCloud and should be able to log into the service without trouble. The student will also have a first project with some files.

1.5 Going Further with the Basic Objects

The Assignment

- Go back through the exercises in this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in Chapter One: Basic Objects. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

Chapter 2

Linear Equations

This chapter is dedicated to the first major problem of linear algebra: solving systems of linear equations. Restated as a set of questions, we will consider these.

- What is the set of solutions to a given system of linear equations?
- When does a given system have solution, and when does it not?
- If there is a solution, how many solutions are there?
- What ways do we have of describing the collection of solutions?
- Is there a computationally effective way to find those solutions?

Though we begin with the first question, we answer the last one first. As we explore the process of finding solutions, we will start to build the tools we need to finish answering the other questions in a later chapter. In this chapter, we aim to get as complete understanding as we can for at least a special case: **square systems**.

We will begin with a study of the two ways that vectors let us make pictures of systems of linear equations. Then we take up a basic process for finding solutions, where matrices will appear as a convenient notational device. But as we dig a little further, matrices will become interesting in their own way, so we will study those. But what we study will relate back to the fundamental issue of solving systems of linear equations.

This chapter has two short review sections in it. One just after we learn about elimination, and another at the end of the chapter.

2.1 Three Geometric Models

The Assignment

- Read section 2.1 of *Strang* (pages 31-40).
- Read the following and complete the exercises below

Learning Goals

Before class, a student should be able to:

- Translate back and forth between the three algebraic representations:
  - A system of linear equations.

  - An equation involving a linear combination of vectors.
  - A matrix equation.

- Can write down the $n \times n$ **identity matrix**.

Sometime in the near future, a student should be able to:

- Given a system, interpret and plot the “row picture”.
- Given a system, interpret and plot the “column picture”.
- Use a matrix as a model of a **transformation**, including stating the **domain** and the **range**.

Discussion

Now we have a little experience with vectors and related things, it is time to be aware of what we have done so we can use it as a foundation for future work. So far, we have talked about two geometric interpretations for a system of linear equations, the **row picture** and the **column picture**.

Does the following picture make sense to you?

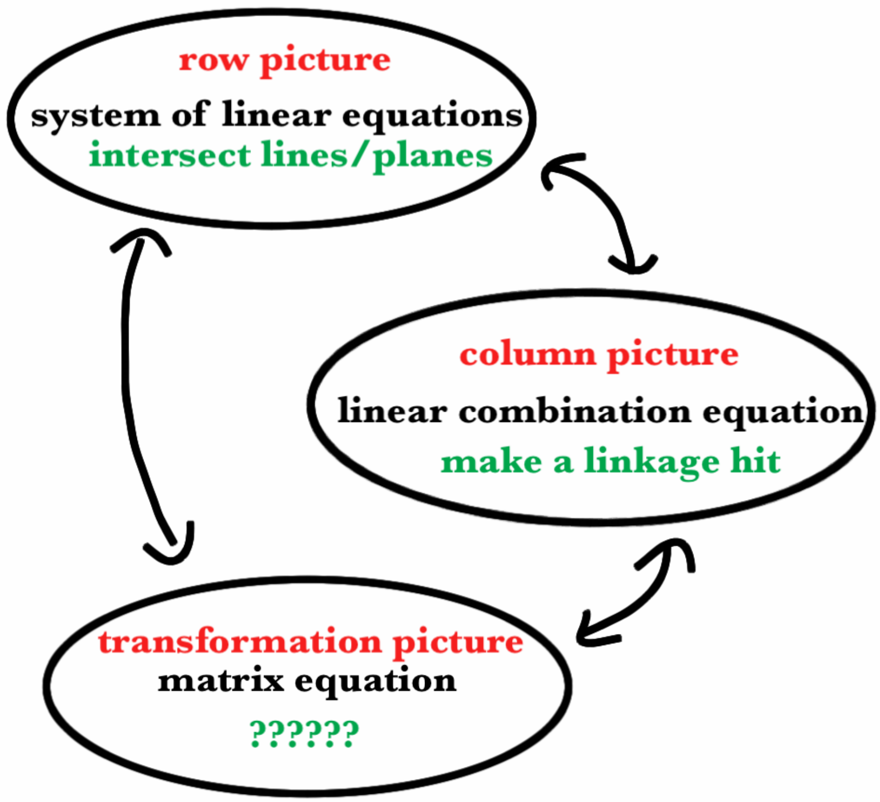

Figure 2.1: The three geometric models of linear algebra

A deep understanding of linear algebra will involve a level of comfort with each of these three views of the subject in the diagram, and also the ability to pass back and forth between them.

The Transformational View

We have seen that matrices can be made to "act upon" vectors by a kind of multiplication. In particular, if A is an $m \times n$ matrix, then A can be multiplied (on the left) with a column vector of size n, and the result is a column vector of size m.

This makes A into a kind of **function**. (We will use the synonyms **mapping** or **transformation**, too.) For every vector v of size n, the matrix A allows us to compute a new vector $T_A(v) = Av$ of size m. This is the basic example of what we will eventually call a **linear transformation**.

$$\mathbb{R}^m \xrightarrow{T_A} \mathbb{R}^n$$

$$v \longmapsto Av.$$

One of our long term goals is to find a way to think about the geometry of linear algebra from this viewpoint, too.

Sage and Plotting for Linear Algebra

There are a few new Sage commands that might be useful here. We have already seen how to take linear equations and turn them into vectors and then turn the vector equation into a matrix equation. But Sage can help us move in the other direction, too.

The keys are commands to pull out the rows and columns from a given matrix. Let's start with a simple situation where the matrix equation is

$$\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x & y \end{pmatrix} = \begin{pmatrix} 3 & 4 \end{pmatrix}.$$

```
A = matrix(QQ, 2,2, [2,1,1,1]); A
```

```
[2 1]
[1 1]
```

```
x, y = var('x␣y')
X = vector([x,y]); X
```

```
(x, y)
```

```
b = vector([3,4]); b
```

```
(3, 4)
```

To get Sage to pull out the columns, we can use the `.columns()` method. If we want just one column, we can use the `.column()` method, but then we have to remember to specify which column we want.

```
A.columns() # this will return a list
```

```
[(2, 1), (1, )]
```

```
A.column(1)
```

```
(1, 1)
```

Big important note: Sage always numbers lists starting with zero. so the first element of every list is the 0 entry, and the second element is the 1 entry.

Now it is possible to make Sage do things like this:

```
column_plot = plot(A.column(0), color='red')
column_plot+= plot(A.column(1), color='blue')
column_plot+= plot(b, color='purple')
show(column_plot, figsize=5, aspect_ratio=1)
```

This is an example of the a column picture.

One can also pull out the rows with corresponding row methods. And if you recall the way that matrix multiplication works if you think of rows, you can make a row picture.

```
A.rows()
```

```
[(2, 1), (1, 1)]
```

```
expr1 = A.row(0).dot_product(X) == b[0]
expr2 = A.row(1).dot_product(X) == b[1]

print expr1
print expr2
```

```
2*x + y == 3
x + y == 4
```

And now the picture:

```
row_plot = implicit_plot(expr1, [x,-5,5], [y,-1,9], color='blue')
row_plot+= implicit_plot(expr2, [x,-5,5], [y,-1,9], color='red')
show(row_plot, axes=True)
```

Exercises

Task 2.1. Make an example of a system of linear equations which some students might find challenging to change into an equation involving a linear combination. Explain what the challenge is and how you can think clearly to overcome it.

Task 2.2. Make an example of a linear combination equation which some students might find challenging to change into a system of linear equations. Explain what the challenge is and how you can think clearly to overcome it.

Task 2.3. Consider the matrix equation

$$\begin{pmatrix} 1 & 2 & 4 \\ 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}.$$

1. Draw a diagram representing the row picture of this matrix equation.
2. Draw a diagram representing the column picture of this matrix equation.

Task 2.4. Make an example of a system of linear equations so that the corresponding column picture is about linear combinations of four 2-vecs becoming the zero vector.

Task 2.5. Find a linear combination equation so that the corresponding system of linear equations corresponds to finding the intersection of three lines in the plane.

Task 2.6. Find an example of a vector b so that the equation

$$\begin{pmatrix} -1 & 2 \\ 5 & -9 \end{pmatrix} v = b$$

has no solution v, or explain why it is impossible to find such an example.

Task 2.7. In each of the below, find an example of a matrix B which has the described effect.

1. $B\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ x \end{pmatrix}$
2. Rotates vectors through $45°$ counter-clockwise.
3. Reflects vectors across the $y$axis.
4. $B\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+y \\ y \end{pmatrix}$.

2.2 Solving Systems

The Assignment

- Read section 2.2 of Strang (pages 45-51).
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to do the following things.

- Clearly state and use the following vocabulary words: pivot, multiplier, triangular matrix, back substitution, singular, non-singular

Sometime after class, a student should be able to the following things.

- Perform elimination to put a system of linear equations into triangular form.
- Solve small systems by hand.
- Explain the two failure modes for elimination, and describe which leads to no solutions, and which leads to infinitely many solutions.
- Solve larger systems with the help of a computer algebra package (Sage).

Discussion: Elimination for Solving Systems of Linear Equations

Now we begin the process of learning how to solve a system of linear equations systematically through a process called **elimination**.

Some terminology

A typical system looks something like this:

$$\left\{\begin{array}{rcrcrcr} 3x_1 & + & 2x_2 & - & \pi x_3 & = & 0 \\ -4x_1 & - & 33x_2 & + & x_3 & = & 12 \end{array}\right.$$

This situation is *two* equations in *three* unknowns. The unknowns here are the three numbers x_1, x_2 and x_3 for which we search. Usually, we bundle the numbers together as a vector (x_1, x_2, x_3). If we can find a vector which makes all of the equations true simultaneously, we call that vector a **solution**.

Keep in mind that the process involves eliminating instances of the variable below **pivots**. Strang describes the process pretty well, and gives good examples. What Strang describes in this section is sometimes called **the forward pass** elimination.

Watch out for situations which are **singular** in that they have fewer pivots than unknowns. A system is called **non-singular** if it has as many pivots as unknowns.

Keeping track of things

Playing with all of the equations is nice, but all that really matters is the collection of coefficients, and the numbers on the right hand sides of the equal signs. Experienced solvers get tired of copying notation from line to line in a computation, so they only keep track of the matrix of coefficients, **augmented** by the vector on the right-hand side. In the example above, that augmented matrix is

$$\left(\begin{array}{ccc|c} 3 & 2 & -\pi & 0 \\ -4 & -33 & 1 & 12 \end{array}\right)$$

All of the row operations can be performed on just this augmented matrix, without losing any of the essential information.

Sage and Row Operations

The process of elimination for systems of equations involves performing operations on the equations. When translated to matrix form, it involves operations on the rows of the coefficient matrix. The corresponding matrix methods come in two types.

The first type of method modifies the matrix "in place", which means that it *Changes the input matrix*.

- `A.rescale_row(r, num)` multiplies row `r` by the factor of `num`.
- `A.swap_rows(r1, r2)` switches the places of rows `r1` and `r2`.
- `A.add_multiple_of_row(target, useful, num)`. This adds `num` times row `useful` to row `target`.

Throughout, please remember that Sage uses 0-based indexing! So the rows are labeled `0, 1, 2, ...`

```
A = matrix(QQ, 3,3, [0,2,4, 1,1,5, 6,2,5]); A
```

```
[0 2 4]
[1 1 5]
[6 2 5]
```

```
A.swap_rows(0,1); A
```

```
[1 1 5]
[0 2 4]
[6 2 5]
```

```
A.add_multiple_of_row(2,0,-6)
A # this should add -6 times row 0 to row 2
```

```
[  1   1   5]
[  0   2   4]
[  0  -4 -25]
```

```
A.rescale_row(1,1/2); A
```

```
[  1   1   5]
[  0   1   2]
[  0  -4 -25]
```

```
A.add_multiple_of_row(2,1,4)
A # this should add 4 times row 2 to row 2
```

```
[  1   1   5]
[  0   1   2]
[  0   0 -17]
```

```
A.rescale_row(2,-1/17); A
```

```
[1 1 5]
[0 1 2]
[0 0 1]
```

This just did the whole process of **forward pass elimination**. (Well, we did a bit more than Strang would. He wouldn't rescale the rows.)

Sometimes you do not want to change the matrix `A`. If instead, you want to leave `A` alone, you can use these methods, which return a new object and do not change `A`.

- `A.with_rescaled_row(r, num)`
- `A.with_swapped_rows(r1, r2)`
- `A.with_added_multiple_of_row(t, u, num)`

Let's do the same operations as above, but without changing `A`. This will mean making a bunch of new matrices. In fact, let's also change the name of our matrix to `B`

```
B = matrix(QQ, 3,3, [0,2,4, 1,1,5, 6,2,5]); B
```

```
[0 2 4]
[1 1 5]
[6 2 5]
```

```
B1 = B.with_swapped_rows(0,1); B1
```

```
[1 1 5]
[0 2 4]
[6 2 5]
```

```
B2 = B1.with_added_multiple_of_row(2,0,-6)
B2 # this should add -6 times row 0 to row 2
```

```
[  1   1   5]
[  0   2   4]
[  0  -4 -25]
```

```
B3 = B2.with_rescaled_row(1,1/2); B3
```

```
[  1   1   5]
[  0   1   2]
[  0  -4 -25]
```

```
B4 = B3.with_added_multiple_of_row(2,1,4)
B4 # this should add 4 times row 2 to row 2
```

```
[  1   1   5]
[  0   1   2]
[  0   0 -17]
```

```
B5 = B4.with_rescaled_row(2,-1/17); B5
```

```
[1 1 5]
[0 1 2]
[0 0 1]
```

This second option has some advantages. At any point, you can revise your work, because the original matrix is still in memory, and so are all of the intermediate steps. Let's display all six of the matrices at once to see that they all still exist.

```
B, B1, B2, B3, B4, B5
```

Exercises

Task 2.8. Use the elimination method to transform this system into an easier one. (Can you make it triangular?) Circle the pivots in the final result.

$$\begin{cases} 2x + 3y + z = 8 \\ 4x + 7y + 5z = 20 \\ \quad\; - 2y + 2z = 0 \end{cases}$$

What two operations do you use to do this efficiently? Now use back substitution to solve the system.

Task 2.9. *(Sage Exercise)*: Because the last system can be transformed in two operations, there are three equivalent systems generated through the process (the original, the intermediate, and the final).

Make row picture plots for each of the three systems. [Hint: Sage] How do the operations transform the pictures?

Task 2.10. Suppose that a system of three equations in three unknowns has two solutions (a, b, c) and (A, B, C). Explain why the system must have other solutions than these two. Describe clearly two other solutions in terms of a, b, c, A, B, C.

Task 2.11. Find three examples of numbers a so that elimination will fail to give three pivots for this coefficient matrix:

$$A = \begin{pmatrix} a & 2 & 3 \\ a & a & 4 \\ a & a & a \end{pmatrix}$$

Task 2.12. How many ways can two lines in the plane meet? Make examples to represent as many *qualitatively different* situations as you can.

Task 2.13. Complete the following to make an example of a system of two equations in two unknowns which is singular but still has a solution, or explain why no such example exists.

$$\begin{cases} 2x + 3y = 1 \\ \bullet x + \bullet y = \bullet \end{cases}$$

Task 2.14. Complete the following to a system of three equations in three unknowns which is singular and does not have a solution, or explain why no such example exists.

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

Task 2.15. Complete the following to a system of three equations in three unknowns which is singular but still has a solution, or explain why no such example exists.

$$\begin{cases} x + y + z = 1 \\ 2x + y + 2z = 0 \\ \bullet x + \bullet y + \bullet z = \bullet \end{cases}$$

Task 2.16. How many ways can three planes in three dimensional space meet? Make examples to represent as many *qualitatively different* situations as you can.

2.3 Elimination using Matrices

The Assignment

- Read section 2.3 of *Strang*.
- Read the discussion below and work out the exercises.

Learning Goals

Before class, a student should be able to:

- Translate a system of linear equations into the form of an augmented matrix and back.
- Perform the forward pass elimination process to an augmented matrix.
- Multiply a pair of square matrices having the same size.
- Identify the matrix which performs the operation "add a multiple of row i to row j."
- Identify the matrix which performs the operation "swap the places of row i and row j."

Some time after class, a student should be able to:

- Use the steps from a forward pass elimination step to write a correct equation of the form
$$E_\bullet E_\bullet \cdots E_\bullet (A \mid b) = (U \mid b')$$
where U is an upper triangular matrix.

Discussion: Elimnation and Using Matrices as "Transformations"

Let us focus (for now) on square systems of equations, where the number of unknowns is equal to the number of equations.

The Four Ways to Write a System

Recall that there are three equivalent ways to write the typical linear algebra problem: (1) a system of linear equations to be solved simultaneously, (2) an equation expressing some linear combination of vectors with unknown coefficients as equal to another vector, and (3) a matrix equation.

Here is an example: This system of linear equations

$$\begin{cases}   3y + 2z = 8 \\ x - y + z = -1 \\ 3x + 2y + 3z = 1 \end{cases}$$

is equivalent to this equation using a linear combination of vectors

$$x\begin{pmatrix}0\\1\\3\end{pmatrix} + y\begin{pmatrix}3\\-1\\2\end{pmatrix} + z\begin{pmatrix}2\\1\\3\end{pmatrix} = \begin{pmatrix}8\\-1\\1\end{pmatrix}$$

and both of those are equivalent to this matrix equation

$$\begin{pmatrix}0 & 3 & 2\\1 & -1 & 1\\3 & 2 & 3\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}8\\-1\\1\end{pmatrix}.$$

Each of these viewpoints has its advantages when talking about the geometry of linear algebra. But one way that things steadily improve as you move down the page is in the amount of notational baggage. From each version to the next, we lose repetitive bits of symbol that one can just remember from context. Often, this is taken one step further! We now throw away the unknowns, the equal signs and some of the parentheses surrounding the matrices and vectors, and just write an *augmented* matrix.

$$\left(\begin{array}{ccc|c} 0 & 3 & 2 & 8 \\ 1 & -1 & 1 & -1 \\ 3 & 2 & 3 & 1 \end{array}\right)$$

This is the minimum fuss way to keep track of all the information you need to solve the original system.

Representing Elimination with Matrices

The process of elimination starts by performing operations on the system of equations. In the example above, one simplification we can make is to add -3 times equation (ii) to equation (iii).

Then the new system looks like this:

$$\left\{\begin{array}{rcrcrcr} & & 3y & + & 2z & = & 8 \\ x & - & y & + & z & = & -1 \\ & & 5y & & & = & 4 \end{array}\right.$$

Let's translate that into the linear combination format:

$$x \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + y \begin{pmatrix} 3 \\ -1 \\ 5 \end{pmatrix} + z \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 8 \\ -1 \\ 4 \end{pmatrix}.$$

What has happened to each of the vectors? Well, in each case, we have added -3 times the second component to the third component. We have seen that one way to change vectors into other vectors is by (left-)multiplying them by matrices. Could we be so lucky that the operation "add -3 times the second component to the third component" is representable by a matrix operation? YES. It is not hard to check that the matrix we need is

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix}.$$

In fact, let's check it now for each of the four vectors we have in our system:

$$E \begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

and

$$E \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ 5 \end{pmatrix}$$

and

$$E \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$$

and

$$E \begin{pmatrix} 8 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 8 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ -1 \\ 4 \end{pmatrix}.$$

Ha Ha! It all checks out. Those are the four vectors from our second system. This means that we can even use a simple subsitution to rewrite things. (It is not obvious at the moment why this is helpful. Hang on a bit.)

$$x * E\begin{pmatrix}0\\1\\3\end{pmatrix} + y * E\begin{pmatrix}3\\-1\\2\end{pmatrix} + z * E\begin{pmatrix}2\\1\\3\end{pmatrix} = E\begin{pmatrix}8\\-1\\1\end{pmatrix}$$

Boy, it sure would be nice if we had a compact way to write that down...

Matrix Multiplication

We make a compact way to write down that last equation by defining an operation of multiplying two matrices. If E and A are two matrices, we define their **matrix product** to be a new matrix as follows:

First, write A as a collection of columns v_i

$$A = \begin{pmatrix}v_1 & v_2 & v_3\end{pmatrix}$$

and then we declare that EA is the matrix made up of the columns Ev_i in the corresponding order.

$$EA = \begin{pmatrix}Ev_1 & Ev_2 & Ev_3\end{pmatrix}$$

By way of example, we have already considered the matrices

$$E = \begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & -3 & 1\end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix}0 & 3 & 2\\1 & -1 & 1\\3 & 2 & 3\end{pmatrix}.$$

You should check that their product is now

$$EA = \begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & -3 & 1\end{pmatrix}\begin{pmatrix}0 & 3 & 2\\1 & -1 & 1\\3 & 2 & 3\end{pmatrix} = \begin{pmatrix}0 & 3 & 2\\1 & -1 & 1\\0 & 5 & 0\end{pmatrix}$$

Finally, let's see how this influences our last two forms of the equations. The matrix form of our system was $Ax = v$ where A is as above, and the vectors are $v = \left(\begin{smallmatrix}8\\-1\\1\end{smallmatrix}\right)$ and $x = \left(\begin{smallmatrix}x\\y\\z\end{smallmatrix}\right)$. The neat part is that our new definition of matrix multiplication means that our elimination step transformed the equation

$$Ax = v \quad \text{or} \quad \begin{pmatrix}0 & 3 & 2\\1 & -1 & 1\\3 & 2 & 3\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}8\\-1\\1\end{pmatrix}$$

into the newer equation

$$(EA)x = Ev \quad \text{or} \quad \begin{pmatrix}0 & 3 & 2\\1 & -1 & 1\\0 & 5 & 0\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}8\\-1\\4\end{pmatrix}.$$

The same thing works for the augmented matrix. The augmented matrix form is really just writing down

$$\begin{pmatrix}A \mid v\end{pmatrix} = \left(\begin{array}{ccc|c}0 & 3 & 2 & 8\\1 & -1 & 1 & -1\\3 & 2 & 3 & 1\end{array}\right)$$

and the elimination step changes this into

$$\begin{pmatrix}EA \mid Ev\end{pmatrix} = \left(\begin{array}{ccc|c}0 & 3 & 2 & 8\\1 & -1 & 1 & -1\\0 & 5 & 0 & 4\end{array}\right).$$

Matrices as Transformations

Take a moment and reflect on the key transition in what happens above. The most important thing that made it all work was that a matrix (the elimination matrix E) was used to perform some sort of operation on vectors.

This is a key property of matrices. The matrix E defines a kind of function. For every vector w with three components, we can compute exactly one new vector Ew, still with three components. This means that E defines a function from the set of 3-vectors to the set of 3-vectors.

Sage and Matrix Multiplication

Sage has built-in matrix multiplication. You do the obvious thing and it works.

```
A = matrix(QQ, 2,2, [[2,1],[1,1]])
B = matrix(QQ, 2,2, [[0,3],[1,1]])
A, B
```

```
(
[2 1]  [0 3]
[1 1], [1 1]
)
```

```
A*B
```

You can check that it works with the way we defined matrix multiplication as a linear combination of vectors, too.

First, we define the column vectors by pulling out the entries from `B` and arranging them. To be sure, we ask Sage to display them as columns.

```
b1 = vector([B[0,0], B[1,0]])
b2 = vector([B[0,1], B[1,1]])
b1.column(), b2.column()
```

```
(
[0]  [3]
[1], [1]
)
```

```
A*b1.column(), A*b2.column()
```

```
(
[1]  [7]
[1], [4]
)
```

Now we can pile these rows into a matrix and then use the transpose to put them in columns.

```
C = matrix([A*b1, A*b2]).transpose()
C
```

```
[1 7]
[1 4]
```

And we can double check everything by asking Sage if these things are equal.

```
C == A*B
```

```
True
```

This kind of test can be useful for checking our work! The discussion above has this multiplication:

```
D = matrix(QQ,3,4, [0,3,2,8,1,-1,1,-1,3,2,3,1]); D
```

```
[ 0  3  2  8]
[ 1 -1  1 -1]
[ 3  2  3  1]
```

```
E = matrix.identity(3)
E[2,1] = -3
E
```

```
[ 1  0  0]
[ 0  1  0]
[ 0 -3  1]
```

```
E*D
```

```
[ 0  3  2  8]
[ 1 -1  1 -1]
[ 0  5  0  4]
```

Ta-Da!!!

Exercises

Task 2.17. In the main example above

$$\begin{cases} & & 3y & + & 2z & = & 8 \\ x & - & y & + & z & = & -1 \\ 3x & + & 2y & + & 3z & = & 1 \end{cases}$$

we would rather have our first pivot in the upper left corner (i.e. the first row should have a non-zero coefficient for x). This can be achieved by swapping the positions of rows (i) and (ii).

Find a matrix P_{12} so that multiplying by P_{12} on the left performs the corresponding row switching operation on the augmented matrix

$$\left(\begin{array}{ccc|c} 0 & 3 & 2 & 8 \\ 1 & -1 & 1 & -1 \\ 3 & 2 & 3 & 1 \end{array}\right)$$

Task 2.18. Consider the system

$$\begin{cases} 6x & - & y & = & 14 \\ 97x & - & 16y & = & 2/3 \end{cases}$$

Write this system in the other three forms: (1) an equation involving a linear combination of vectors; (2) an equation involving a 2×2 matrix; (3) an augmented matrix.

Task 2.19. Perform an elimination step on the system from the last exercise to put the sytem in triangular form. You should get two pivots.

Write the new system in each of our four forms.

Task 2.20. Still working with the same system of equations, use Sage to make two column picture plots:

- One showing the three relevant column vectors from the original system.

- One showing the three relevant column vectors from the system after the elimination step.

You may find it helpful to look through the Sage examples in previous sections of this workbook.

Task 2.21. One more time, stay with the same system of equations. Use Sage to make two row picture plots:

- One showing the two relevant lines in the original system.
- One showing the two relevant lines from the system after the elimination step.

You may find it helpful to look through the Sage examples in previous sections of this workbook.

Task 2.22. Consider this system of three equations in three unknowns:

$$\begin{cases} -x & + & \frac{2}{3}y & + & z & = & 1 \\ x & + & 6y & + & z & = & 1 \\ 3x & & & + & 3z & = & 1 \end{cases}$$

Perform the elimination steps to transform this system into a triangular one.

Write down the corresponding matrices you use to perform each of these steps on the augmented matrix version of this system.

Task 2.23. Still working with the system of equations from the last task, use Sage to make two column picture plots:

- One showing the four relevant column vectors from the original system.
- One showing the four relevant column vectors from the system after the elimination step.

You may find it helpful to look through the Sage examples in previous sections of this workbook.

Task 2.24. One more time, stay with the system of equations from previous two tasks. Use Sage to make two row picture plots:

- One showing the two relevant lines in the original system.
- One showing the two relevant lines from the system after the elimination step.

You may find it helpful to look through the Sage examples in previous sections of this workbook.

2.4 Going Further with Elimination

The Assignment

- Go back through the exercises in the first three sections of this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in Chapter Two: Linear Equations. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

2.5 Matrix Algebra

The Assignment

- Read section 2.4 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Add and subtract matrices of the same size.
- Multiply matrices of appropriate sizes by one method.
- Compute powers A^p of a given square matrix A.
- Use the distributive law for matrix multiplication and matrix addition correctly.

Sometime after our meeting, a student should be able to:

- Multiply block matrices.
- Multiply matrices by *three* methods.
- Give examples to show how matrix multiplication is not like ordinary multiplication of real numbers: including the trouble with commutativity, and the difficulty with inverses.

Discussion on Matrix Algebra

At the simplest level, this section is just about how to deal with the basic operations on matrices. We can add them and we can multiply them. We have already encountered matrix multiplication, and addition is even more natural.

But a subtle and important thing is happening here. Matrices are taking on a life of their own. They are becoming first class objects, whose properties are interesting and possibly useful.

This is an instance of the beginnings of *Modern Algebra*, which is the study of the algebraic structures of abstracted objects. In this case, we study whole collections of matrices of a common shape, and we try to treat them like generalized numbers. Then the natural questions are how much like "regular numbers" are these matrices?

Addition is about as well-behaved as you can expect, but multiplication is a bit trickier. Suddenly, two properties of multiplication for numbers don't quite work for matrices:

- multiplication does not necessarily commute: It need not be the case that AB is the same as BA.
- we may not always have inverses: just because there is a matrix A which is not the zero matrix, it may not be the case that we can make sense of A^{-1} and get $AA^{-1} = I$.

Sage and Matrix Algebra

Sage is aware of the basic matrix operations, and it won't let you get away with nonsense. Matrix multiplication and matrix addition are only defined if the dimensions of the matrices line up properly.

```
A = matrix(QQ, 2,3, [0,1,2,3,6,6])# A 2 by 3 matrix
B = matrix(QQ, 2,2, [4,2,3,1]) # 2 by 2 square matrix
C = matrix(QQ, 3,3, [2,1,2,1,2,1,2,1,2]) # a 3 by 3 square matrix
D = matrix(QQ, 2,3, [1,1,1,1,1,1]) # another 2 by 3 matrix
E = matrix(QQ, 3,2, [3,4,2,5,6,1]) # a 3 by 2 matrix
```

Let's see which of these Sage doesn't like. Can you predict, before evaluating the cells below, which of these will return an error?

```
A*B
```

```
B*A
```

```
[ 6 16 20]
[ 3  9 12]
```

```
A+B
```

```
A*C
```

```
[ 5  4  5]
[24 21 24]
```

```
C*A
```

```
A*D
```

```
A+D
```

Sage and Matrix Addition

Matrix addition works a lot like addition of integers, as long as you fix a size first.

- There is a zero element.
- There are additive inverses (i.e. *negatives*).
- The operation is commutative.

```
#This constructs the zero matrix
Z = zero_matrix(2,3); Z
```

```
[0 0 0]
[0 0 0]
```

Let us add `A` and `Z`:

```
A + Z
```

```
[0 1 2]
[3 6 6]
```

We can check that adding `Z` doesn't change anything.

```
A + Z == A
```

```
True
```

And we can do the natural thing to get an additive inverse.

```
L = -A; L
```

```
[ 0 -1 -2]
[-3 -6 -6]
```

Finally, this last thing should return zero.

```
A + L
```

```
[0 0 0]
[0 0 0]
```

Sage and Matrix Multiplication

Sage already has the structure of matrix multiplication built-in, and it can help with investigating the ways that matrix multiplicaiton is different from regular multiplication of numbers.

We have seen above that Sage will not let us multiply matrices whose sizes do not match correctly. Of course, one way around that trouble is to stick to square matrices. But even there we can have trouble with the fact that matrix multiplication might not commute. It is rarely the case that $XY = YX$.

For those of you who will eventually study Modern Algebra, the collection of all n-square matrices is an example of a non-commutative ring with unit.

```
A*E, E*A
```

```
(
         [12 27 30]
[14  7]  [15 32 34]
[57 48], [ 3 12 18]
)
```

Sage knows about the ring structure. We can check for an inverse.

```
B.is_invertible()
```

```
True
```

```
C.is_invertible()
```

```
False
```

And we can ask for the inverse in a couple of ways.

```
B.inverse()
```

```
[-1/2   1]
[ 3/2  -2]
```

```
B^(-1)
```

```
[-1/2   1]
[ 3/2  -2]
```

One can even construct the whole ring of all $n \times n$ matrices and play around inside it.

```
M = MatrixSpace(QQ, 2,2); M
```

```
Full MatrixSpace of 2 by 2 dense matrices over Rational Field
```

It is then not too hard to construct the identity element, which is the regular identity matrix of the correct size.

```
M(1)
```

```
[1 0]
[0 1]
```

Also, the zero matrix of the correct size is easy to make.

```
M(0)
```

```
[0 0]
[0 0]
```

In fact, this allows you to short cut the construction of any matrix in `M`. This can be really useful if you are going to work with a lot of matrices of the same shape.

```
H = M([2,1,1,1]); H
```

```
[2 1]
[1 1]
```

You can even use this to make complicated expressions out of matrix operations. As long as everything makes sense, Sage will do all the work.

```
H^2 + H
```

```
[7 4]
[4 3]
```

```
H * (H - 5*M(1) + 2*H^2)
```

```
[21 14]
[14  7]
```

```
# this one should turn out to be zero.
# I know this because of a theorem.
H^2 - 3*H + M(1)
```

```
[0 0]
[0 0]
```

Sage and Block Matrices

Sage has a built-in method for constructing matrices out of blocks, too. It is easiest to see an example.

To make this work, you have to tell Sage when you have a vector if you want it to be either a row or a column vector in the appropriate places.

```
X = matrix(QQ, 2,2, [0,-1,1,0])
v = vector([2,3])
w = matrix(QQ, 1,1, [1])
Blockey = block_matrix(QQ, 2,3,
                       [[X, X, v.column()],
                       [v.row(), v.row(), w]])
Blockey
```

```
[ 0 -1| 0 -1| 2]
[ 1  0| 1  0| 3]
[-----+-----+--]
[ 2  3| 2  3| 1]
```

Exercises

Task 2.25. Make an example of a 2×3 matrix and a 3×3 matrix, and use this to demonstrate the three different ways to multiply matrices.

Task 2.26. Give an example of a pair of 2×2 matrices A and B so that $AB = 0$ but $BA \neq 0$, or explain why this is impossible.

Task 2.27. Give an example of a 3×3 matrix A such that neither A nor A^2 is the zero matrix, but $A^3 = 0$.

Task 2.28. Find all examples of matrices A which commute with both $B = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ and $C = \left(\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$. That is, find all matrices A so that $AB = BA$ and $AC = CA$. How do you know you have all such matrices?

Task 2.29. Consider the matrix

$$A = \begin{pmatrix} 2 & 1 & 0 \\ -2 & 0 & 1 \\ 8 & 5 & 3 \end{pmatrix}.$$

Which elimination matrices E_{21} and E_{31} produce zeros in the $(2, 1)$ and $(3, 1)$ positions of $E_{21}A$ and $E_{31}A$?

Find a single matrix E which produces both zeros at once. Multiply EA to verify your result.

Task 2.30. Let's take a different view of the last computation. Block multiplication says that column 1 is eliminated by a step that looks like this one:

$$EA = \begin{pmatrix} 1 & 0 \\ -c/a & I \end{pmatrix} \begin{pmatrix} a & b \\ c & D \end{pmatrix} = \begin{pmatrix} a & b \\ 0 & D - cb/a \end{pmatrix}.$$

Here I is the 2×2 identity matrix, D is a 2×2 matrix, etc.

So, in the last exercise, what are a, b, c and D and what is $D - cb/a$? Be sure to describe what shape each matrix has: the number of rows and columns.

Task 2.31. Suppose that we have already solved the equation $Ax = b$ for the following three special choices of b:

$$Ax_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \; Ax_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \text{ and } Ax_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

If the three solutions are called x_1, x_2 and x_3 and then bundled together to make the columns of a matrix

$$X = \begin{pmatrix} | & | & | \\ x_1 & x_2 & x_3 \\ | & | & | \end{pmatrix},$$

what is the matrix AX? What does this mean about X?

2.6 Matrix Inverses

The Assignment

- Read section 2.5 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- State the definition of **invertible** matrix.
- Solve an equation $Ax = b$ using the inverse of A if it exists.
- State how inverses and multiplication interact.
- Use Gauss-Jordan elimination to compute the inverse of a matrix.
- State a test for invertibility of square matrices using pivots.

Some time after class, a student should be able to:

- Describe the connection between Gauss-Jordan elimination and solving n different systems of equations.
- Describe the connection between Gauss-Jordan elimination, computing matrix inverses, and the process of elimination by matrix multiplication.
- State the definition of the determinant of a square matrix.
- State the connection between the determinant of a square matrix and invertibility.
- State the distinction between a matrix being **invertible** and a matrix being **singular**.

Discussion

Matrix Inverses

The main point of this section is to start focusing on the first big problem in linear algebra. How can you tell, in advance, that a system of n equations in n unknowns will have a solution?

Of course, like all things we have been studying, this will have several different faces, all of which are equivalent. The one front and center right now is this: When does an $n \times n$ square matrix have an inverse?

Finding an Inverse: Gauss-Jordan Elimination

There is an effective method for finding the inverse, and it is Gauss-Jordan elimination. (This is sometimes just called *Gaussian elimination.*) Essentially, you wish to solve n different systems $Ax = b$ of size $n \times n$ all at the same time, with specially chosen right hand sides.

The process is an algorithm, so it is very specific. If you do this some other way, you aren't doing Gauss-Jordan Elimination. The name is applied to the process.

Gauss-Jordan Elimination

- *Augment:* Tack on a copy of the identity matrix of the same size to the right hand side of your matrix. It should now look like $(A \mid I)$.

- *Forward Pass:* This is a nested procedure:
  - *preparation:* If necessary, use a row swap to make a non-zero entry in the upper left entry.
  - *make zeros:* The upper left entry is our first pivot. Use the operation of adding a multiple of the first row to the other rows to kill the entries below this first pivot.
  - *step down:* Step down to the second row and repeat the above, but ignoring rows and columns above and to the left. Repeat as necessary till you run out of rows.

  If at any point in the process you get a row consisting of only zeros, perform a row switch to suffle it to the bottom. When the forward pass is complete, you should have an upper triangular matrix.
- *Backward Pass:* This is also nested, like the forward pass, except that instead of working down and to the right, you begin at the lower right with the last pivot and work up and to the left. When complete, the matrix should have at most one non-zero entry in each row. This entry will be a pivot.
- *Rescale:* rescale rows to make the pivots into 1's.

At the end of the whole process, you should have something that looks like this: $(I \mid B)$. The wonderful part: B is the inverse of A. Well, almost. The process can fail! If along the line you find that the left hand block of your big augmented matrix doesn't have n pivots in it, then your matrix was not invertible.

What you have computed in the left hand block with the Gauss-Jordan elimination is the *reduced row-echelon form* of your original matrix.

The Big Theorem: invertibility, singularity, and the determinant

What is the key?

Theorem 2.2. *An $n \times n$ matrix A is invertible exactly when it has n pivots. Equivalently, its reduced row-echelon form has n non-zero entries down the diagonal. The inverse will be computed by Gauss-Jordan elimination.*

This is huge. The algorithm is not difficult, and it answers an important question exactly.

Note that we said a square matrix was *singular* when it did not have enough pivots. So what the above says is that a matrix is invertible if and only if it is non-singular.

A simple test

We can use the above to make a simple numerical test of when a matrix is invertible. First do the forward pass of elimination to obtain an upper triangular matrix. Take the product of the diagonal entries. This will be zero if and only if one of the diagonal entries is zero, which will only happen if there are fewer than n pivots. This product is then helpful enough to test for invertibility, and so it deserves its own name: the **determinant**. We shall learn more about this quantity later.

Sage and Gauss-Jordan Elimination

We have already seen that Sage has commands for constructing matrices and performing row operations. Those are the operations used to perform Gauss-Jordan

Elimination. But there are several interesting and useful commands in this neighborhood we have not yet discussed.

Let us construct my favorite matrix so we have something to play with.

```
A = matrix(QQ, 2,2, [2,1,1,1])
```

We can use the `.is_invertible()` method to check that `A` is invertible. In general, this method returns `True` or `False`.

```
A.is_invertible()
```

```
True
```

And we can get Sage to just compute the inverse for us.

```
A.inverse()
```

```
[ 1 -1]
[-1  2]
```

Just so we can see what happens if the matrix is not invertible, we try another matrix.

```
B = matrix(QQ, 2,2, [0,1,0,0])
B.is_invertible()
```

```
False
```

```
B.inverse()
```

```
Error in lines 1-1
...
ZeroDivisionError: input matrix must be nonsingular
```

We can also ask Sage to compute determinants with the `.determinant()` method.

```
A.determinant(), B.determinant()
```

```
(1,0)
```

Sage is also capable of computing the reduced row echelon form (the "rref") of a matrix with the appropriately named `.rref()` method.

```
A.rref()
```

```
[1 0]
[0 1]
```

The method `.rref()` does not change the matrix `A`. There is another command which will work the same way for our purposes, `.echelon_form()`.

```
A.echelon_form()
```

```
[1 0]
[0 1]
```

There is a related command which *will find the rref and then update the matrix.* It is called `.echelonize()`. Because I don't really want to mess with `A`, we will make a copy first.

```
C = copy(A) # fancy Python trick! (not so fancy)
C.echelonize()
```

Now we ask sage to print those out for us.

```
print A
print '\n'
print C
```

```
[2 1]
[1 1]

[1 0]
[0 1]
```

Now, we can be just a bit more hands-on with Gauss-Jordan elimination if we do it this way. We will combine commands we have used before to do this.

```
M = MatrixSpace(QQ, 2,2)
M(1) # this is the 2x2 identity
```

```
[1 0]
[0 1]
```

Now we do the algorithm.

```
D = A.augment(M(1))
D.rref()
```

```
[ 1  0  1 -1]
[ 0  1 -1  2]
```

That was good. But we only need the right-hand submatrix. We can get Sage to report just that!

```
E = D.rref().matrix_from_columns([2,3]); E
```

```
[ 1 -1]
[-1  2]
```

It is often convenient to chain methods together like this. Then you can read what happens from left to right.

Exercises

Keep this in mind. The computations are simple, but tedious. Perhaps you want to use an appropriate tool.

Task 2.32. Use Gauss-Jordan elimination to find the inverse of the matrix A below.

$$A = \begin{pmatrix} 3 & 17 \\ 1 & 6 \end{pmatrix}$$

Be sure to clearly write down the operations you use and the matrices which perform the operations by left multiplication.

Task 2.33. Use Gauss-Jordan elimination to find the inverse of the matrix X below.

$$X = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Be sure to clearly write down the operations you use and the matrices which perform the operations by left multiplication.

Task 2.34. Use Gauss-Jordan elimination to find the inverse of the matrix B below.

$$B = \begin{pmatrix} 3 & 4 & -1 \\ 1 & 6 & 1 \\ 0 & 3 & -1 \end{pmatrix}$$

Be sure to clearly write down the operations you use and the matrices which perform the operations by left multiplication.

Task 2.35. Use Gauss-Jordan elimination to find the inverse of the matrix B below.

$$B = \begin{pmatrix} 0 & 3 & 4 & -1 \\ 0 & 1 & 6 & 1 \\ 2 & 0 & 3 & -1 \\ 5 & -1 & 1 & 3 \end{pmatrix}$$

Be sure to clearly write down the operations you use and the matrices which perform the operations by left multiplication.

Task 2.36. Use Gauss-Jordan elimination to find the inverse of the matrix D below.

$$D = \begin{pmatrix} 3 & 17 & -1 & 3 & 1 \\ 1 & 6 & -2 & 1 & 1 \\ 2 & 2 & 1 & -5 & 1 \\ 0 & 0 & 3 & 1 & -3 \\ -2 & 3 & 4 & 1 & 1 \end{pmatrix}$$

Task 2.37. Suppose that for the matrix D in the last exercise we imagine solving the matrix equation $Dx = b$ for some vector b of the appropriate size. What might one mean by the row picture in this case? What might the column picture mean?

Task 2.38. Design a 6×6 matrix which has the following properties:

- no entry equal to zero
- the reduced row echelon form should have exactly 5 pivots
- the 5 pivots should be different numbers
- no pair of rows should be scalar multiples of one another

Is your matrix invertible? How do you know? Does Sage say it is invertible?

2.7 The LU Decomposition

The Assignment

- Read section 2.6 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Use Gaussian Elimination to find the LU and LDU decompositions of a matrix.
- Describe when the process of Gaussian Elimination will fail to produce an LU decomposition.

Sometime after class, a student should be able to:

- Solve a system of equations by using the LU decomposition and two triangular systems.
- Explain the connection between matrix elimination and the LU or LDU factorization of a matrix.

Discussion: The LU Decomposition of a Matrix

We now look at the ideas behind elimination from a more advanced perspective. If we think about the matrix multiplication form of the forward pass, we can realize it a *matrix decomposition theorem*:

Theorem 2.3. *Any square matrix A can be written as a product $A = LU$ where L is a lower triangular matrix and U is an upper triangular matrix. Moreover, the matrix L will have 1's down its diagonal.*

There are three key observations that make this work:

- Each of the matrices E_{ij} that affects a row operation of the form *add a multiple of row i to row j* is an invertible matrix, with an easy to find inverse.
- If we make a sequence of row operations in the forward pass using matrices E_k, then we are essentially computing a big product

  $$E_k \dots E_1 A = U$$

  where each of the E_i's is a lower triangular matrix and the matrix U is upper triangular. This can the be rewritten as

  $$A = \left(E_1^{-1} \dots E_k^{-1}\right) U.$$

  Note that the inverses have to be done *in reverse order* for things to cancel out properly.
- Finally, the product $L = E_1^{-1} \dots E_k^{-1}$ is really easy to compute, because its entries are simply the negatives of the multipliers we used to do the operations in the forward pass.

A Nice Computational Result

One important output of this comes into play when we want to compute solutions to equations like $Ax = b$. Since we can write $A = LU$, then our equation can be split into two (big) steps:

1. First find the solution to the equation $Ly = b$.

2. Then find the solution to the equation $Ux = y$.

First, note that this is a good thing because both of the systems $Ly = b$ and $Ux = y$ are triangular. They can be solved by back substitution. In $Ly = b$ you work from the top down, and in $Ux = y$ you work from the bottom up.

Second, this works because following this process gives us a vector x which will satisfy this:

$$Ax = (LU)x = L(Ux) = Ly = b.$$

Third, this doesn't really save time when you only want to solve one equation $Ax = b$. But if you have lots of different values of b_i, and you want to solve all of the equations $Ax = b_i$, it becomes a lot faster to factor the matrix $A = LU$ once and do two back substitutions for each value of b_i.

Sage and The LU Decomposition

A neat feature of linear algebra is that simple facts about solving equations have several different incarnations. This section contains the first big example: Gaussian Elimination leads to a multiplicative decomposition (a factorization) for matrices.

Each step of Gaussian elimination is a simple row operation, and if we do the process in the standard order, then the LU decomposition can be read out directly, without any extra computation.

First, let us recall how Sage can help us check bits of the three key observations above.

```
M = MatrixSpace(QQ,3,3)
One = M(1); One
```

```
[1 0 0]
[0 1 0]
[0 0 1]
```

Consider a matrix which performs an elementary row operation of the form “add a multiple of one row to another”. The matrix E below performs the operations *add* -4 *times row 2 to row 3.*

```
E = One.with_added_multiple_of_row(2,1,-4); E
```

```
[1  0  0]
[0  1  0]
[0 -4  1]
```

```
E.is_invertible()
```

```
True
```

```
E.inverse()
```

```
[1 0 0]
[0 1 0]
[0 4 1]
```

Note that the inverse just came from changing the sign of that one entry. This makes sense for the following reason: the opposite operation to "add -4 times row 2 to row 3" should be "Add 4 times row 2 to row 3". That is the way you undo the operation!

Study Break: Try it yourself

Make your own 3×3 matrix and check the whole procedure.

Sage Commands to short-cut the process

Here is the basic command for getting Sage to compute the LU decomposition directly.

```
A = M([2,3,1,-1,3,5,6,5,4]); A
```

```
[ 2  3  1]
[-1  3  5]
[ 6  5  4]
```

```
A.LU()
```

```
(
[0 0 1]  [   1    0    0]  [     6      5      4]
[0 1 0]  [-1/6    1    0]  [     0   23/6   17/3]
[1 0 0], [ 1/3 8/23    1], [     0      0 -53/23]
)
```

Hold on, the output is three matrices. Not two, but three. One is upper triangular, one is lower triangular, but the first one is a **permutation matrix**. (It switches rows 1 and 3.) What is going on? If you perform a search in the Sage documentation, you find this page. There is a description of the command, and the first bit is something about a "pivoting strategy" and row swaps. But we don't want row swaps.

By reading carefully, we can see what the way through is, too. We can specify our pivoting strategy by adding the keyword argument `pivot="nonzero"` inside the parentheses. Then the algorithm used will match the one Strang describes.

(If you are using SMC, you can access the help using many other ways. But a Google search for `Sage math "topic"` will hit the documentation pretty reliably.)

```
A.LU(pivot='nonzero')
```

```
(
[1 0 0]  [   1    0    0]  [   2    3    1]
[0 1 0]  [-1/2    1    0]  [   0  9/2 11/2]
[0 0 1], [   3 -8/9    1], [   0    0 53/9]
)
```

Aaah! There we go, now the permutation part is the identity. Note that the command returns a "tuple". This is a collection of things, kind of like a list. (Technical Python details omitted here.) To grab the information out, we assign the parts of that output to different names so we can use them.

```
P, L, U = A.LU(pivot='nonzero')
```

```
L # this is the lower triangular part
```

```
[   1    0    0]
[-1/2    1    0]
[   3 -8/9    1]
```

```
U # this is the upper triangular part
```

```
[   2    3    1]
[   0  9/2 11/2]
[   0    0 53/9]
```

Those parts should be factors of A. We can check:

```
L*U # this should multiply to A
```

```
[ 2  3  1]
[-1  3  5]
[ 6  5  4]
```

And we can have Sage check if they are really equal.

```
L*U == A
```

```
True
```

What about the LDU decomposition?

For now, Sage has no built-in LDU decomposition.

An insurmountable obstacle

Some matrices *require* permutations of rows. In these cases, we have to have some pivoting strategy *must* be employed. Consider this example.

```
B = M([0,2,2,1,3,1,1,1,1]); B
```

```
[ 0  2  2]
[ 1  3 -1]
[ 1  1  1]
```

```
B.LU(pivot='nonzero')
```

```
(
[0 1 0]  [ 1  0  0]  [ 1  3 -1]
[1 0 0]  [ 0  1  0]  [ 0  2  2]
[0 0 1], [ 1 -1  1], [ 0  0  4]
)
```

This has a row-swap permutation matrix, and *it must*. Since the (1,1) entry of `B` is zero, but numbers below that are not zero, we cannot use zero as a pivot. We'll sort out how to handle this in the next section.

Exercises

Keep this in mind. The computations are simple, but tedious. Perhaps you want to use an appropriate tool.

Task 2.39. Consider the following system of 3 linear equations in 3 unknowns.

$$\begin{cases} x + y + z = 5 \\ x + 2y + 3z = 7 \\ x + 3y + 6z = 11 \end{cases}$$

Perform the forward pass of elimination to find an equivalent upper triangular system. Write down this upper triangular system. What three row operations do you need to perform to make this work?

Use the information you just found to write a matrix decomposition $A = LU$ for the coefficient matrix A for this system of equations. (Be sure to multiply the matrices L and U to check your work.)

Task 2.40. Solve the two systems $Ly = b$ and $Ux = y$ obtained in the last exercise.

Solve the system $Ax = b$ directly using Gauss-Jordan elimination (hint: use Sage) and make sure that the results are the same.

Task 2.41. Consider the matrix A below. Find the matrix E which transforms A into an upper triangular matrix $EA = U$. Find $L = E^{-1}$. Use this to write down the LU decomposition $A = LU$ of A.

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 4 & 2 \\ 6 & 3 & 5 \end{pmatrix}$$

Task 2.42. The matrix below is **symmetric**, because if you flip it across its main diagonal you get the same thing. Find the LDU triple decomposition of this symmetric matrix.

$$B = \begin{pmatrix} 2 & 4 \\ 4 & 11 \end{pmatrix}$$

Task 2.43. The matrix below is **symmetric**, because if you flip it across its main diagonal you get the same thing. Find the LDU triple decomposition of this symmetric matrix.

$$C = \begin{pmatrix} 1 & 4 & 0 \\ 4 & 12 & 4 \\ 0 & 4 & 0 \end{pmatrix}$$

Task 2.44. The matrix below is **symmetric**, because if you flip it across its main diagonal you get the same thing. Find the LU decomposition of this symmetric matrix.

$$D = \begin{pmatrix} a & a & a & a \\ a & b & b & b \\ a & b & c & c \\ a & b & c & d \end{pmatrix}$$

What conditions on the variables a, b, c, and d will guarantee that this matrix has four pivots?

Task 2.45. Find an example of a 3×3 matrix A which has all of its entries non-zero, so that the LU decomposition has $U = I$, where I is the identity matrix, or explain why no such example exists.

2.8 Permutation Matrices

The Assignment

- Read section 2.7 of *Strang*
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Compute the transpose of a matrix.
- Correctly perform calculations where the transpose interacts with the operations of matrix sum, matrix product, and matrix inverse.
- Compute inner and outer products using the transpose.
- Decide if a matrix is symmetric or not.
- Recognize permutation matrices, and design permutation matrices which correspond to given row swaps.

Some time after class, a student should be able to:

- Find the LDL^T decomposition for symmetric matrices.
- Explain how the necessity of permuting rows during Gaussian elimination leads to the decomposition $PA = LU$.
- Explain why $P^T = P^{-1}$ for permutation matrices.

Discussion: Transposes, Symmetric Matrices, and Permutations

An important operation on matrices we have yet to encounter is called the **transpose**. If A is an $m \times n$ matrix, the transpose A^T of A is made by changing the roles of the rows and the columns. The result A^T will be an $n \times m$ matrix, because of this switch.

For now, the transpose will feel like some random thing, but its primary importance comes from its connection with the dot product. If we think of column vectors u and v of size n as if they are $n \times 1$ matrices, then the dot product $u \cdot v$ can be computed with a nice combination of matrix multiplication and the transpose:

$$u \cdot v = u^T v.$$

On the right, this is matrix multiplication! That makes sense because u^T is $1 \times n$ and v is $n \times 1$. This means that the result is a 1×1 matrix, i.e. a number.

Since the dot product contains all of the geometry of Euclidean space in it, the transpose becomes an important operation. I know that sounds weird, but the dot product contains all of the information we need to measure lengths and angles, so basically all of the *metric* information in Euclidean geometry is there.

Algebraic results about the transpose

There are some key results about the way the transpose interacts with other matrix operations, each of these can be checked with some tedious computation:

- If A and B are matrices of the same shape, then $(A+B)^T = A^T + B^T$.
- If A and B are of sizes so that AB is defined, then $(AB)^T = B^T A^T$.
- If A is an invertible matrix, with inverse A^{-1}, then A^T is also invertible and it has inverse $\left(A^T\right)^{-1} = \left(A^{-1}\right)^T$.

Symmetric Matrices

A matrix A is called *symmetric* when $A^T = A$. These pop up in lots of interesting places in linear algebra. A neat result is that a symmetric matrix has a symmetric looking LDU decomposition:

$$\text{if } A^T = A, \text{ then } A = LDL^T.$$

That is, in the LDU decomposition, $U = L^T$.

There are several ways to get symmetric matrices. For example, if A is any matrix, the new matrix $B = A^T A$ will be symmetric. (Check this.) Also, the matrix $S = A^T + A$ will be symmetric.

Permutation Matrices and Pivoting strategies in Gauss-Jordan Elimination

It is sometimes the case that Gauss-Jordan elimination requires a row swap. As we have seen, the operation of swapping a row can be achieved by left multiplying by a matrix of a special type. If we take a bunch of those and multiply them together, we still get a matrix which is in a special class: *the permutation matrices*.

A permutation matrix is square matrix having a single 1 in each column and in each row. A helpful property of permutation matrices is that they are invertible, and their inverses are the same as their transposes:

$$P^{-1} = P^T.$$

Gauss-Jordan elimination is easy enough to understand, now. It is time to let go of performing all those arithmetic operations by hand. So, permutation matrices become important for a different reason! Even if Gauss-Jordan elimination can be done without a row swap, it may be numerically better for a computer to swap out for a larger number as a pivot, so a row swap is used anyway. This partial pivoting strategy is encapsulated in most computer algebra algorithms in some way, and is part of the computation involved in computing a PLU decomposition. Strang has a decent discussion of the choices, below we will discuss how Sage handles this.

Sage and Transposes, Symmetry, Permutations, and Pivots

There is a lot going on in this little section. At first glance, it is a bit intimidating. But we have seen most of the ideas before.

The Transpose

The transpose of a matrix is what you get by switching the roles of rows and columns. Sage has a simple method for this.

```
M = MatrixSpace(QQ, 3,3)
A = M([1,2,3,4,5,6,7,8,9]); A
```

```
[1 2 3]
[4 5 6]
[7 8 9]
```

```
A.transpose()
```

```
[1 4 7]
[2 5 8]
[3 6 9]
```

One place that the transpose is useful is in describing the dot product. Check this out.

```
u = vector([1,2,3])
v = vector([4,5,6])
u.dot_product(v)
```

```
32
```

```
U = u.column(); U # this puts u into a column matrix
```

```
[1]
[2]
[3]
```

To be sure, we check what the "parent" of `U` is.

```
U.parent()
```

```
Full MatrixSpace of 3 by 1 dense matrices over Integer Ring
```

See! Sage thinks of `U` as a matrix with 3 rows and 1 column.
Now we do the same with `v`

```
V = v.column()
V
```

```
[4]
[5]
[6]
```

Now the magic.

```
U.transpose()*V
```

```
32
```

```
V.transpose()*U
```

```
32
```

That is the dot product, but stuffed into a 1×1 matrix!

Other Properties

The transpose has other useful properties. Strang lists the big ones, including how the transpose interacts with matrix multiplication and matrix inverses.

Symmetry

A matrix is called **symmetric** when it is equal to its transpose. Sage has some built-in commands for this.

```
B = M([2,1,0,1,1,0,0,0,1])
B
```

```
[2 1 0]
[1 1 0]
[0 0 1]
```

```
B.transpose()
```

```
[2 1 0]
[1 1 0]
[0 0 1]
```

```
B.is_symmetric()
```

```
True
```

```
C = M([1,0,1,1,1,1,0,0,0]); C
```

```
[1 0 1]
[1 1 1]
[0 0 0]
```

```
C.is_symmetric()
```

```
False
```

Strang notes a really neat property of symmetric matrices. Their LDU decompostions are nicer than average.

```
P, L, U = B.LU(pivot='nonzero')
```

```
P # here, things are good and no row swaps are needed
```

```
[1 0 0]
[0 1 0]
[0 0 1]
```

```
L
```

```
[  1   0   0]
[1/2   1   0]
[  0   0   1]
```

```
U
```

```
[  2   1   0]
[  0 1/2   0]
[  0   0   1]
```

```
D = M([2,0,0,0,1/2,0,0,0,1])
Uprime = D.inverse()*U
Uprime
```

```
[  1 1/2   0]
[  0   1   0]
[  0   0   1]
```

```
B == L*D*Uprime
```

```
True
```

```
L.transpose() # this is the neat part
```

```
[  1 1/2   0]
[  0   1   0]
[  0   0   1]
```

Permutations and Pivots

We have seen that elimination sometimes requires us to perform a row operation of swapping the position of two rows to put a pivot in a good place. At first, we want to do this to avoid a zero. But for computational reasons, a machine really likes to have a *big* number as a pivot. So software often uses rows swaps even when not strictly needed.

If all we care about is finding the reduced row echelon form (rref), then this won't worry us. You do whatever operations you want, and the rref is always the same thing. But if we want to keep track with matrices, things get a little complicated.

Here is the important stuff to remember:

1. A row swap is performed by a permutation matrix. A permutation matrix is a matrix with exactly one 1 in each column and in each row. These matrices have the important property that their transposes and their inverses are equal. That is, if P is a permutation matrix, then P^T is equal to P^{-1}. (Not every matrix with this extra property is a permutation matrix. Be careful.)

2. It is possible to figure out what all of the row swaps should be, and then rearrange all of the amtrices in an LU decomposition routine. If you do it correctly, you get:
$$P'A = LU$$
or
$$A = PLU$$
where P' and P are permutation matrices.

Note: Strang prefers to write things as $P'A = LU$, but Sage writes $A = PLU$. Fortunately, there is a simple relationship here. Strang's P' is the transpose (and hence the inverse!) of Sage's P.

If you haven't figured it out by now, I think that row reduction by hand is really for chumps. Sage (or whatever computational tool you use) makes it waaaaaaaaay easier.

```
# using 'partial pivoting' where we get "big pivots"
P, L, U = A.LU()
x = '{0!r}\n\n{1!r}\n\n{2!r}'.format(P,L,U)
print x # fancy python tricks for readable display
```

```
[0 1 0]
[0 0 1]
[1 0 0]

[  1   0   0]
```

```
[1/7   1   0]
[4/7 1/2   1]

[   7    8    9]
[   0  6/7 12/7]
[   0    0    0]
```

```
P*L*U
```

```
[1 2 3]
[4 5 6]
[7 8 9]
```

```
A == P*L*U
```

```
True
```

```
P.transpose()*A == L*U
```

```
True
```

```
P.transpose()*A
```

```
[7 8 9]
[1 2 3]
[4 5 6]
```

Exercises

Keep this in mind. The computations are simple, but tedious. Perhaps you want to use an appropriate tool.

Task 2.46. Find an example of a matrix A such that $A^T A = 0$, but $A \neq 0$.

Task 2.47. These are true or false questions. If the statement is true, explain why you know it is true. If the statement is false, give an example that shows it is false.

1. The block matrix $\left(\begin{smallmatrix} A & 0 \\ 0 & A \end{smallmatrix}\right)$ is automatically symmetric.
2. If A and B are symmetric, then their product AB is symmetric.
3. If A is not symmetric, then A^{-1} is not symmetric.

Task 2.48. If P_1 and P_2 are permuation matrices, then so is $P_1 P_2$. Give examples with $P_1 P_2 \neq P_2 P_1$ and $P_3 P_4 = P_4 P_3$.

Task 2.49. Explain the following phenomena in terms of row operations.

1. For any permutation matrix P, it is the case that $P^T P = I$.
2. All row exchange matrices are symmetric: $P^T = P$. (other permutation matrices may or may not be symmetric.)
3. If P is a row exchange matrix, then $P^2 = I$.

Task 2.50. For each of the following, find an example of a 2×2 symmetric matrix with the given property:

1. A is not invertible.

2. A is invertible but cannot be factored into LU.
3. A can be factored into LDL^T, but not into LL^T because D has negative entries.

Task 2.51. This is a new factorization of A into *triangular times symmetric*:

Start with $A = LDU$. Then $A = BS$, where $B = L\left(U^T\right)^{-1}$ and $S = U^T DU$.

Explain why this choice of B is lower triangular with 1's on the diagonal. Expain why S is symmetric.

2.9 Matrix Algebra: Going Further

The Assignment

- Go back through the exercises in the last four sections of this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in Chapter Two: Linear Equations. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

Chapter 3

Vector Spaces and Subspaces

In the last chapter we took on the central problem of linear algebra: solving systems of linear equations. We built a lot of useful tools, but we only answered the questions in the case of *square systems* where the number of variables is equal to the number of equations.

Now it is time to solve the problem in general. As a reminder, the set of questions is this:

- What is the set of solutions to a given system of linear equations?
- When does a given system have solution, and when does it not?
- If there is a solution, how many solutions are there?
- What ways do we have of describing the collection of solutions?
- Is there a computationally effective way to find those solutions?

We will run into the need for a few new tools, each of which has been hiding in the background in our work so far. For square systems, we could get away without this level of detail, but now we will need these concepts to clarify our work: **subspaces**, **rank** and **reduced row echelon form**, and **bases**.

This chapter concludes with a description of the *four fundamental subspaces* associated to a matrix. These will help us put together an understanding of the transformational model of an $m \times n$ matrix as a function with domain $\mathbb{R}^n$ and target $\mathbb{R}^m$.

3.1 Subspaces of $\mathbb{R}^n$

The Assignment

- Read Chapter 3 section 1 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Identify the column space of a matrix.
- Decide if a set in $\mathbb{R}^n$ is a subspace or not.

Some time after class, a student should be able to:

- Use the notion of column space to decide if a given linear system has a solution or not.

Discussion: Vector Spaces and Subspaces

The key concepts in this section are those of a **vector space** and of a **subspace**. The basic idea is that a vector space is a kind of place where the basic operations involved in a linear combination make sense. There is a set of rules for being a vector space, but they are all aimed at the fact that there are two operations (addition and scalar multiplication) and we can form linear combinations with them that make sense.

The biggest thing is that we possibly enlarge the kinds of things we call "vectors." To a professional mathematician, a vector is anything that is an element of some vector space under consideration. My favorites are things like this:

- The set $M_{2,2}$ of 2×2 matrices is a vector space. But now the things we call vectors are actually matrices.
- The set $\mathcal{C}(\mathbb{R})$ of continuous functions with domain and range both equal to the set of real numbers is a vector space. But now the things we call vectors are actually functions.
- The set $\ell(\mathbb{R})$ of sequences $(x_1, x_2, x_3, x_4, \ldots)$ of real numbers is a vector space. But now the things we call vectors are actually whole infinite sequences.

In more advanced mathematics, vector spaces are extremely important. They show up everywhere. But for us, we will mostly stick to the family of vector spaces $\mathbb{R}^n$. It will be much more important for us to understand subspaces.

The idea of a subspace is some subset, some part, of a vector space which is a vector space in its own right. The prototype is the xy-plane inside of $\mathbb{R}^3$.

For now, the most important subspaces we see will be derived from individual matrices. Our first example is the column space of a matrix A. If A is an $m \times n$ matrix, then the column space $\mathrm{col}(A)$ of A is the collection of n-vectors which can be expressed as linear combinations of the columns of A. This is our first exposure to the idea of a **span**. The column space of A is the subspace of $\mathbb{R}$ spanned by the columns of A.

Sage and subspaces

Sage allows you to construct different kinds of vector spaces. The most important is the standard vector space $\mathbb{R}^n$.

```
V = VectorSpace(QQ, 3)
V
```

```
Vector space of dimension 3 over Rational Field
```

This is like the `MatrixSpace` construction we have seen before. `V` is now the collection of all vectors of size 3 with rational numbers as entries.

More important for us is the fact that Sage knows how to find the subspaces associated to matrices.

```
A = matrix(QQ, 2,3, [4,5,3,0,1,1])
A
```

```
[4 5 3]
[0 1 1]
```

```
A.column_space()
```

```
Vector space of degree 2 and dimension 2 over Rational Field
Basis matrix:
[1 0]
[0 1]
```

Note that Sage gives information in terms of a basis. We will talk about this concept soon.

```
B = matrix(QQ, 3,3 , [1,2,3,4,5,6,7,8,9])
B
```

```
[1 2 3]
[4 5 6]
[7 8 9]
```

```
B.column_space()
```

```
Vector space of degree 3 and dimension 2 over Rational Field
Basis matrix:
[ 1  0 -1]
[ 0  1  2]
```

This last bit here is interesting. Our original matrix has size 2×3, so the columns are 3-vectors. But this basis is displayed as a matrix where the columns are size 2. What is going on?

Sage prefers to display vectors as rows! This is a big interal preference. We just have to get over it. Well, we have to *remember it* and get over it. To access the basis vectors and display them as columns, we will use the following:

```
B.column_space().basis()[0].column() # first basis vector
```

```
[ 1]
[ 0]
[-1]
```

```
B.column_space().basis()[1].column() # second basis vector
```

```
[0]
[1]
[2]
```

Exercises

Task 3.1. Find an example of a vector which is not in the column space of the matrix

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix},$$

or explain why that is not possible.

Task 3.2. Find an example of a vector which is not in the column space of the matrix

$$B = \begin{pmatrix} 3 & 2 \\ 6 & 4 \end{pmatrix},$$

or explain why it is not possible.

Task 3.3. Let $\mathcal{P}$ be the set of polynomials of degree 3 or less. Explain why $\mathcal{P}$ is a vector space, or explain why it is not.

Task 3.4. Consider the vector space $\mathbb{R}^2$. Explain why the following are not subspaces:

- The unit circle.
- The line $x + y = 4$.
- The union of lines $2x + 3y = 0$ and $x - y = 0$.
- The first quadrant where $x \geq 0$ and $y \geq 0$.

Task 3.5. Let W be the set of functions f in $\mathcal{C}(\mathbb{R})$ which satisfy the differential equation $f''(x) = 0$. Decide if W is a subspace of $\mathcal{C}(\mathbb{R})$, and explain your thinking.

Task 3.6. Design a 2×3 matrix whose column space does not contain the vector $\begin{pmatrix} 4 \\ 4 \end{pmatrix}$, or explain why this is not possible.

3.2 The Nullspace

The Assignment

- Read chapter 3 section 2 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before coming to class, a student should be able to:

- Find the special solutions associated to a matrix.
- Compute the **reduced row echelon form** of a matrix.
- Identify the free variables and the pivot variables associated to a matrix.
- Describe the nullspace of a matrix with the "simplest" set of equations possible.

Some time after class, a student should be able to:

- Describe the complete solution to a homogeneous system of equations $Ax = 0$.
- Give an argument for why the nullspace is a vector subspace.

Discussion: The Nullspace, RREF, and Special Solutions

Strang aims right at the heart of things in this section, and does not waste any space. Let me highlight a few things:

Let A be an $m \times n$ matrix. We do not require that A is a square matrix, that is, it may not be the case that $m = n$. What we are most interested in is solving the equation $Ax = 0$. Note that if A is not square, then the vectors x and 0 have different sizes.

By the way, an equation like $Ax = 0$ where the right hand side is zero is called a **homogeneous equation**.

- The nullspace of A is the set of vectors x such that $Ax = 0$. It is a theorem that this is a vector subspace of $\mathbb{R}^n$. It is a common to use the synonym **the kernel of** A in place of the terminology **the nullspace of** A**.**
- The key to everything is to not mind that your matrix isn't square. Just do Gauss-Jordan elimination anyway. The end result is called the **reduced row echelon form** of the matrix, or RREF for short.
- Note that there is no discussion of using an augmented matrix in this section, even though we are solving systems of equations $Ax = 0$. This is because the vector on the right hand side is zero and will stay zero. There is nothing interesting to track!
- The RREF is good for several things. One that is often overlooked is that one can use it to rewrite the system of equations in a "simplest form." I mean that the equations left over in the RREF are somehow the easiest equations to use to cut out the solution as an intersection of hyperplanes.
- In the RREF, all of the structure can be inferred from the splitting of the columns into two types: the pivot columns and the free columns. Since each column is associated to a variable in our system of linear equations (the columns hold coefficients!), it is also common to refer to pivot variables and free variables.

- The number of free columns basically determines the “size” of the nullspace. This is an entry point to the concept of the **dimension** of a vector space. We shall see this in more detail later.
- Strang points out an easy way to find some individual vectors in the null space: he calls these the **special solutions**. This is because they are solutions to the equation $Ax = 0$.

Sage and the Nullspace

The main idea in this section is to get solutions to homogeneous systems $Ax = 0$ by using the reduced row echelon form. This will compute the nullspace of the matrix A. Sage has some useful built in commands for this. Let us explore an example.

```
entries = [1,2,3,4,5,6,7,8,9,10,11,12]
A = matrix(QQ, 3,4, entries); A
```

```
[ 1  2  3  4]
[ 5  6  7  8]
[ 9 10 11 12]
```

The most direct way is to ask Sage for the nullspace. But Sage doesn’t call it that. It uses the synonym **kernel**. Also, because we do matrix vector multiplication with vectors on the right, we ahve to tell Sage to do it that way.

```
A.right_kernel()
```

```
Vector space of degree 4 and dimension 2 over Rational Field
Basis matrix:
[ 1  0 -3  2]
[ 0  1 -2  1]
```

Notice that Sage returns the nullspace by specifying a basis. This is a complete set of special solutions! How can we be sure?

Let’s check that those vectors are actually in the nullspace. This command should return two zero vectors in $\mathbb{R}^3$.

```
sage1 = vector([1,0,-3,2])
sage2 = vector([0,1,-2,1])
A*sage1, A*sage2
```

```
((0, 0, 0), (0, 0, 0))
```

Remember that the method to find the reduced row echelon form is .rref().

```
A.rref()
```

```
[ 1  0 -1 -2]
[ 0  1  2  3]
[ 0  0  0  0]
```

What this means is that the null space is described by the two equations

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

That is, the nullspace is the intersection of these two hyperplanes in $\mathbb{R}^4$.

```
# Let’s see what Sage’s solve command tells us:
v,x,y,z = var('v x y z')
expr1 = v - y - 2*z == 0
expr2 = x + 2*y + 3*z == 0
solve([expr1, expr2], [v,x,y,z])
```

```
[[v == 2*r5 + r6, x == -3*r5 - 2*r6, y == r6, z == r5]]
```

Aha! Sage is trying to tell us that the pivot variables v and x should be written in terms of the free variables y and z. Strang's "easy way out" is to choose the free variables to be 0's and 1's in combination. Here we have two of them, so we will alternate. One special solution will be to choose $y = 1$ and $z = 0$. Then we solve to get $v = 1$ and $x = -2$.

```
special1 = vector([1,-2,1,0])

# let's check
A*special1
```

```
(0, 0, 0)
(0, 0, 0)
```

Similarly, we can choose $y = 0$ and $z = 1$ to get a second special solution. This leads to $v = 2$ and $x = -3$.

```
special2 = vector([2,-3,0,1])

#let's check
A*special2
```

```
(0, 0, 0)
```

These two vectors form a **basis** for the nullspace. Every vector in the nullspace can be written as a linear combination of these two vectors.

What is weird is that Sage always wants its basis to look like 1's and 0's at the beginning, and our process makes them look like that at the end!

How can we be sure everything lines up? Well, it is possible to express the two vectors that Sage gives us as a linear combination of the ones we just found, and vice versa. So we are getting *two* descriptions of the same space.

```
sage1 == -3* special1 + 2* special2
```

```
True
```

```
sage2 == -2* special1 + special2
```

```
True
```

```
special1 ==  sage1 - 2* sage2
```

```
True
```

```
special2 == 2* sage1 - 3* sage2
```

```
True
```

Can you see the matrix and its inverse hiding behind that? I found those relationships using row operations and a matrix inverse.

Anyway, what has happened is that Sage has performed our calculations above, and then taken the extra steps of putting the matrix of basis vectors into reduced row echelon form, too.

Exercises

For the first three exercises, your job is to find both

- The minimal set of equations which cuts out the nullspace of the given matrix as an intersection of hyperplanes; and
- the size of the nullspace, expressed through the number "independent directions" it contains.

As always, it will be crucial to explain how you know you are correct.

Task 3.7. Consider the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 8 & 5 \\ -1 & -2 & 0 \end{pmatrix}$$

Task 3.8. Consider the matrix

$$A = \begin{pmatrix} 2 & 1 & 3 & 7 \\ 1 & 1 & 1 & -3 \end{pmatrix}$$

Task 3.9. Consider the matrix

$$A = \begin{pmatrix} 2 & 3 & 5 & 6 \\ 1 & 1 & 2 & 3 \\ 0 & 1 & 1 & 0 \\ -1 & -1 & -2 & -3 \\ 1 & 1 & 2 & 3 \end{pmatrix}$$

For the next three exercises, your job is to find a complete set of special solutions to the homogeneous equation $Ax = 0$. By a "complete set," we mean "enough special solutions so that any vector in the nullspace of A can be expressed as a linear combination of your vectors."

As always, it will be crucial to explain how you know you are correct.

Task 3.10. Consider the matrix

$$A = \begin{pmatrix} 6 & 7 \\ 7 & 8 \\ 1 & 0 \\ 4 & 5 \end{pmatrix}$$

Task 3.11. Consider the matrix

$$A = \begin{pmatrix} 23 & 17 & 9 & 2 \\ 1 & -2 & 4 & 1 \\ 22 & 19 & 5 & 1 \end{pmatrix}$$

Task 3.12. Consider the matrix

$$A = \begin{pmatrix} -3 & -6 & -9 & -12 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ -1 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

3.3 Rank and the RREF

The Assignment

- Read chapter 3 section 3 of *Strang*.
- Read the following and do the exercises below.

Learning Goals

Before class, a student should be able to:

- State a definition of the rank of a matrix.
- Compute the rank of a matrix.
- Describe the relation between the rank and the size of the column space.
- Construct the nullspace matrix associated to a matrix.

Some time after class, a student should be able to:

- Use the RREF and special solutions associated to a matrix to describe how free columns are linear combinations of earlier pivot columns.

Discussion: The Rank of a Matrix

The new concept for this section is the **rank** of a matrix. There are *three* definitions of rank given in the first two pages of this section. It is important to know them and see that they are equivalent.

Another important part of what happens here is the realization that putting the matrix into reduced row echelon form, which uses *row* operations, actually tells us something about the *columns* of the matrix. This is kind of amazing.

Take a matrix A and put it into reduced row echelon form $R = \mathrm{rref}(A)$. Then A and R have the same shape. In particular, they have the same number of columns. But in R it is easy to see the location of the pivots. This divides up the columns of R into those columns having a pivot, the **pivot columns**, and those which do not, the **free columns**. We will label a column of A as a pivot column or a free column in a way that corresponds. Here is the startling fact:

Theorem 3.1. *Let A be a matrix. The pivot columns of A are those which cannot be expressed as linear combinations of the columns to their left. The free columns of A are those which can be expressed as linear combinations of the columns to their left.*

How exactly can we see this? Strang uses the construction of a **nullspace matrix**. The special solutions we learned about in the last section can be bundled together as the columns of a matrix N, called the nullspace matrix. This matrix has two important properties:

- $AN = 0$; and
- Each column of N (i.e. each special solution) holds the coefficients needed to write down a interesting linear combination equation on the columns of A.

Finally, please use some caution when reading through the area with blue boxes and equations (4) and (5) on page 147. Note that Strang introduces a big simplifying assumption that makes his work easier. The general principles will hold, but those nice, neat equations won't always look so good for an arbitrary matrix.

Sage and the Rank

Sage has a built-in command for the rank of a matrix. It is called `.rank()`, of course.

```
A = matrix(QQ, 3,4, [1,2,3,4,5,6,7,8,9,10,11,12])
A
```

```
[ 1  2  3  4]
[ 5  6  7  8]
[ 9 10 11 12]
```

```
A.rank()
```

```
2
```

Sage knows how to do the underlying computations, too. Let's have Sage compute the reduced row echelon form of `A`:

```
A.rref()
```

Nice! Note that this fits the "special case" form that Strang uses on page 147. Let's see what else it can do.

```
A.right_kernel()
```

```
Vector space of degree 4 and dimension 2 over Rational Field
Basis matrix:
[ 1  0 -3  2]
[ 0  1 -2  1]
```

We can make Sage give us just the matrix in that last response.

```
A.right_kernel().basis_matrix()
```

```
[ 1  0 -3  2]
[ 0  1 -2  1]
```

That is *almost* Strang's version of a nullspace matrix in this case. Part of what we need to do is make the rows into columns. The transpose command is the way to do that:

```
N = A.right_kernel().basis_matrix().transpose()
N
```

```
[ 1  0]
[ 0  1]
[-3 -2]
[ 2  1]
```

That is *NOT* exactly Strang's nullspace matrix. But it is very close. In fact, it still has this crucial property:

```
A*N
```

What has happened here is that Sage constructs a *different* version of the nullspace matrix than Strang does. Strang's version is easier to find by hand and tends to have 1's and 0's and the end. The Sage version comes from a routine that strongly prefers to start with 1's and 0's.

The two version of nullspace matrix are related, of course. Let's see how:

```
E = N.matrix_from_rows([2,3]); E
```

```
[−3 −2]
[ 2  1]
```

```
F = N*E.inverse(); F
```

```
[ 1  2]
[−2 −3]
[ 1  0]
[ 0  1]
```

That is Strang's nullspace matrix.

```
A*F
```

```
[0 0]
[0 0]
[0 0]
[0 0]
```

Where does this relationship come from? I have a hint: *Column Operations*. How can we use the structure of matrix multiplication to see why left multiplication by `E.inverse()` performs some column operations?

Here is another thing to think about: where does that `E` come from? Compare `A` and `N`. Notice anything? Remember that Strang wants the nullspace matrix to have 1's and 0's in the free columns.

Exercises

Task 3.13. Give an example of a 5×3 matrix which has

- rank equal to 3, and
- all non-zero entries,

or explain why no such example is possible.

Task 3.14. Give an example of a 3×3 matrix which has

- rank equal to 2,
- a nullspace of $\{0\}$, and
- all non-zero entries,

or explain why no such example is possible.

Task 3.15. Consider the matrix

$$A = \left(\begin{smallmatrix} 2 & 1 & 5 \\ 1 & 1 & 7 \end{smallmatrix}\right).$$

Find the reduced row echelon form $R = \mathrm{rref}(A)$ of A. Track the row operations you use, and use them to find an invertible matrix E so that $EA = R$.

Task 3.16. Contine with the matrix of the last exercise. Find the null space matrix of A. Use the information contained in the nullspace matrix to write down a linear combination equation on the columns of $R = \mathrm{rref}(A)$ of the form

$$a * (\text{column1}) + b * (\text{column2}) + c * (\text{column3}) = 0.$$

Explain why the matrix E allows us to translate this equation into this one on the columns of A:

$$a \begin{pmatrix} 2 \\ 1 \end{pmatrix} + b \begin{pmatrix} 1 \\ 1 \end{pmatrix} + c \begin{pmatrix} 5 \\ 7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Task 3.17. Consider the matrix N given below. Make an example of two different matrices A and B which have different shapes and each have N as nullspace matrix, or explain why such an example is not possible.

$$N = \begin{pmatrix} -3 & 2 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 6 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Task 3.18. Consider the matrix $T = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 1 \end{smallmatrix}\right)$. What is the nullspace matrix of T?

3.4 Solving a System

The Assignment

- Read chapter 3 section 4 of Strang.
- Read the following and complete the tasks below.

Learning Goals

Before class, a student should be able to:

- Identify a particular solution to a matrix-vector equation $Ax = b$. (Provided there is one.)
- Find the complete solution to a matrix-vector equation $Ax = b$ as a parametrized object. (Provided there is one.)

After class, a student should be able to:

- Describe the complete solution to a matrix-vector equation $Ax = b$ as an implicit object, cut out by equations.
- Describe the possibilities for the number of solutions to a matrix-vector equation $Ax = b$ in terms of the shape of the matrix.

Discussion: The Complete Solution to a System of Equations

This is the big day! We finally learn how to write out the general solution to a system of linear equations. We have spent so much time understanding things related to this, that it should go pretty quickly.

The tiny little facts underneath the analysis for this section are these: For a matrix A, vectors v and w and a scalar λ, all chosen so that the equations make any sense,

$$\begin{aligned} A(v+w) &= Av + Aw \\ A(\lambda v) &= \lambda(Av) \end{aligned}$$

The first is a kind of **distributive property**, and the second is a kind of **commutative property**. When taken together, these things say that the operation of "left-multiply by the matrix A" is a special kind of function. The kind of function here is important enough that we have a special word for this combined property: it is called **linearity**. That is, left-multiplication by A is a **linear operation** or a **linear transformation**.

The linearity property makes it possible to check the following two results.

Theorem 3.2. *Let $Ax = b$ be a system of linear equations, and let $Ax = 0$ be the associated homogeneous system. If x_p and x_p' are two particular solutions to $Ax = b$, then $x_p - x_p'$ is a solution to the homogeneous system $Ax = 0$.*

Theorem 3.3. *Let $Ax = b$ be a system of linear equations, and let $Ax = 0$ be the associated homogeneous system. If x_p is some particular solution to $Ax = b$ and x_n is some solution to $Ax = 0$, then $x_p + x_n$ is another solution to $Ax = b$.*

And if we put these two theorems together, we find this result which sounds fancier, but has exactly the same content.

Theorem 3.4. *The complete set of solutions to the system $Ax = b$ is the set*

$$\{x_p + x_n \mid x_n \in \operatorname{null}(A)\},$$

where x_p is any one particular solution to $Ax = b$.

This leads us to Strang's very sensible advice about finding the complete solution:

- Form the augmented matrix $(A \mid b)$ and use Gauss-Jordan elimination to put it in reduced row echelon form $(R \mid d)$.
- Use the information from the RREF to find a particular solution x_p by solving for the pivot variables from the vector d and setting the free variables to zero.
- Use the special solutions $s_1, s_2, \ldots, s_k$ (if any exist!) to describe the nullspace null(A).
- Write down the resulting general solution:

$$x = x_p + a_1 s_1 + a_2 s_2 + \cdots + a_k s_k, \quad \text{for any scalars } a_i \in \mathbb{R}.$$

Sage and Solving General Systems

Sage has many built-in methods for solving systems of linear equations. We will investigate three common ones with a single example considered several times.

```
A = matrix(QQ, 3,4, [1,0,2,3, 1,3,2,0, 2,0,4,9])
b = vector([2,5,10])
print A
print b
```

```
[1 0 2 3]
[1 3 2 0]
[2 0 4 9]
(2, 5, 10)
```

Method One: RREF and the Nullspace

First we find a particular solution.

```
X = A.augment(b, subdivide=True).rref()
X  # this subdivide thing is pretty handy!
```

```
[ 1  0  2  0|-4]
[ 0  1  0  0| 3]
[ 0  0  0  1| 2]
```

This clearly has three pivots, and all belong in the original matrix. So there will be a solution. We pull out the particular solution.

```
xp = vector([-4, 3, 0, 2])
xp
```

```
(-4, 3, 0, 2)
```

Since we typed that in by hand, we should check our work.

```
A*xp == b
```

```
True
```

Now we need to find the nullspace and the special solutions.

```
A.right_kernel()
```

```
Vector space of degree 4 and dimension 1 over Rational Field
Basis matrix:
[   1    0 -1/2    0]
```

The basis has only one row, so there is only one special solution. This matches our expectation. Our system is 3×4 and has rank 3. So there is only one free column, and hence only one special solution.

```
s1 = vector([-2, 0, 1, 0])
A*s1 == 0
```

```
True
```

Now we can check the "general solution".

```
t = var('t')
gensol = xp + t * s1
gensol
```

```
(-2*t - 4, 3, t, 2)
```

```
A*gensol == b
```

```
True
```

Method Two: A Sage built-in

Sage has a built-in method that looks like "Matrix division". Here we "left divide" by the matrix. This is odd notation, and is just something Sage allows.

```
A\b
```

```
(-4, 3, 0, 2)
```

It is weird, but this works even if `A` is not invertible, like now.

```
A.inverse()*b
```

```
Error in lines 2-2
Traceback (most recent call last):
...
ArithmeticError: self must be a square matrix
```

The downside to this particular method is that in only gives you one particular solution. It does not produce the complete solution. You have to do that bit for yourself, maybe like the above.

Method Three: Another Sage Built-in

Finally, Sage will also try to solve the system if you apply the `.solve_right()` method to `A`. You have to supply the vector `b` as an argument to the command.

```
A.solve_right(b)
```

```
(-4, 3, 0, 2)
```

Again, this only pulls out a single particular solution. It is up to you to figure out the rest.

Exercises

Task 3.19. (Strang ex 3.4.4) Find the complete solution (also called the general solution) to

$$\begin{pmatrix} 1 & 3 & 1 & 2 \\ 2 & 6 & 4 & 8 \\ 0 & 0 & 2 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix}.$$

Task 3.20. (Strang ex 3.4.6) What conditions on b_1, b_2, b_3 and b_4 make each of these systems solvable? Find a solution in those cases.

1. $$\begin{pmatrix} 1 & 2 \\ 2 & 4 \\ 2 & 5 \\ 3 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}.$$

2. $$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 2 & 5 & 7 \\ 3 & 9 & 12 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}.$$

Task 3.21. (Strang ex 3.4.11) It is impossible for a 1×3 system of equations to have $x_p = (2, 4, 0)$ and $x_n =$ any multiple of $(1, 1, 1)$. Explain why.

Task 3.22. (Strang ex 3.4.13) Each of the statments below is false. Find a 2×2 counterexample to each one.

1. The complete solution is any linear combination of x_p and X_n.
2. A system $Ax = b$ has at most one particular solution.
3. The solution x_p with all free variables zero is the shortest solution, in that it has the minimum norm $||x_p||$.
4. If A is an invertible matrix, there is no solution x_n in the nullspace.

Task 3.23. (Strang ex 3.4.21) Find the complete solution in the form $x_p + x_n$ to these full rank systems.

1. $$x + y + z = 4$$

2. $$\begin{array}{ccccccc} x & + & y & + & z & = & 4 \\ x & - & y & + & z & = & 4 \end{array}$$

Task 3.24. (Strang ex 3.4.24) Give examples of matrices A for which the number of solutions to $Ax = b$ is

1. 0 or 1, depending on b;
2. ∞, regardless of b;
3. 0 or ∞, depending on b;
4. 1, regardless of b.

Task 3.25. (Strang ex 3.4.31) Find examples of matrices with the given property, or explain why it is impossible:

1. The only solution of $Ax = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ is $x = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
2. The only solution of $Bx = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ is $x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$.

Task 3.26. (Strang ex 3.4.33) The complete solution to the equation $Ax = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ is $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Find the matrix A. Write the set of equations that corresponds to $Ax = b$. (This is the *implicit* description of this set!)

3.5 Going Further with Solving Systems

The Assignment

- Go back through the exercises in the first four sections of this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in the first four sections of Chapter Three: Vector Space and Subspaces. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?

- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.

- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?

- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

3.6 Bases

The Assignment

- Read *Strang* chapter 3 section 5.

- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Use the row space algorithm to decide if a set of vectors is linearly independent or linearly dependent.

- Use the column space algorithm to decide if a set of vectors is linearly independent or linearly dependent

- Compute the dimension of a subspace.

Sometime after class, a student should be able to:

- Explain the connection between a set of vectors being linearly independent, a spanning set, and a basis.

Discussion: Linear Independence, Spanning, Basis, and Dimension

The purpose of this lesson is to introduce specific terms for several concepts we have been dancing around. This is a spot that sometimes gives students difficulty because they are unused to the way mathematicians talk. So, here is a big warning:

> "When I use a word," Humpty Dumpty said in rather a scornful tone,
> "it means just what I choose it to mean – neither more nor less."
>
> –*Through the Looking Glass*, Lewis Carroll

That is the essence of it. We have several new words, and they will mean *exactly* what we declare them to mean, no more, no less.

What are the new terms?

- A set $\{v_1, v_2, \ldots, v_k\}$ of vectors is **linearly independent**, or **linearly dependent**.

- A set of vectors **spans** a vector space or a subspace.

- A set of vectors is a **basis** or not.

- The **dimension** of a vector space, or of a vector subspace.

You should take away from this reading what those four terms are, how to check them, and some examples and non-examples.

A bit of notation: If we have a set $\{v_1, v_2, \ldots, v_k\}$ of vectors in some vector space, then we denote the subspace which they span by $\mathrm{span}(\{v_1, v_2, \ldots, v_k\})$. (This has to be the easiest possible notational choice ever.)

Other Vector Spaces

Once you understand these terms as they apply to the Euclidean spaces $\mathbb{R}^n$ and their subspaces (especially those associated to matrices), you should pause to admire your achievement.

But next, realize that those terms apply generally to all sorts of vector spaces! Can you make examples and non-examples in some of these other situtations?

- The set $\mathcal{P}_3$ of polynomials in x of degree no more than 3?
- The set $M_{m,n}$ of $m \times n$ matrices.
- The set $\mathcal{C}(\mathbb{R})$ of continuous functions.
- The set of all functions which are solutions to the differential equation $y'' = y$.

Two Methods of Sorting out Linear Independence

We have enough information to collect two ways to answer the question: "Is this set linearly independent?" Well, at least when working with vectors in some Euclidean space $\mathbb{R}^n$.

The Column Space Algorithm
Given a set of vectors $\{v_1, v_2, \dots, v_k\}$ from $\mathbb{R}^n$:

- Form the $n \times k$ matrix $A = \left(\begin{smallmatrix} v_1 & v_2 & \cdots & v_k \end{smallmatrix} \right)$.
- Put A into reduced row echelon form $R = \mathrm{rref}(A)$. (Really, you only need to go to echelon form, here.)
- Read out pivot columns and free columns of R. Those columns of A which are free columns are linear combinations of previous columns to the left! So, if any column of R (and A) is a free columns, the set of vectors is linearly dependent. If all of the columns of R (and A) are pivot columns, then the set is linearly independent.

This method is particularly good for identifying which subset of our original set of vectors would form a basis of the vector subspace $\mathrm{span}(\{v_1, v_2, \dots, v_k\})$.

The Row Space Algorithm (We will see the reason for the name soon.) Given a set of vectors $\{v_1, v_2, \dots, v_k\}$ from $\mathbb{R}^n$:

- Form the $k \times n$ matrix $A = \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{pmatrix}$.
- Put A into reduced row echelon form $R = \mathrm{rref}(A)$.
- The rows of R will contain a basis for the vector subspace $\mathrm{span}(\{v_1, v_2, \dots, v_k\})$ (written as rows instead of columns, of course). If R has any zero rows, the original set was linearly dependent, otherwise it was linearly independent.

This method is good at picking out a simple basis of the vector subspace $\mathrm{span}(\{v_1, v_2, \dots, v_k\})$, but the resulting vectors *probably won't come from your original set.*

Sage and Bases

Sage has several commands which are useful for dealing with the concpets of this section.

Commands for span, dimension, and basis

This first command will construct a vector subspace of 3-space which is spanned by the two vectors we pass in as arguments.

```
v1 = vector([1,0,2])
v2 = vector([1,1,3])

V = span([v1,v2], QQ)
V
```

```
Vector space of degree 3 and dimension 2 over Rational Field
Basis matrix:
[1 0 2]
[0 1 1]
```

Note that Sage already gives us some information:

- dimension
- a basis

Sage has chosen its preferred basis, as usual.

```
V.dimension()
```

```
V.basis()
```

```
[
(1, 0, 2),
(0, 1, 1)
]
```

The Algorithms for Checking Linear Independence

Sage does not have built-in commands with names for checking linear independence or linear dependence. Instead, you have to just use the algorithms.

The Row Space Algorithm

```
row_matrix = matrix(QQ, 2,3, [v1,v2])
row_matrix
```

```
[1 0 2]
[1 1 3]
```

```
R = row_matrix.rref()
R
```

```
[1 0 2]
[0 1 1]
```

(Where have we seen that before?) From this, it is clear that $\{v_1, v_2\}$ is linearly dependent because both rows have pivots.

The Column Space Algorithm

We work much the same way here.

```
col_matrix = matrix(2,3, [v1,v2]).transpose()
col_matrix
```

```
[1 1]
[0 1]
[2 3]
```

```
col_matrix.rref()
```

```
[1 0]
[0 1]
[0 0]
```

This tells us that our first two columns are pivot columns, so we should keep those as part of our basis for span (v_1, v_2).

Spaces Associated to a Matrix

Sage knows about the column space and row space associated to a matrix. For this next example, we will work over the ring `AA` of "algebraic numbers", so we can include $\sqrt{2}$.

```
entries = [453, 1/3, 34, 2.sqrt(),
           9, 11/9, -3, 8, 98, 10,
           21, -4]
A = matrix(AA, 3,4, entries); A
```

```
[               453                1/3                 34
    1.414213562373095?]
[                 9               11/9                 -3
                     8]
[                98                 10                 21
                    -4]
```

```
A.column_space()
```

```
Vector space of degree 3 and dimension 3 over Algebraic Real Field
Basis matrix:
[1 0 0]
[0 1 0]
[0 0 1]
```

```
A.column_space().dimension()
```

```
3
```

```
A.column_space().basis()
```

```
[
(1, 0, 0),
(0, 1, 1),
(0, 0, 1)
]
```

```
A.row_space()
```

```
Vector space of degree 4 and dimension 3 over Algebraic Real Field
Basis matrix:
[                   1                    0                    0
    0.12115291228302765?]
[                   0                    1                    0
    1.751194192226904?]
[                   0                    0                    1
    -1.589758444095512?]
```

```
A.row_space().dimension()
```

```
3
```

Or we could do this in other ways.

```
span(A.columns()).dimension()
```

```
3
```

```
span(A.rows()).dimension()
```

```
3
```

```
A.right_kernel().dimension()
```

```
1
```

Exercises

Note: We may not present all of these in class.

Task 3.27. Make an example of a vector $v \in \mathbb{R}^4$ so that the set

$$\left\{ \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \\ 0 \\ -5 \end{pmatrix}, v \right\}$$

is a linearly independent set, or explain why it is impossible to find such an example. Is your resulting set a basis?

Task 3.28. Make an example of a vector $w \in \mathbb{R}^3$ so that the set

$$\left\{ \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ -4 \\ -5 \end{pmatrix}, w \right\}$$

is a spanning set, or explain why it is impossible to find such an example. Is your resulting set a basis?

Task 3.29. (Strang 3.5.10) Find two independent vectors on the plane $x + 2y - 3z - t = 0$ in $\mathbb{R}^4$. Then find three independent vectors on this plane. Why not four? Find a matrix for which this plane is the nullspace.

Task 3.30. (Strang 3.5.21) Suppose that the columns of a 5×5 matrix A are a basis for $\mathbb{R}^5$.

1. Explain why the equation $Ax = 0$ has only the solution $x = 0$.
2. What fact about the column vectors guarantees that for any b in $\mathbb{R}^5$ the equation $Ax = b$ is solvable?

Conclusion: A is invertible. Its rank is 5. Its rows are also a basis for $\mathbb{R}^5$.

Task 3.31. (Strang 3.5.22) Suppose that S is a 5-dimensional subspace of $\mathbb{R}^6$. Determine if the following statements are true or false. If true, give an explanation for why it is true. If false, give a counterexample.

1. Every basis for S can be extended to a basis for $\mathbb{R}^6$ by adding one more vector.
2. Every basis for $\mathbb{R}^6$ can be reduced to a basis for S by removing one vector.

Task 3.32. (Strang 3.5.26) Find a basis (and the dimension) for each of these subspaces of the vector space of 3×3 matrices:

1. All diagonal matrices.
2. All symmetric matrices. ($A^T = A$)
3. All skew-symmetric matrices. ($A^T = -A$)

Task 3.33. (Strang 3.5.35) Find a basis for the space of polynomials $p(x)$ of degree less than or equal to 3. Find a basis for the subspace with $p(1) = 0$.

3.7 The Four Subspaces

The Assignment

- Read chapter 3 section 6 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Identify the four subspaces associated to a matrix by giving a basis of each.
- Determine the dimension of each of the four subspaces associated to a matrix.

Some time after class, a student should be able to:

- Describe each of the four subspaces associated to a matrix by giving minimal sets of equations which "cut them out."
- State and use the Fundamental Theorem of Linear Algebra (FTLA) to reason about matrices.
- Use the RREF of a matrix to explain why the FTLA is true.

Discussion: The Four Subspaces

This section summarizes a big tool for understanding the behavior of a matrix as a function. Recall that if A is an $m \times n$ matrix, then we can think of it as defining a function

$$\begin{aligned} T_A : \mathbb{R}^n &\to \mathbb{R}^m \\ v &\mapsto Av \end{aligned}$$

which takes as inputs vectors from $\mathbb{R}^n$ and has as outputs vectors in $\mathbb{R}^m$. We have also seen that properties of matrix multiplication translate into properties that make into a **linear transformation**.

We now have four fundamental subspaces associated to the matrix A.

- The column space, $\mathrm{col}(A)$, spanned by all of the columns of A. This is a subspace of $\mathbb{R}^m$.
- The row space, $\mathrm{row}(A)$, spanned by all of the rows of A. This is a subspace of $\mathbb{R}^n$. This also happens to be the column space of A^T.
- The nullspace (or kernel), $\mathrm{null}(A)$, consisting of all those vectors x for which $Ax = 0$. This is a subspace of $\mathbb{R}^n$.
- The left nullspace, which is just the nullspace of A^T. This is a subspace of $\mathbb{R}^m$.

And we have a big result:

Theorem 3.5. *If A is an $m \times n$ matrix with rank $\mathrm{rank}(A) = r$, then*

- $\dim(\mathrm{col}(A)) = \dim(\mathrm{row}(A)) = r$*,*
- $\dim(\mathrm{null}(A)) = n - r$*, and*
- $\dim(\mathrm{null}(A^T)) = m - r$*.*

(*Study Hint: Write that out in English, with no notation. It will help you remember it.*)

We will have more to say about these spaces when we reconsider the uses of the dot product in chapter 4.

Sage and the Four Subspaces

We have already seen enough Sage commands to work with the four subspaces: `.row_space()`, `.column_space()`, `.left_kernel()`, and `.right_kernel()` all work. Alternatively, we need only remember that the left nullspace and the row space are just the nullspace and column space of the transpose. Let us take a look at some of the options.

```
A = matrix(AA, 2,5, [3,4,5,-1,-1, 1,1,2,-1,1])
A
```

```
[ 3  4  5 -1 -1]
[ 1  1  2 -1  1]
```

```
A.column_space()
```

```
Vector space of degree 2 and dimension 2 over Algebraic Real Field
Basis matrix:
[1 0]
[0 1]
```

```
A.right_kernel()
```

```
Vector space of degree 5 and dimension 3 over Algebraic Real Field
Basis matrix:
[   1    0    0    2    1]
[   0    1    0  5/2  3/2]
[   0    0    1  7/2  3/2]
```

We have computed column spaces and nullspaces before. What about our new friends?

```
A.row_space()
```

```
Vector space of degree 5 and dimension 2 over Algebraic Real Field
Basis matrix:
[ 1  0  3 -3  5]
[ 0  1 -1  2 -4]
```

```
A.transpose().column_space()
```

```
Vector space of degree 5 and dimension 2 over Algebraic Real Field
Basis matrix:
[ 1  0  3 -3  5]
[ 0  1 -1  2 -4]
```

```
A.transpose().column_space() == A.row_space()
```

```
True
```

Since Sage prefers rows under the hood, the left nullspace is easy to find.

```
A.left_kernel()
```

```
Vector space of degree 2 and dimension 0 over Algebraic Real Field
Basis matrix:
[]
```

```
A.transpose().right_kernel()
```

```
Vector space of degree 2 and dimension 0 over Algebraic Real Field
Basis matrix:
[]
```

```
A.left_kernel() == A.transpose().right_kernel()
```

```
True
```

And there you have it. Sage can construct all four fundamental subspaces, and each comes with a basis computed by Sage. (Using the row algorithm!) Note that the FTLA works in this case.

Questions for Section 3.6

Task 3.34. Find the four subspaces, including a basis of each, for the matrix

$$A = \begin{pmatrix} 7 & -1 & 3 \\ -2 & 4 & -5 \\ 1 & 11 & -12 \end{pmatrix}.$$

Task 3.35. Find the four subspaces, including a basis of each, for the matrix

$$B = \begin{pmatrix} 1 & 3 & -2 & 0 & 2 & 0 \\ 2 & 6 & -5 & -2 & 4 & -3 \\ 0 & 0 & 5 & 10 & 0 & 15 \\ 2 & 6 & 0 & 8 & 4 & 18 \end{pmatrix}.$$

Task 3.36. (Strang 3.6.12)

Find an example of a matrix which has $(1, 0, 1)$ and $(1, 2, 0)$ as a basis for its row space and its column space.

Why can't this be a basis for the row space and the nullspace?

Task 3.37. (Strang 3.6.14) Without computing A, find bases for its four fundamental subspaces:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 6 & 1 & 0 \\ 9 & 8 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \end{pmatrix}.$$

Task 3.38. (Strang 3.6.16) Explain why the vector $v = (1, 0, -1)$ cannot be a row of A and also in the nullspace of A.

Task 3.39. (Strang 3.6.24)

The equation $A^T y = d$ is solvable exactly when d lies in one of the four subspaces associated to A. Which is it?

Which subspace can you use to determine if the solution to that equation is unique? How do you use that subspace?

3.8 Going Further with the Fundamental Theorem

The Assignment

- Go back through the exercises in the last two sections of this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in the first four sections of Chapter Three: Vector Space and Subspaces. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

Chapter 4

Orthogonality

We introduced the dot product in Chapter One, but we have not found a lot of use for it so far. In this chapter, we pick it back up and see what we can do with it. Of course, the dot product hides a lot of geometry in its mysteries, but for us the key is the notion of *orthogonality*.

In this chapter we will address two important questions:

1. How can we find *approximate solutions* to equations $Ax = b$ when we know (or suspect) that finding a true solutions is impossible?

2. How can we use geometry to find a *good* basis for a subspace?

The answers to both of these questions will involve using the dot product to check for orthogonality. This will be leveraged into the technique of **orthogonal projection.**

Now, to begin, we shall explore the idea of *subspaces* being orthogonal, rather than just vectors, and strengthen the Fundamental Theorem of Linear Algebra.

4.1 Orthogonality and the Four Subspaces

The Assignment

- Read section 4.1 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Find the four subspaces associated to a matrix and verify that they are orthogonal subspaces.
- Draw the "Big Picture" associated to a matrix as a schematic drawing, with the four subspaces properly located and their dimensions identified.

Some time after class, a student should be able to:

- Find the orthogonal complement to a subspace.
- Reason about the structure of a matrix as a transformation using information about its four subspaces.

Discussion: Orthogonality for subspaces

Previously, we had the notion of orthogonality for two vectors in Euclidean space. In this section, the concept gets extended to subspaces.

Definition 4.1. Let V and W be subspaces of $\mathbb{R}^n$. We say that $\mathbb{R}^n$ and $\mathbb{R}^m$ are **orthogonal** when for each vector $v \in V$ and each vector $w \in W$ we have $v \cdot w = 0$.

Two orthogonal subspaces always have as intersection the trivial subspace $\{0\}$. The reason for this is that if some vector x lay in both V and W, then we must have that $x \cdot x = 0$. (Think of the first x as lying in V, and the second in W.) But the properties of the dot product then mean that x is the zero vector.

There is a further concept:

Definition 4.2. Let V be a vector subspace of $\mathbb{R}^n$. The **orthogonal complement** of V is the set

$$V^{\perp} = \{w \in \mathbb{R}^n \mid w \cdot v = 0 \text{ for all } v \in V\}.$$

The basic idea is that two spaces are orthogonal complements if they are orthogonal, and together they contain enough vectors to span the entire space. The definition looks like it is a one-directional thing: for a subspace, you find its orthogonal complement. But really it is a *complementary* relationship. If W is the orthogonal complement to V, then V is the orthogonal complement to W.

Recall the four fundamental subspaces associated to an $m \times n$ matrix A.

The column space, $\mathrm{col}(A)$, spanned by all of the columns of A. This is a subspace of $\mathbb{R}^m$.

The row space, $\mathrm{row}(A)$, spanned by all of the rows of A. This is a subspace of $\mathbb{R}^n$. This also happens to be the column space of A^T.

The nullspace (or kernel), $\mathrm{null}(A)$, consisting of all those vectors x for which $Ax = 0$. This is a subspace of $\mathbb{R}^n$.

The left nullspace, which is just the nullspace of A^T. This is a subspace of $\mathbb{R}^m$.

And we have another big result, which is a sharpening of the Fundamental Theorem of Linear Algebra from the end of Chapter Three.

Theorem 4.3. *If A is an $m \times n$ matrix, then*

- *The nullspace of A and the row space of A are orthogonal complements of one another.*
- *The column space of A and the left nullspace of A are orthogonal complements of one another.*

Sage and the Orthogonal Complement

It is not hard to find the four subspaces associated to a matrix with Sage's built-in commands. But Sage also has a general purpose `.complement()` method available for vector subspaces which can be used.

```
A = matrix(QQ, 3,2, [12,3,4,5,6,7])
A
```

```
[12 3]
[ 4 5]
[ 6 7]
```

```
A.right_kernel() # the nullspace
```

```
Vector space of degree 2 and dimension 0 over Rational Field
Basis matrix:
[]
```

```
A.column_space()
```

```
Vector space of degree 3 and dimension 2 over Rational Field
Basis matrix:
[   1    0 1/24]
[   0    1 11/8]
```

```
A.row_space()
```

```
Vector space of degree 2 and dimension 2 over Rational Field
Basis matrix:
[1 0]
[0 1]
```

Those are the easy ones to find. The left nullspace is just a touch trickier. What is the deal? It is just the nullspace of the matrix A^T, of course. But by the Fundamental Theorem of Linear Algebra, the left nullspace is the orthogonal complement of the column space. Let's see if they agree.

```
A.column_space().complement()
```

```
Vector space of degree 3 and dimension 1 over Rational Field
Basis matrix:
[  1  33 -24]
```

```
A.transpose().right_kernel()
```

```
Vector space of degree 3 and dimension 1 over Rational Field
Basis matrix:
[  1  33 -24]
```

```
A.left_kernel()
```

```
Vector space of degree 3 and dimension 1 over Rational Field
Basis matrix:
[  1  33 -24]
```

That is good news! It looks like the three ways we have of computing the left nullspace agree. As a check for understanding, you should be able to ask Sage if the rowspace of A is the orthogonal complement to the nullspace of A.

Exercises

Task 4.1. (Strang ex. 4.1.2) Draw the "Big Picture" for a 3×2 matrix of rank 2. Which subspace has to be the zero subspace?

Task 4.2. (Strang ex. 4.1.3) For each of the following, give an example of a matrix with the required properties, or explain why that is impossible.

1. The column space contains $\begin{pmatrix} 1 \\ 2 \\ -3 \end{pmatrix}$ and $\begin{pmatrix} 2 \\ -3 \\ 5 \end{pmatrix}$, and the nullspace contains $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$.

2. the row space contains $\begin{pmatrix}1\\2\\-3\end{pmatrix}$ and $\begin{pmatrix}2\\-3\\5\end{pmatrix}$, and the nullspace contains $\begin{pmatrix}1\\1\\1\end{pmatrix}$.
3. $Ax = \begin{pmatrix}1\\1\\1\end{pmatrix}$ has a solution, and $A^T\begin{pmatrix}1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix}$.
4. Every row is orthogonal to every column, but A is not the zero matrix.
5. The columns add up to a column of zeros, and the rows add up to a row of 1's.

Task 4.3. (Strang ex 4.1.4) If $AB = 0$, how do the columns of B relate to the subspaces of A? How do the rows of A relate to the subspaces of B? Why can't we make an example of this where A and B are both 3×3 matrices of rank 2?

Task 4.4. (Strang ex 4.1.9) Use the four subspace of A to explain why this is always true:

$$\text{If } A^TAx = 0, \text{then } Ax = 0.$$

(This fact will be useful later! It will help us see that A^TA and A have the same nullspace.)

Task 4.5. (Strang ex 4.1.12) Find the pieces x_r and x_n and draw the "Big Picture" carefully when:

$$A = \begin{pmatrix}1 & -1\\0 & 0\\0 & 0\end{pmatrix} \qquad \text{and } x = \begin{pmatrix}2\\0\end{pmatrix}.$$

Task 4.6. (Strang ex. 4.1.22) Let P be the hyperplane of vectors in $\mathbb{R}^4$ which satisfy the equation

$$x_1 + x_2 + x_3 + x_4 = 0.$$

Write down a basis for $P^\perp$. Construct a matrix X which has P as its nullspace.

4.2 Projections Onto Subspaces

The Assignment

- Read Chapter 4 section 2 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Compute the projection of a vector onto a line.
- Find the projection matrix which computes the projections of vectors onto a given line.
- Draw the schematic picture of a projection: the line, the vector, the projected vector, and the difference.

Sometime after class, a student should be able to:

- Compute the projection of a vector onto a subspace.
- Find the projection matrix which computes the projections of vectors onto a given subspace.
- Explain the process for finding the equations which determine the projection matrix, and say why the transpose makes an appearance.

Discussion: Orthogonal Projections

One good use of the geometry in $\mathbb{R}^n$ is the concept of orthogonal projection. The basic idea is to mimic the behavior of shadows under sunlight. Our everyday experience leads us to thinking about the projection of a vector onto a plane (the ground–its roughly a plane), but if you imagine holding out a pencil you can summon up the visual of projection onto a line, too.

The key concept is to use the basic condition of orthogonality ($u \cdot v = 0$) to figure things out.

Note that everything in this section is done by projecting onto *subspaces*! This is a bit of a restriction. In practice, this restriction can be removed by translating your whole problem to have a new origin.

Sage and Orthogonal Projection

Sage has no built-in commands for orthogonal projections. But let us recall those parts of Sage that will be useful right now:

```
A = matrix(QQ, 3,2, [1,2,3,4,5,6])
A
```

```
A.transpose()
```

```
[1 3 5]
[2 4 6]
```

```
A.inverse()
```

```
Error in lines 1-1
...
ArithmeticError: self must be a square matrix
```

Sorry, that matrix isn't even square, so it can't be invertible. But this will be:

```
(A.transpose()*A).inverse()
```

```
[  7/3 -11/6]
[-11/6 35/24]
```

Finally, this makes sense:

```
P = A * (A.transpose()*A).inverse() * A.transpose()
P
```

```
[ 5/6  1/3 -1/6]
[ 1/3  1/3  1/3]
[-1/6  1/3  5/6]
```

This process should have some basic properties. Let's check them.

```
B = A.transpose()*A
print B.is_invertible()
print B.is_symmetric()
print B.parent()
```

```
True
True
Full MatrixSpace of 2 by 2 dense matrices over Rational Field
```

So $A^T A$ is square, symmetric, and invertible.

```
print P.transpose() == P
print P.is_symmetric()
print P*P == P
```

```
True
True
True
```

Also as expected.

Exercises

Task 4.7. Find the projection matrix which computes projections of vectors in $\mathbb{R}^2$ onto the line $3x + 2y = 0$. (Since it goes through zero, it is a subspace.)

Find the orthogonal projection of the vector $(17, 3)$ onto this line.

Task 4.8. Find the projection matrix which computes projections of vectors in $\mathbb{R}^3$ onto the line which is the intersection of the planes $x - 2y + 3z = 0$ and $y + 2z = 0$. (Again, that is a subspace.)

Find the orthogonal projection of the vector $(1, 1, 1)$ onto this line.

Task 4.9. Find the projection matrix which computes projections of vectors in $\mathbb{R}^3$ onto the plane $-2x + y + 3z = 0$.

Find the orthogonal projection of the vector $(9, 7, -5)$ onto this plane.

Task 4.10. Find the projection matrix which computes projections of vectors in $\mathbb{R}^4$ onto the plane which is the intersectoin of $5x+y+w = 0$ and $z+y+z+w = 0$. (This subspace is the 2 dimensional plane where these two 3-dimensional hyperplanes meet.)

Find the orthogonal projection of the vector $(-3, 1, -3, 1)$ on this plane.

4.3 Going Further: Projections

The Assignment

- Go back through the exercises in this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in the first two sections of Chapter Four. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

4.4 Approximate Solutions: Least Squares

The Assignment

- Read section 4.3 of Strang.
- Read the discussion below.
- Complete exercises 1-11 from section 4.3 in Strang.
- Prepare the items in the exercises for presentation.

Discussion: Least Squares Approximation

(It is probably best to read this after you read the section in Strang.)

Some Perspective

Scientific problems often come down to something as simple as this: make a bunch of observations, and then try to fit those observations with some sort of model for greater understanding.

But data found in scientific problems is often noisy, or infected with error in some way. This leads researchers to gather *more* data so that chance variations and small errors might get smoothed out. How might we fit a curve to a lot of data? Lots of data points means that we likely have too many points to have a curve of our specified model type actually hit all of those points.

For example, fitting a line to five points is already problematic: any two points gives us a line, and there is no reason to believe that the other three points will all sit on that line.

If we set the problem up as a system of equations, things go like this: We have a bunch of data set up as input-output pairs $\{(a_i, y_i)\}$; we are looking for a function f which has a specified type (linear, quadratic, exponential, etc.) that passes through those points.

- Each data point leads us to an equation $f(a_i) = y_i$.
- The modelling function f has some parameters in it, and we want to find the best value of those parameters so that the curve "fits" the data well. These parameters are the unknowns in our equations.

This is generally a challenging problem. The method of least squares is a technique for solving it when the resulting equations make a linear system.

Some History

Gauss discovered the technique described in this section in the late 1790's. In 1801 he used it to help astronomers calculate the orbit of the newly discovered asteroid Ceres, and thus find it after it re-emerged from behind the sun.

See how the pattern fits? Several weeks worth of data about the position of Ceres was known, but it surely had measurement errors in it. Since the time of Kepler (Newton), we have known that the motion of the asteroid must be an ellipse. This is a simple equation with only a few parameters (the coefficients of the equation defining the ellipse). So, the question confronting Gauss was this: find the ellipse which best fits the data.

But plugging all the data into the correct model shape (a conic!) leads to a rather large system of linear equations where the unknowns are the coefficients we seek.

So, what is really happening here?

In the end, we get a system of the form $Ax = y$. Here A is an $m \times n$ matrix and ym is an n-vector, where m is the number of equations and n is the number of parameters we must find. Typically, m is much larger than n, so the matrix A is tall and skinny.

So the system likely has no solution. Instead, we will find the orthogonal projection $\hat{y}$ of y onto the column space $\mathrm{col}(A)$ of A, and then solve $Ax = \hat{y}$. That's the secret. Since we have already mastered projections, this is no big deal.

Sage instructions

There are no new commands for dealing with matrices here, as we already have all that we need. If you are interested, Sage does have a built-in function called `find_fit`.

```
Data = [[1,2], [-1,3], [4,1],[2,1],[1,.5]]
a, b, x = var('a b x')
model(x) = a*x + b

best = find_fit(Data, model, solution_dict=True)
best
```

```
{b: 2.030303030305277, a: -0.37878787879088605}
```

Just to check that, let's plot the data and the curve.

```
curve = model.subs(best)
plot(curve, (x,-1,6)) + points(Data, color='red', size=20)
```

That is not so terrible. Keep in mind that, from a linear algebra perspective, we just found the projection of $y = (2, 3, 1, 1, .5) \in \mathbb{R}^5$ on the column space of the matrix

$$A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 4 & 1 \\ 2 & 1 \\ 1 & 1 \end{pmatrix},$$

which is a 2-dimensional plane in that 5-dimensional space. If I could, I would draw that picture, but five dimensions in challenging.

Exercises

The best thing you can do to understand this is work some examples. Do Strang 1 - 11 from section 4.3. We will present these:

Task 4.11. Exercise 1 from section 4.3 of Strang.

Task 4.12. Exercise 5 from section 4.3 of Strang.

Task 4.13. Exercise 6 from section 4.3 of Strang.

Task 4.14. Exercise 7 from section 4.3 of Strang.

Task 4.15. Exercise 8 from section 4.3 of Strang.

Task 4.16. Exercise 9 from section 4.3 of Strang.

Task 4.17. Exercise 10 from section 4.3 of Strang.

Task 4.18. Exercise 11 from section 4.3 of Strang.

4.5 Orthonormal Bases and Gram-Schmidt

The assignment

- Read section 4.4 of *Strang*.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to:

- Correctly decide if a matrix is orthogonal or not.
- Use the Gram-Schmidt algorithm to turn a basis into an orthonormal basis.

Some time after class, a student should be able to:

- Find the QR decomposition of a matrix.
- Describe the Gram-Schmidt process geometrically using orthogonal projections.

Discussion: Gram-Schmidt

There are four main points to take away from this section:

- The idea of an orthonormal basis.
- The idea of an orthogonal matrix. The special property that $Q^T = Q^{-1}$ for an orthogonal matrix Q.
- The Gram-Schimdt algorithm for constructing an orthonormal basis.
- The QR decomposition of a matrix A.

Sage and the QR decomposition

Sage has a built-in command to find the QR decomposistion of a matrix. Essentially, it does the Gram-Schmidt algorithm under the hood.

If you check the documentation, you will see that the matrix has to be defined over a special type of ring, so use `QQbar`.

```
A = matrix(QQbar, 4,4,
           [1,-1,0,0, 0,1,-1,0,
           0,0,1,-1, 1,1,1,1])
A
```

```
[ 1 -1  0  0]
[ 0  1 -1  0]
[ 0  0  1 -1]
[ 1  1  1  1]
```

```
Q, R = A.QR()
Q
```

```
[ 0.7071067811865475? -0.5773502691896258? -0.3162277660168379?
    -0.2581988897471611?]
[                   0  0.5773502691896258? -0.6324555320336758?
    -0.516397779494323?]
[                   0                    0  0.6324555320336758?
    -0.774596669241484?]
[ 0.7071067811865475?  0.5773502691896258?  0.3162277660168379?
    0.2581988897471611?]
```

This matrix should be an orthogonal matrix. Let's check that.

```
Q*Q.transpose()
```

```
[1.000000000000000?           0.?e-17           0.?e-16
    0.?e-17]
[          0.?e-17 1.000000000000000?           0.?e-16
    0.?e-17]
[          0.?e-16           0.?e-16 1.000000000000000?
    0.?e-16]
[          0.?e-17           0.?e-17           0.?e-16
    1.000000000000000?]
```

That is machine language for "I am pretty sure that's the identity." Those question marks are for machine precision representation of exact numbers.

```
R
```

```
[  1.414213562373095?             0.?e-18  0.7071067811865475?
    0.7071067811865475?]
[                   0   1.732050807568878?             0.?e-17
    0.5773502691896258?]
[                   0                    0   1.581138830084190?
    -0.3162277660168379?]
[                   0                    0                    0
    1.032795558988645?]
```

```
Q*R
```

```
[ 1.000000000000000? -1.000000000000000?            0.?e-17
                0.?e-16]
[                  0  1.000000000000000? -1.000000000000000?
                0.?e-16]
[                  0                   0  1.000000000000000?
    -1.000000000000000?]
[ 1.000000000000000?  1.000000000000000?  1.000000000000000?
    1.000000000000000?]
```

```
Q*R == A
```

```
True
```

Exercises

Task 4.19. (Strang ex. 4.4.12) If a_1, a_2, a_3 is a basis for $\mathbb{R}^3$, then any vector b can be written as

$$b = x_1a_1 + x_2a_2 + x_3a_3$$

or

$$\begin{pmatrix} | & | & | \\ a_1 & a_2 & a_3 \\ | & | & | \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = b.$$

1. Suppose the a_i's are orthonormal. Show that $x_1 = a_1^T b$.
2. Suppose the a_i's are orthogonal. Show that $x_1 = a_1^T b / a_1^T a_1$.
3. If the a_i's are independent, x_1 is the first component of something times b. What is the something?

Task 4.20. (Strang ex. 4.4.18) Find orthogonal vectors A, B, C by Gram-Schmidt from a, b, c:

$$a = (1, -1, 0, 0), \quad b = (0, 1, -1, 0), \quad c = (0, 0, 1, -1).$$

Note that both of these collections are bases for the hyperplane perpendicular to $(1, 1, 1, 1)$.

Task 4.21. (Strang ex. 4.4.19) If $A = QR$, then check that $A^T A$ is the same thing as $R^T R$. What can we say about the shape of the matrix R? Note that this means that Gram-Schmidt on A corresponds to *elimination on* $A^T A$! The pivots for $A^T A$ must be the squares of the diagonal entries of R. Find Q and R by Gram-Schmidt for this A:

$$A = \begin{pmatrix} -1 & 1 \\ 2 & 1 \\ 2 & 4 \end{pmatrix}.$$

Compare with the structure of this matrix:

$$A^T A = \begin{pmatrix} 9 & 9 \\ 9 & 18 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 9 & 0 \\ 0 & 9 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Task 4.22. (Strang ex. 4.4.21) Find an orthonormal basis for the column space of A, and then compute the projection of b onto that column space.

$$A = \begin{pmatrix} 1 & -2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 3 \end{pmatrix} \qquad \text{and} \qquad b = \begin{pmatrix} -4 \\ -3 \\ 3 \\ 0 \end{pmatrix}.$$

Task 4.23. (Strang ex. 4.4.23) Let a, b, c be the columns of the matrix

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 0 & 0 & 5 \\ 0 & 3 & 6 \end{pmatrix}$$

Note that these vectors are linearly independent, so make a basis for $\mathbb{R}^3$. Find an orthonormal basis q_1, q_2, q_3 and express the vectors as linear combinations of a, b, c. Finally, write A as QR.

Task 4.24. (Strang ex. 4.4.24)

1. Find a basis for the subspace S in $\mathbb{R}^4$ spanned by all solutions of

$$x_1 + x_2 + x_3 - x_4 = 0.$$

2. Find a basis for the orthogonal complement $S^\perp$.
3. Find b_1 in S and b_2 in $S^\perp$ so that $b_1 + b_2 = b = (1, 1, 1, 1)$.

4.6 Going Further: Least Squares and Gram-Schmidt

The Assignment

- Go back through the exercises in this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in the last two sections of Chapter Four. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

Chapter 5

Determinants

Our next goal is to find a better understanding of the **determinant** of a square matrix. We have seen this come up before as a measurement that allows us to "determine" if a square matrix is invertible. But it has some more information in it than that, and it has some interesting properties.

Our first section will take up the interesting properties, with special attention to those properies which make the determinant easier to compute. Then we will look at a few alternate methods of computation.

5.1 Properties of the Determinant

The Assignment

- Read section chapter 5 section 1 of Strang.
- Read the following and complete the exercises below.

Learning Goals

At some point, a student should be able to compute the determinant of a square matrix using the properties outlined here.

Discussion: The Determinant

Generally, the most interesting matrices to look at are the square ones. For square matrices, there is an important number called the **determinant** which helps us determine if the matrix is invertible or not.

Strang lists 10 important properties of determinants in this section, and verifies them for 2×2 matrices. The verifications for general matrices aren't any harder, but they sure are longer, so I am glad he skipped them. Anyway, these properties are enough to get by when it is time to compute. In fact, clever use of these properties can save you a lot of time.

1. The determinant of the identity matrix is always one: $\det(I) = 1$.
2. If we exchange two rows of a matrix, the determinant changes sign.
3. The determinant is linear in each row separately. (The fancy new word here is that det is a **multilinear** function.)
4. If two rows are equal, then the determinant is zero.

5. The row operation of "add a multiple of one row to another row" does not change the determinant of a matrix.
6. If a matrix has a row of zeros, then its determinant is zero.
7. If A is triangular, then $\det(A)$ is the product of the diagonal entries of A.
8. A matrix is singular if its deterimant is zero. A matrix is invertible exactly when its determinant is non-zero.
9. The determinant is a **multiplicative** function at the level of matrices: $\det(AB) = \det(A)\det(B)$.
10. The determinant of a matrix and its transpose are the same.

That last property can be helpful in a variety of ways: it allows us to translate all of those statements about rows into statements about columns!

Interpretation

Suppose that an $n \times n$ matrix A is represented as a collection of its column vectors:

$$A = \begin{pmatrix} | & | & & | \\ v_1 & v_2 & \dots & v_n \\ | & | & & | \end{pmatrix}.$$

Then the geometric significance of the determinant is this: The number $\det(A)$ represents the signed n-dimensional volume of the n-dimensional box in $\mathbb{R}^n$ with sides $v_1, v_2, \dots, v_n$.

This takes a bit of getting used to, and the hardest part is the choice of signs. We choose a positive sign if the vectors $v_1, v_2, \dots, v_n$ have the same orientation as the standard basis.

Sage and the Determinant

Sage has a built-in command for the determinant of a square matrix. It is just what you expect: `A.determinant()`.

```
A = matrix(QQ, 3,3, [4,3,5, 2,-1,0, -5,2,10])
A
```

```
[ 4  3  5]
[ 2 -1  0]
[-5  2 10]
```

To be sure that this works properly, we can do it the old way, too:

```
P, L, U = A.LU(pivot="nonzero")
print(P)
print(L)
print(U)
```

```
[1 0 0]
[0 1 0]
[0 0 1]
[     1      0      0]
[   1/2      1      0]
[  -5/4 -23/10      1]
[   4    3    5]
[   0 -5/2 -5/2]
[   0    0 21/2]
```

Clearly, there are no row swaps used, so it is easy to see that $\det(A) = 4 * (-5/2) * (21/2) = -105$.

Exercises

Task 5.1. (Strang Ex 5.1.1) If a 4×4 matrix A has $\det(A) = 1/2$, find

1. $\det(2A)$,
2. $\det(-A)$, and
3. $\det(A^2)$.

Explain which properties of determinants you need to make your deductions.

Task 5.2. (Strang Ex 5.1.2) If a 3×3 matrix has $\det(A) = -1$, find

1. $\det(\frac{1}{2}A)$,
2. $\det(-A)$,
3. $\det(A^2)$, and
4. $\det(A^{-1})$

Task 5.3. (Strang Ex 5.1.10)
Suppose that A is a matrix with the property that the entries in each of its rows add to zero. Solve the equation $Ax = 0$ to prove that $\det(A) = 0$.
Then suppose that instead the entries in each row add to one. Show that $\det(A - I) = 0$. Does this mean that $\det(A) = 1$? Give an example or an argument.

Task 5.4. (Strang Ex 5.1.12) The inverse of a 2×2 matrix seems to have determinant 1 all the time:

$$\det(A^{-1}) = \det \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{ad - bc}{ad - bc} = 1.$$

What is wrong with this argument? What should the value of $\det(A^{-1})$ be?

Task 5.5. (Strang Ex 5.1.14) Use row operations to reduce these matrices to upper triangular ones, then find the determinants:

$$\det \begin{pmatrix} 1 & 2 & 3 & 0 \\ 2 & 6 & 6 & 1 \\ -1 & 0 & 0 & 3 \\ 0 & 2 & 0 & 7 \end{pmatrix} \quad \text{and} \quad \det \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

Task 5.6. (Strang Ex 5.1.15) Use row operations to simplify and compute these determinants:

$$\det \begin{pmatrix} 101 & 201 & 301 \\ 102 & 202 & 302 \\ 103 & 203 & 303 \end{pmatrix} \quad \text{and} \quad \det \begin{pmatrix} 1 & t & t^2 \\ t & 1 & t \\ t^2 & t & 1 \end{pmatrix}.$$

Task 5.7. (Strang Ex 5.1.23)
Suppose that $A = \left(\begin{smallmatrix} 4 & 1 \\ 2 & 3 \end{smallmatrix}\right)$ and that λ is some number. Find A^2, A^{-1} and $A - \lambda I$ and their determinants.
Which two numbers λ lead to $\det(A - \lambda I) = 0$?

5.2 Computing Determinants

The Assignment

- Read Chapter 5 section 2 of Strang.
- Read the following and complete the exercises below.

Learning Goals

Before class, a student should be able to compute the determinant by using cofactors. A student should also be able to compute a determiant using the "big formula" for matrices of size 2 or 3.

Some time after class, a student should be comfortable with the different parts of the invertible matrix theorem.

Discussion: The Importance of the Determinant

Strang devotes all of his energy in this section to the different ways to compute the determinant. I don't have much to add to that.

The real importance of the determinant is described in the following theorem. Note that this is a special result for square matrices. The shape is crucial for this result.

Theorem 5.1. *(The Invertible Matrix Theorem)*
Let A be an $n \times n$ matrix. Then the following conditions are equivalent:

- *The columns of A are linearly independent.*
- *The columns of A are a spanning set for $\mathbb{R}^n$.*
- *The colums of A are a basis for $\mathbb{R}^n$.*
- *The rows of A are linearly independent.*
- *The rows of A are a spanning set for $\mathbb{R}^n$.*
- *The rows of A are a basis for $\mathbb{R}^n$.*
- *For any choice of vector $b \in \mathbb{R}^n$, the system of linear equations $Ax = b$ has a unique solution.*
- *A is invertible.*
- *The transpose A^T is invertible.*
- $\det(A) \neq 0$.
- $\det(A^T) \neq 0$.

Exercises

Task 5.8. (Strang 5.2.2) Compute determinants of the following matrices using the big formula. Are the columns of these matrices linearly independent?

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, C = \begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}, D = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}.$$

Task 5.9. (Strang 5.2.3) Show that $\det(A) = 0$, no matter what values are used to fill in the five unknowns marked with dots. What are the cofactors of row 1? What is the rank of A? What are the six terms in the big formula?

$$A = \begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & 0 & \bullet \\ 0 & 0 & \bullet \end{pmatrix}.$$

Task 5.10. (Strang 5.2.4) Use cofactors to compute the determinants below:

$$\det \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \qquad \det \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 3 & 4 & 5 \\ 5 & 4 & 0 & 3 \\ 2 & 0 & 0 & 1 \end{pmatrix}.$$

Task 5.11. (Strang 5.2.5) What is the smallest arrangement of zeros you can place in a 4×4 matrix to guarantee that its determinant is zero? Try to place as many non-zero entries as you can while keeping $\det A \neq 0$.

Task 5.12. Decide if the columns of this matrix are linearly dependent without doing any row operations:

$$A = \begin{pmatrix} 4 & 21 \\ 3 & 16 \end{pmatrix}.$$

Task 5.13. Complete this matrix to one with determinant zero in four genuinely different ways. How did you make that happen?

$$X = \begin{pmatrix} 2 & 1 & \bullet \\ 1 & 1 & \bullet \\ -1 & 1 & \bullet \end{pmatrix}.$$

5.3 Going Further: The Determinant

The Assignment

- Go back through the exercises in this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in the last two sections of Chapter Five. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

Chapter 6

Eigendata and the Singular Value decomposition

We have studied the algebraic structure of matrices, and we have studied the way that these algebraic properties are reflected in a geometric model where a matrix represents a transformation from one Euclidean space to another.

In this chapter we shall study how to do this in a different way that cares more about the geometry adapted to the specific matrix. First we shall take up square matrices, where we will study **eigenvalues** and **eigenvectors**. We shall see that some matrices can be reimagined as if they are diagonal matrices. This idea of **diagonalization** is powerful, but it doesn't always work. We will study one situation where we know it will always work in the **spectral theorem** for symmetric matrices.

Then we shall apply all that we have learned this term and put it together to study the **singular value decomposition** of a general rectangular matrix. This will be a good geometric understanding of how a matrix behaves as a function.

6.1 Eigenvectors and Eigenvalues

The assignment

- Read section 6.1 of Strang (pages 283-292).
- Read the following and complete the exercises below.

Discussion: Eigenvalues and Eigenvectors

We have discussed the "transformational view" of the geometry of a system of m linear equations $Ax = b$ in n unknowns x, where we view the $m \times n$ matrix A as defining a function from $\mathbb{R}^m$ to $\mathbb{R}^n$. In the case of a square matrix $m = n$, the domain and the target are the same space $\mathbb{R}^n$. So we can think of A as making a function from one space to itself.

This means it might be interesting to think about how A moves a vector about inside of $\mathbb{R}^n$. Usually, the vector v will get turned into a vector Av which has a different length and points in a completely different direction. But sometimes, sometimes, v and Av will point in the same direction. This is an eigenvector.

A number λ is called an **eigenvalue** of the matrix A when the matrix $A - \lambda I$ is singular. A vector v is called an **eigenvector** of A corresponding to λ when v is not zero but still lies in the null space of $A - \lambda I$. We exclude 0 from being an eigenvector because it is boring. The zero vector lies in every subspace, including the nullspace of any matrix.

As Strang discusses, the eigenvalues are found as roots of the **characteristic polynomial** $\det(A - \lambda \cdot I) = 0$. That's right, we only need to find the roots of a polynomial! Sounds great, but as a general thing this is pretty hard. Don't get too excited. Have you heard this fact before? It is both depressing and interesting: there is no general formula to find the roots of a polynomial of degree 5 or more.

Sage and Eigenvectors

Since eigenvalues and eigenvectors are found using standard techniques, we can use Sage to compute them without any new techniques.

Using Nullspaces and root finding commands

Let's use basic sage commands we have seen before to compute eigenvalues and eigenvectors. We start by finding the characteristic polynomial of the mundane example matrix X below.

```
X = matrix(QQ, 3,3, [1,2,3,4,5,6,7,8,9])
X
```

```
[1 2 3]
[4 5 6]
[7 8 9]
```

```
t = var('t')
poly = (X - t*identity_matrix(3)).determinant()
poly
```

```
-((t - 5)*(t - 9) - 48)*(t - 1) + 29*t + 3
```

Now we need the roots of that polynomial. Sage has a simple built-in for that, which returns a list of pairs: (root, multiplicity).

```
poly.roots()
```

```
[(-3/2*sqrt(33) + 15/2, 1), (3/2*sqrt(33) + 15/2, 1), (0, 1)]
```

In this case each of the three roots has (algebraic) multiplicity equal to one. For now, we will look at just the first one. Let's pull it out of this list, give it a more convenient name, and use it to find a corresponding eigenvector.

```
lam = poly.roots()[0][0]
V = (X - lam * identity_matrix(3)).right_kernel()
V
```

So, that looks like a mess. We can get Sage to display the basis vector more nicely. This is our eigenvector.

```
show(V.basis()[0])
```

Well, that probably needs a simplification or two. But there it is!

Built-in Sage Commands

Sage has useful built-in commands that get at the same computations. But for them to work, your matrix must be defined over a set of numbers that is big enough to take roots of polynomials. We will use `AA`, which stands for the real algebraic numbers.

```
A = matrix(AA, 4,4, [3,134,-123,4, 2,1,34,4, 2,36,54,7, 0,0,3,1])
A
```

```
[   3  134 -123    4]
[   2    1   34    4]
[   2   36   54    7]
[   0    0    3    1]
```

```
A.characteristic_polynomial()
```

```
x^4 - 59*x^3 - 990*x^2 + 18135*x - 14675
```

```
A.eigenvalues()
```

The eigenvectors can be computed with this command, which again returns pairs: (eigenvalue, eigenvector).

```
A.eigenvectors_right()
```

Or you can ask for the **eigenspaces** which return pairs: (eigenvalue, subspace consisting of all eigenvectors which correspond).

```
A.eigenspaces_right()
```

One more example...

```
B = matrix(AA, 2,2, [5,1,0,5])
B
```

```
[5 1]
[0 5]
```

This matrix has only one eigenvalue, but it has algebraic multiplicity 2.

```
B.eigenvalues()
```

```
[5, 5]
```

But 5 has **geometric multiplicity** only equal to one, because the corresponding eigenspace has dimension one.

```
B.eigenvectors_right()
```

```
B.eigenspaces_right()
```

Questions for Section 6.1

Task 6.1. Exercise 5 from section 6.1 of Strang.

Task 6.2. Exercise 6 from section 6.1 of Strang.

Task 6.3. Exercise 7 from section 6.1 of Strang.

Task 6.4. Exercise 12 from section 6.1 of Strang.

Task 6.5. Exercise 14 from section 6.1 of Strang.

Task 6.6. Exercise 19 from section 6.1 of Strang.

6.2 Diagonalizing Matrices

The assignment

- Read section 6.2 of Strang (pages 298-307).
- Read the following.
- Prepare the items below for presentation.

Diagonalizing Matrices

The big result here is this:

Theorem 6.1. *Let A be an $n \times n$ square matrix. Then the following two conditions are equivalent:*

- *There is a basis $\beta = \{v_1, v_2, \ldots, v_n\}$ for $\mathbb{R}^n$ consisting of eigenvectors for A.*
- *It is possible to find an invertible matrix S so that $A = S\Lambda S^{-1}$, where Λ is a diagonal matrix whose entries are the eigenvalues of A.*

The connection between the two conditions is that the matrix S has as its columns the eigenvectors of A. (In fact, that is really the heart of the proof of this theorem. The rest is just details.)

If a matrix satisfies these two conditions, then we say it is **diagonalizable**. We should note right away that not all matrices are diagonalizable. We have already seen examples of matrices where the geometric multiplicity of an eigenvalue is less than the algebraic multiplicity, like $A = \left(\begin{smallmatrix} 5 & 1 \\ 0 & 5 \end{smallmatrix}\right)$. In this case, it becomes impossible to find a basis consisting of eigenvectors.

In a way, this allows us to see something interesting: maybe a matrix really wants to be a diagonal matrix, but we are looking at the transformation A using "the wrong basis." By wrong, here I mean that the standard basis is not the most convenient one, and another one makes our lives easier.

Sage and Diagonalization

Sage has built-in commands about diagonalization. We shall try a few out here. We need a matrix to play with, so we take this one:

```
A = matrix(AA, 3,3, [1,2,3,4,5,6,7,8,-1])
A.rank()
```

```
3
```

We chose to define this matrix over AA because we need to find roots of polynomials when looking for eigenvalues. AA is the set of **algebraic numbers**, which just means the collection of all roots of polynomials with integer coefficients.

```
A.is_diagonalizable()
```

```
True
```

Sage has a command for finding the eigenvector decomposition $A = S\Lambda S^{-1}$.

```
A.eigenmatrix_right()
```

```
([ 11.816056999423874?                    0                    0]
[                   0 -0.3954315737468559?                    0]
[                   0                    0  -6.420625425677017?], [
    1.000000000000000?   1.000000000000000?   1.000000000000000?]
[  2.369820536283515?  -0.866496699124881?   1.460961877127081?]
[  2.025471975618948? 0.11252060816763521?  -3.447516393310393?])
```

As you see, Sage returns a pair of matrices. One of them is diagonal, so that is probably Λ. We'll use tuple unpacking to assign the matrices to sensible names.

```
Lambda, S = A.eigenmatrix_right()
```

```
Lambda
```

```
[ 11.816056999423874?                    0                    0]
[                   0 -0.3954315737468559?                    0]
[                   0                    0  -6.420625425677017?]
```

```
S
```

```
[  1.000000000000000?   1.000000000000000?   1.000000000000000?]
[  2.369820536283515?  -0.866496699124881?   1.460961877127081?]
[  2.025471975618948? 0.11252060816763521?  -3.447516393310393?]
```

Note that S has the eigenvectors of A as its columns, and the corresponding eigenvalues are lined up as the diagonal entries of Λ.

```
A.eigenvectors_right()
```

```
[(11.816056999423874?, [
(1.000000000000000?, 2.369820536283515?, 2.025471975618948?)
], 1), (-0.3954315737468559?, [
(1.000000000000000?, -0.866496699124881?, 0.11252060816763521?)
], 1), (-6.420625425677017?, [
(1.000000000000000?, 1.460961877127081?, -3.447516393310393?)
], 1)]
```

Anyway, now we can check that everything lines up correctly:

```
S * Lambda * S.inverse()
```

```
[ 1.000000000000000?  2.000000000000000?  3.000000000000000?]
[ 4.000000000000000?  5.000000000000000?  6.000000000000000?]
[ 7.000000000000000?   8.00000000000000? -1.000000000000000?]
```

```
S * Lambda * S.inverse() == A
```

```
True
```

Questions for Section 6.2

Task 6.7. Let e_1, e_2, e_3 be the standard basis of $\mathbb{R}^3$:

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Make an example of an invertible 3×3 matrix S. Write your matrix as a matrix of column vectors.

$$S = \begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix}$$

How do you know that the set $\{Se_1, Se_2, Se_3\}$ is a basis for $\mathbb{R}^3$?

What is the connection between Se_1, Se_2, Se_3, $S^{-1}v_1$, $S^{-1}v_2$, $S^{-1}v_3$ and the original vectors $e_1, e_2, e_3, v_1, v_2, v_3$?

Finally, how do we use this to understand the way that the decomposition $A = S\Lambda S^{-1}$ works?

Task 6.8. Exercise 1 from section 6.2 of Strang.

Task 6.9. Exercise 2 from section 6.2 of Strang.

Task 6.10. Exercise 3 from section 6.2 of Strang.

Task 6.11. Exercise 13 from section 6.2 of Strang.

Task 6.12. Exercise 19 from section 6.2 of Strang.

6.3 The Spectral Theorem

The Assignment

- Read chapter 6 section 4 of *Strang*
- Read the following and complete the exercises below.

Discussion: The Spectral Theorem for Symmetric Matrices

We are now in a position to discuss a major result about the structure of symmetric (square) matrices: The Spectral Theorem.

Theorem 6.2. *Suppose that A is a symmetric $n \times n$ matrix. Then there exists an orthonormal basis $\{q_1, q_2, \dots, q_n\}$ of $\mathbb{R}^n$ consisting of eigenvectors of A. This means that we can factor A as a product*

$$A = Q\Lambda Q^{-1} = Q\Lambda Q^T,$$

where Q is an orthogonal matrix having the vectors q_i as its columns, and Λ is a diagonal matrix with the eigenvalues of A as its entries.

I think Strang's argument for the truth of this theorem is too terse, and hence confusing for a first time reader. The main points of the argument are these:

1. If X is a symmetric matrix, then all of its eigenvalues are real numbers.
2. If X is a symmetric matrix and v and w are eigenvectors of X which correspond to *different* eigenvalues, then v and w are orthogonal vectors.
3. If X is a symmetric matrix and λ is an eigenvalue of X, then subspace of λ-eigenvectors for X has dimension equal to the multiplicity of λ as a root of the characteristic polynomial of X. (This point is often stated by saying that the **geometric multiplicity** of λ is equal to the **algebraic multiplicity** of λ.)

Let's clear up that bit about the different types of multiplicity. We can identify eigenvalues by finding them as roots of the characteristic polynomial $p_A(t) = \det(A - t \cdot I)$ of A. Of course, an particular root can be a root *multiple times.* For example, 5 is a root of the polynomial $t^2 - 5t + 25 = 0$ twice. So we say 5 has multiplicity two. In the context of eigenvalues, this multiplicity is called the **algebraic multiplicity** of an eigenvalue, since it comes out of the consideration of the algebra.

Another way to count up the number of times a number counts as an eigenvalue is to use the number of eigenvectors corresponding to that number. But we only want to count up truly independent directions, so we should use the dimension of the subspace of eigenvectors. This is the **geometric multiplicity** of an eigenvalue. It is a fact that the geometric multiplicity is not greater than the algebraic multiplicity. But the two can be different. For example, consider this matrix:

$$G = \begin{pmatrix} 5 & 1 \\ 0 & 5 \end{pmatrix}.$$

Now, how does one understand the Spectral Theorem? It basically guarantees that we can always find (a) enough eigenvalues (as real numbers), and (b) for each eigenvalue, enough eigenvectors. The hardest parts of the proof come from part (b) where you have to produce enough eigenvectors. But in practice, if you have an example of a symmetric matrix, you can find the decomposition mentioned in the theorem pretty easily. First, find the eigenvalues. Then for each eigenvalue λ, find an orthonormal basis for the **eigenspace**

$$E_\lambda = \text{null}(A - \lambda \cdot I).$$

That second bit can be done in two steps, first find a basis for E_λ (special solutions!) and then apply the Gram-Schmidt algorithm to find an orthonormal basis for E_λ. Collecting all of these bases together will make a basis for $\mathbb{R}^n$.

Sage and the Spectral Theorem

Sage does not have any built-in commands that deal with the spectral decomposition of a symmetric square matrix. But here are a few commands that you might find useful as you hack your solution together by hand:

The first command you might find useful is `.change_ring()`. This is helpful for those times when you define a matrix over some convenient ring like `QQ`, but then want to work with eigevalues and eigenvectors and so need a bigger ring that you can take roots in. Using this command doesn't change the matrix, so much as tell Sage to think of it as having entries from a different set of numbers.

```
A = matrix(QQ, 2,2, [1,2,3,4])
A.change_ring(AA)
```

```
[1, 2]
[3, 4]
```

The command `.jordan_form(transformation=True)` will return a pair consisting of a diagonal matrix with the eigenvalues as entries and an invertible matrix consisting of a basis of eigenvectors. These eigenvectors will NOT be an orthonormal basis. You will have to use Gram-Schmidt to fix this to a proper basis promised by the theorem.

Note: the **Jordan Form** is a generalization of the diagonalization process that works for matrices which might not be symmetric. We'll use it here to short-cut some of the work.

Let's do an example. First, we will make a symmetric matrix. Then we will find the eigenvalues and eigenvectors

```
A = matrix(QQ, 3,3, [1,2,4, 5,3,2, -1,3,3])
X = A.transpose()*A
Y = X.change_ring(AA); Y
```

```
[27 14 11]
[14 22 23]
[11 23 29]
```

```
D, S = Y.jordan_form(transformation=True)
D
```

```
[1.606673922554331?|                 0|                 0]
[------------------+------------------+------------------]
[                 0|17.88104756294764?|                 0]
[------------------+------------------+------------------]
[                 0|                 0|58.51227851449803?]
```

```
S
```

```
[  1.000000000000000?   1.000000000000000?   1.000000000000000?]
[ -4.402902344870359? -0.2014369844407669?   1.214399978615443?]
[  3.295209704612669? -0.5726213322619663?   1.319152619443804?]
```

In this case, we have three different 1-dimensional eigenspaces, so things are not too hard! If we apply Gram-Schmidt, we will just normalize those vectors.

```
Q, R = S.QR()
Q * D * Q.transpose()
```

```
[27.00000000000000? 14.00000000000000? 11.00000000000000?]
[14.00000000000000? 22.00000000000000? 23.00000000000000?]
[11.00000000000000? 23.00000000000000? 29.00000000000000?]
```

That is about as close as we can get to displaying the original `X`.

Exercises

Task 6.13. From Strang section 6.4, do exercise 3.

Task 6.14. From Strang section 6.4, do exercise 4.

Task 6.15. From Strang section 6.4, do exercise 5.

Task 6.16. From Strang section 6.4, do exercise 6.

Task 6.17. From Strang section 6.4, do exercise 8.

Task 6.18. From Strang section 6.4, do exercise 11.

Task 6.19. From Strang section 6.4, do exercise 12.

Task 6.20. From Strang section 6.4, do exercise 24.

6.4 Going Further with Eigendata

The Assignment

- Go back through the exercises in the first three sections of this chapter. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in the first three sections of Chapter Six. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?

- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.

- If your first solution to an exercise involve a “guess-and-check” approach, can you now complete the exercise in a *purposeful* and systematic manner?

- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.

6.5 Singular Value Decomposition

The Assignment

- Read chapter 6 section 7 of *Strang*
- Read the following and complete the exercises below.

Discussion: The Singular Value Decomposition

The **Singular Value Decomposition** is an adaptation of the ideas behind eigenvectors and eigenvalues for non-square matrices. If our matrix A is $n \times m$, the idea is to choose

- An orthonormal basis $\{v_1, \dots, v_n\}$ for $\mathbb{R}^n$,
- and an orthonormal basis $\{u_1, \dots, u_n\}$ for $\mathbb{R}^m$

so that

$$Av_i = \sigma_i u_i, \qquad \text{for } 1 \leq i \leq r$$

where $r = rank(A)$. If we pile up all of those equations, we get a statement like this one:

$$A \begin{pmatrix} | & | & \dots & | \\ v_1 & v_2 & \dots & v_r \\ | & | & \dots & | \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r \end{pmatrix} \begin{pmatrix} | & | & \dots & | \\ u_1 & u_2 & \dots & u_r \\ | & | & \dots & | \end{pmatrix}$$

Eventually, this will lead us to a matrix decomposition of the form

$$A = U\Sigma V^T.$$

How to do it

So, suppose that A is an $m \times n$ matrix. The key fact we need is that A^TA is a symmetric $n \times n$ matrix. Later, when discussing properties, it is important that AA^T is a symmetric $m \times m$ matrix, and also that $A(A^TA) = (AA^T)A$.

1. Step One: Compute a spectral decomposition of A^TA. Since A^TA is a square symmetric matrix, we can find an orthonormal basis $v_1, v_2, \dots, v_n$ for $\mathbb{R}^n$ which consists of eigenvectors for A^TA. If we bundle these together as the columns of a matrix V, we can make the spectral decomposition

   $$A^TA = VDV^T,$$

   where D is a diagonal matrix having the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ for entries. NOTE: We will organize things so that the eigenvalues get smaller as we go further in the list.

   Now, some of the eigenvaules might be zero, but none will be negative. We will have exactly $r = \text{rank}(A)$ non-zero eigenvalues, and the other $n - r$ will be equal to zero. (Those come from the directions in the null space!) Don't sweat it. Everything is going to be fine.

2. Step Two: For each $0 \leq i \leq r$, set

   $$\sigma_i = \sqrt{\lambda_i}.$$

   These are the "Singular Values" named in the singular value decomposition.

3. Step Three: for each $0 \leq i \leq r$, set
$$u_i = \frac{1}{\sigma_i} A v_i.$$
By the way things are set up, we are sure σ_i is not zero! The vectors we just made are a basis for the column space of A.

4. Step Four: If $m > r$, Choose an orthonormal basis $u_{r+1}, \ldots, u_m$ for the null space of A^T.

5. Step Five: Bundle together the v_i's as columns of an $n \times n$ matrix V, and the u_i's as the columns of an $m \times m$ matrix U. Both of these are orthogonal matrices. Then place the σ_i's on the main diagonal of a rectangular $m \times n$ matrix which is otherwise filled with zeros. Call that new matrix Σ.

Strang has a good run-down of the neat properties this set up has.

Sage and the SVD

Sage has a built in command for the singular value decomposition of a matrix. If you have a matrix `A`, the command is `A.SVD()`. But there is a little trick to using it! At present, Sage only has this function implemented for matrices defined over rings of "floating point numbers". The best way around this is to either define your matrix with entries in the ring `RDF`, or use the `.change_ring(RDF)` method on you matrix before you use the SVD.

```
A = matrix(QQ, 2,2, [2,1,1,1])
A
```

```
[2, 1]
[1, 1]
```

```
A.change_ring(RDF).SVD()
```

```
(
[-0.850650808352 -0.525731112119]  [2.61803398875            0.0]
[-0.525731112119  0.850650808352], [          0.0
    0.38196601125],

[-0.850650808352 -0.525731112119]
[-0.525731112119  0.850650808352]
)
```

Exercises

Task 6.21. Complete exercise 6.7.1 from *Strang*.

Task 6.22. Complete exercise 6.7.2 from *Strang*.

Task 6.23. Complete exercise 6.7.3 from *Strang*.

Task 6.24. Complete exercise 6.7.4 from *Strang*.

Task 6.25. Complete exercise 6.7.5 from *Strang*.

Task 6.26. Complete exercise 6.7.6 from *Strang*.

Task 6.27. Complete exercise 6.7.9 from *Strang*.

Task 6.28. Complete exercise 6.7.10 from *Strang*.

Task 6.29. Complete exercise 6.7.11 from *Strang*.

Task 6.30. Complete exercise 6.7.12 from *Strang*.

6.6 Going Further with the SVD

The Assignment

- Go back through the exercises in the last section. Complete any items you did not complete the first time through. Prepare any that we have not discussed in class so that you will be ready to present them.

Discussion

Now we take a short break to revisit and consolidate the learning you have done so far. Revisit the reading and the exercises you have done in Chapter Six. The important feature of this work should be learning to think about your own thinking. This sort of **meta-cognition** characterizes expert learners. Eventually, you want to be able to monitor your work at all times and recognize when you understand deeply and when you do not. This will allow you to self-correct.

To help you get started with meta-cognition, I listed learning goals in each section. To go further, you need to explicitly go through the process of reviewing what you can do and what you cannot. Here are some prompts to help you get started with this process.

- Review the learning goals from each section. Can you do the things described? Can you do them sometimes, or have you mastered them so you can do them consistently?
- Look through all of the tasks and go deeper into them. Can you connect each exercise to one of our pictures? Try to build a mental model of how the exercise and its solution work.
- If your first solution to an exercise involve a "guess-and-check" approach, can you now complete the exercise in a *purposeful* and systematic manner?
- Make a list of concepts or exercises that are not clear to you. Phrase each item in your list as a question, and make each question as specific as possible. Talk with fellow students or your instructor until you can answer your own questions.