please define linear algebra in context - how

Welcome to LinearAlgebraforStatics!

Python is ca

This is a basic notebook to learn how to use linear algebra in the real world and in statics, and python to solve statics problems. After, you can use the notebook to solve any linear system.

First, lets look at some examples of how we can use matrices in the real world and in statics:

In the real world, especially in engineering, we will encounter circuit diagrams. From *Advanced Engineering Mathematics* (Kreyszig), we have one with 3 unknown currents. Matrices come in handy here because once we write the appropriate Kirchoff's Laws equations for the unknowns, then we can solve them via matrices.

please include image

Your audience is a Workshop student. Workshop students haven't ye

In statics, we can use them when we are solving structures using the joints method. From *Engineering Mechanics: Statics* (Meriam et. al.), here is a more life like problem on a signboard where we are asked to find the force in members BE and BC when the horizontal wind load is 712 lb. Specifically that 5/8 of the force is transmitted to the center connection at C and the rest is equally divided between D and B. In this example to use matrices, we would go through joints D, C and E and write down each force equation with the appropriate unknowns.

It is not clear to me why you're using specific numbers, such as

Statics Example

Next, lets learn about matrices and how they can represent linear systems.

Linear Systems with Matrices

To learn about matrices with linear systems, lets look at the circuit diagram example. First, we will learn about matrices in general.

these two sentences contra

Linear Systems must be in this form to solve:
$ x - y + z = 0 $
$ -x + y - z = 0 $
$ 10y + 25z = 90 $
$ 20x + 10y = 80 $

avoid personification and unspecifi

Matrices are collection of numbers that are arranged in arrays, like tables. They are helpful, besides making systems easier to solve, because they can take large amounts of information and represent it in a concise way.

does your audience know what it me

Matrices look like numbers in a rectangle format enclosed by brackets. Lets look at a couple of examples:

$$\begin{bmatrix} 1 & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & 10 & 25 & 90 \\ 20 & 10 & 0 & 80 \end{bmatrix}$$

$$\begi

Is there are reason to have these grouped? Could you divide them up into matrix,

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 0 & 10 & 25 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 90 \\ 80 \end{bmatrix}$$

plural / singular di

The first matrix is called an **augmented matrices**. Augment

this is the first time you've indicated that matric

linear systems. Augmented Matrices contain the coefficients of the variables for each equation in relation to its variable name and what the equations are equal to. In this program, we will use augmented, rectangular and square matrices. Notice that the format are equations that are solely variable and constants with addition and subtraction operators in front of them. The augmented matrix of the above linear system is:

The second matrix is a **rectangular matrix** and a **coefficient matrix** of size 4 by 3 (rows by columns). It is called rectangular because it is in the form of a rectangle and it is called a coefficient matrix because it only represents the coefficients of the linear system. The third matrix has size 3x3 and therefore is in a form of a square, so we call it a **square matrix**. The fourth matrix is a special kind of matrix called a **vector matrix** because it has a singular row (can also have a singular column).

what do you want the students to lear

Here is an example of how matrices will look printed from the library we will use in Python (more on that later):

In [2]:
```
import numpy as np

print(np.array([[1, -1, 1, 0], [-1, 1, -1, 0], [0, 10, 25, 90], [20, 1
0, 0, 80]]))
```

```
[[ 1 -1  1  0]
 [-1  1 -1  0]
 [ 0 10 25 90]
 [20 10  0 80]]
```

Next, lets write Kirchoff's laws equations to the problem. When you write out the laws, you could have something like this:

$i_1 - i_2 + i_3 = 0$
$-i_1 + i_2 - i_3 = 0$
$10i_2 + 25i_3 = 90V$
$20i_1 + 10i_2 = 80V$

You will notice it it the same as the example given above with i_1 as x, i_2 as y, i_3 as z. This is the system we will continue to solve.

Now, we are ready to type our matrix into the code.

The Code

could you make this more interactive? Assuming this is the first time s

In the cell below is all we need to solve linear systems. Let's go through the code line by line.

`import numpy as np`: This is called an import statement. Python comes with a certain range of knowledge which we can extend by importing libraries. numpy is a well known library to do scientific computing. We import it `as np` basically to give it an alias, so in code we do not have to keep typing out numpy in its entirety.

`cofmat = np.array([])`
`vecmat = np.array([])`: These statements are called assignment statements. In python, we also initialize variables in assignment statements. Here, we are setting each of our variables `cofmat` and `vecmat` to numpy "styled" arrays. `np.array([])` will `return` an array that will be written by the used. The basic format for these dot operations is the library the method comes from and then the method you would like to access (`library.method()`). You will see an example later in the code where the method is buried deeper in multiple library and a routine.

`print(cofmat)`
`print(vecmat)`: These are called print statements. After you enter your arrays, these statements so you can see your arrays in the format you saw above. These are added for ease.

`sol = np.linalg.lstsq(cofmat, vecmat)`
`print(sol[0])`
`print(Done)`: Again, we have another assignment statement. You will see we will access the numpy library again and use a method from the routine linalg called lstsq. This stands for the least squares method and an example of a method buried in a routine from a library. The general syntax looks like this `library.routine.method()`. This method returns a list (in a form of a `type list` in python) of useful data. For our purposes, the answer to the system is put in the first entry of the list. We print and access it on the next line. In lists and arrays we count each object starting at 0, which is why we have `sol[0]` written. Finally, we print `Done` to indicate to the user that the program is done.

To use this program, we must type our system in as 2 matrices, a coefficient matrix and a matrix for the constants (a vector matrix). All we have to do is enter each row of the respective matrix as its own list into the array. So, for our example (printed below for convenience) we will type in this:

$ x - y + z = 0 $

$ -x + y - z = 0 $

$ 10y + 25z = 90 $

$ 20x + 10y = 80 $

```
cofmat = np.array([[1, -1, 1], [-1, 1, -1], [0, 10, 25], [20, 10, 0]])
vecmat = np.array([0, 0 , 90, 80])
```

The answer to this matrix is
[2. 4. 2.]

In [1]:
```
import numpy as np

cofmat = np.array([])
vecmat = np.array([])

print(cofmat)
print(vecmat)

sol = np.linalg.lstsq(cofmat, vecmat)
print(sol[0])
print("Done")
```

not sure why there is an error. I didn't run the not

```
[]
[]

---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-1-d54812139cac> in <module>()
      7 print(vecmat)
      8 
----> 9 sol = np.linalg.lstsq(cofmat, vecmat)
     10 print(sol[0])
     11 print("Done")

/usr/lib/python3/dist-packages/numpy/linalg/linalg.py in lstsq(a, b, rcond)
   1901     if is_1d:
   1902         b = b[:, newaxis]
-> 1903     _assertRank2(a, b)
   1904     _assertNoEmpty2d(a, b)  # TODO: relax this constraint
   1905     m  = a.shape[0]

/usr/lib/python3/dist-packages/numpy/linalg/linalg.py in _assertRank2(*arrays)
    194         if a.ndim != 2:
    195             raise LinAlgError('%d-dimensional array given. Array must be '
--> 196                     'two-dimensional' % a.ndim)
    197 
    198 def _assertRankAtLeast2(*arrays):

LinAlgError: 1-dimensional array given. Array must be two-dimensional
```

Now, lets try our statics example in the above cell.

Here is the problem again: Here is a signboard where we would like to find the force in members BE and BC when the horizontal wind load is 712 lb. Specifically that 5/8 of the force is transmitted to the center connection at C and the rest is equally divided between D and B.

Statics Example

I don't know what you have in the image, but please paste in the entire r

It is unclear what you expect the students to do. Please put them in a system. First, solve for the D_x and C_x. Ther how does one "s you can solve very simply (they are the only force in a certain direction) and I put in their values explicitly (CE = 445 T).

again, avoid personification and

From joint D, you should get the equations:
$ 133.5 = .4961DE $
$ 0 = .8682DE - CD $

please make thes

From joint C, you should get the equation:
$ 0 = CD - BC $

* From this joint you also will find CE = 445 T

From joint E, you should get the equations: $ 0 = .8682DE + .8682DE - .8682EF $
$ 445 = -.4961DE + .4961BE + .4961EF $

Remember when you enter your equations into a matrix to order your variables and place 0's where necessary. Making $ a = DE, b = CD, c = BC, d = BE $ and $ e = EF $ (letters chosen for order), here are matrices for cofmat and vecmat:

does adding additional variables help the student think about the probl

Cofmat:

```
[[.4961, 0, 0, 0, 0]
 [.8682, -1, 0, 0 ,0]
 [0, 1, -1, 0, 0]
 [.8682, 0, 0, .8682, -.8682]
 [-.4961, 0, 0, .4961, .4961]]
```

Vecmat:

```
[133.5, 0, 0 , 0, 445]
```

If you have all your forces in the correct direction, here is the answer:
[269.09897198 233.63172747 233.63172747 448.49828664 717.59725862] with D_x = 133.5lb, C_x = 445lb $ CE = 445lb $

or $ DE = 269.0989lb, CD = 233.6317lb, BC = 233.6317lb, BE = 448.4983lb, EF = 717.5972lb, D_x = 133.5lb, C_x = 445lb and CE = 445lb $
* Remember, a negative answer indicated that you put the member in its opposite state (tension vs. compression).

Try it for yourself!

Now, you are free to use this program with any problem!

References:

KREYSZIG ERWIN et. al. “Linear Algebra: Matrices, Vectors, Determinants. Linear Systems.” ADVANCED ENGINEERING MATHEMATICS, 10, John Wiley & Sons, Inc, 2011, 256-282. School of Aeronautics (Neemrana), https://soaneemrana.org/onewebmedia/ADVANCED%20ENGINEERING%20MATHEMATICS%20BY%20ERWIN%20ERESZIG1.pdf

please remove