PH3501 - 1 Math Refresher and Python Introduction

Claus-Dieter Ohl

October 13, 2016

1 Math Refresher

In this course we work with vectors, matrices, and tensors. Vectors can be written as row and column vectors, this is important as it defines how they multiply with matrices and tensors. A column vector is written as a vertical tuple, while the row vector is a horizontal tuple:

$$\vec{a} = \mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = (a_1, a_2, a_3)^T$$

The ^{T} sign transposes the vector or matrix/tensor, rows become columns and vice versa. Please note the different ways we can write the vectors; in this course we'll mostly use the bold face style, i.e. $\mathbf{a}$.

1.1 Scalar Product

The *scalar product* or *dot product* is a product between two vectors where the same components are multiplied and then summed. The result is related to length and $\cos$ of the angle of the two vectors.

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}|\,|\mathbf{b}|\,\cos(\mathbf{a},\mathbf{b}) = a_1 b_1 + a_2 b_2 + a_3 b_3 \quad (1)$$

$$= \sum a_i b_i := a_i b_i \quad (2)$$

The last expression is a definition, the so-called *Einstein summation convention*. In this course we will only use it for identical subscripts in products. Then it states, that we automatically sum over identical subscripts, the summation sign is not written anymore. The reason is that we can conveniently write lengthy products with less mathematical symbols.

Having calculated the scalar product of two vectors, we can easily obtain the angle between the vectors through:

$$\cos(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}|\,|\mathbf{b}|} \quad (3)$$

Homework:

Write above equation using Einstein summation convention.

Below you find how to define column and row vectors in Python. We will always use two packages which work hand in hand together. The first package is called `numpy` (short for NUMerical PYthon) which provides functions for vectors and matrices including all kinds of linear algebra tools. The second package is scipy (SCIentific PYthon) which has all the mathematical functions numerical and work conveniently with the vectors and matrices defined with numpy.

```python
import numpy as np      #Numpy contains the versatile array classfrom scipy
import math #Needed for the transzendental functions math.acos() and the constant math.pi

print "This is a row vector"
a = np.array([2,2,2])
print "a={0}".format(a)
```

```
This is a row vector
a=[2 2 2]
```

```python
print "This is a column vector"
b = np.array([[1],[2],[3]])
print "b={0}".format(b)
```

```
This is a column vector
b=[[1]
 [2]
 [3]]
```

Row and column vectors in Matlab are very different while Python does not really care about it.

```python
print "Dot product of two row vectors"
print "a . a = {0}".format(a.dot(a))
print "which is the same as"
print "a . a = {0}".format(np.dot(a,a))
print "which is the same for two column vectors"
print "b . b = {0}".format(np.vdot(b,b))  #For column vectors we need to use vdot()
```

```
Dot product of two row vectors
a . a = 12
which is the same as
a . a = 12
which is the same for two column vectors
b . b = 14
```

```python
print "Now we define two vectors and show that their angle is 90 degrees"
c = np.array([2,0])
d = np.array([0,2])
print c
print d
```

```
Now we define two vectors and show that their angle is 90 degrees
[2 0]
[0 2]
```

```python
myangle = math.acos(np.vdot(c,d)/(np.linalg.norm(c)/np.linalg.norm(d)))
print "Angle between (c,d) = {0} degree".format(myangle*180./math.pi)
```

```
Angle between (c,d) = 90.0 degree
```

Thus the important functions to remember are the initialization using np.array([a_1 , a_2 , a_3]), the scalar product np.dot(a,b), and the norm using np.linalg.norm(a).

1.2 Vector Product

The vector product (or sometimes called cross product) leads to a vector **c** which is perpendicular to **a** and **b**:

$$\mathbf{a} \times \mathbf{b} = \mathbf{c} \quad .$$

It's length is given by

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}|\,|\mathbf{b}|\,\sin(\mathbf{a}, \mathbf{b})\,.$$

A coordinate system where **a** and **b** are not parallel

$$(\mathbf{a}, \mathbf{b}, \mathbf{a} \times \mathbf{b})$$

is right handed.

A few rules to keep in mind:

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) \neq (\mathbf{a} \times \mathbf{b}) \times \mathbf{c}$$

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$$

$$\mathbf{a} \times \mathbf{a} = \mathbf{0}$$

$$(\mathbf{a} + \mathbf{b}) \times \mathbf{c} = \mathbf{a} \times \mathbf{c} + \mathbf{b} \times \mathbf{c}$$

If $\mathbf{a} \times \mathbf{b} = \mathbf{0}$ then $\mathbf{a} = \mathbf{0}$, $\mathbf{b} = \mathbf{0}$, and/or $\mathbf{a} \parallel \mathbf{b}$.

But how do we calculate the vector product?

$$(a_1, a_2, a_3) \times (b_1, b_2, b_3) = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}^T$$

I have difficulty remembering the order of the variables, an eays way out is to remember that the vector product is the determinant of a matrix with 3 rows composed out of the unit vectors $(\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3})$, the first, and the second vector:

$$\begin{vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

.

The cross product is calculated with the function `np.cross(a,b)` where a and b are two row or column vectors.

```
a = np.array([1,0,0])
b = np.array([0,1,0])
print "We calculate the cross product"
print "a = {0}".format(a)+", b = {0}".format(b)
print "a x b = {0}".format(np.cross(a,b))
```

```
We calculate the cross product
a = [1 0 0], b = [0 1 0]
a x b = [0 0 1]
```

1.3 Matrix and Matrix or Inner Product

A matrix has row and columns and each position of the matrix is addressed with a tuple (i, j), where i refers to the rows and j to the columns. This is an example of a 3x3 matrix:

$$\underline{\underline{\mathbf{A}}} = A_{ij} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

The product of two matrices at position (i, j) in the resulting matrix is the summation of the products of the i-th row of the first matrix and the j-th column of the second matrix. As an example let two 3x3 matrices $\underline{\underline{\mathbf{A}}}$ and $\underline{\underline{\mathbf{B}}}$ multiply and write down the component (2,1) of the resulting matrix:

$$\underline{\underline{\mathbf{A}}}\,\underline{\underline{\mathbf{B}}} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} = \begin{pmatrix} ... & ... & ... \\ A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31} & ... & ... \\ ... & ... & ... \end{pmatrix}$$

The identity matrix is

$$\underline{\underline{\mathbf{I}}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \delta_{ij}$$

The transpose operator switches rows with columns, i.e. transforms $(i, j) \rightarrow (j, i)$. As an example we transpose A:

$$\underline{\underline{\mathbf{A}}}^T = \begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$$

```python
print "define matrix A"
A = np.array([[1,2,3],[3,2,1],[-1,1,-1]])
print A
print "B is a matrix filled with ones:"
B = np.ones((3,3))
print B
print "I is the identity matrix"
I = np.eye(3)                   #identity matrix
print I
print "The product of AB =\n {0}".format(A.dot(B))
```

```
define matrix A
[[ 1  2  3]
 [ 3  2  1]
 [-1  1 -1]]
B is a matrix filled with ones:
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
I is the identity matrix
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
The product of AB =
 [[ 6.  6.  6.]
 [ 6.   6.   6.]
 [-1.  -1.  -1.]]
```

1.4 Inner Product between a vector and a matrix

The inner or matrix product is generated of the product of the *row* of the first matrix and the *columnn* of the second matrix. Thus multiplying a vector with a matrix can be either a row vector, **a**, with a matrix, $\underline{\underline{\mathbf{A}}}$, thus $\mathbf{a}\underline{\underline{\mathbf{A}}}$ or a matrix multiplied with a column vector, **b**, thus $\underline{\underline{\mathbf{A}}}\mathbf{b}$. The resulting vector keeps the shape of the initial vector, i.e. a row or column vector. The product is not communitative.

$$\mathbf{a}\underline{\underline{\mathbf{A}}} = (a_1, a_2, a_3) \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} a_1A_{11} + a_1A_{21} + a_1A_{31} \\ a_2A_{12} + a_2A_{22} + a_2A_{32} \\ a_3A_{13} + a_3A_{23} + a_3A_{33} \end{pmatrix}^T$$

$$\underline{\underline{\mathbf{A}}}\mathbf{b} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} b_1A_{11} + b_2A_{12} + b_3A_{13} \\ b_1A_{21} + b_2A_{22} + b_3A_{23} \\ b_1A_{31} + b_2A_{32} + b_3A_{33} \end{pmatrix}$$

Next we have a look how matrix-vector multiplications are done with Python:

```python
a=np.array([1,2,3]) #row vector
print"a is a row vector,\n a = {0}".format(a)
print "a A = {0}".format(b.dot(A)) #row vector times a matrix
print "A b = {0}".format(A.dot(b)) #matrix times a column vector
print "However if we take the transpose of the matrix and the transpose of the expression"
```

```
a is a row vector,
 a = [1 2 3]
a A = [3 2 1]
A b = [2 2 1]
However if we take the transpose of the matrix and the transpose of the expression
```

Homework:

1. Write the multiplication $\mathbf{a}\underline{\underline{\mathbf{A}}}$ and $\underline{\underline{\mathbf{A}}}\mathbf{b}$ using Einstein summation convention. Hint: Write it first as a sum using the $\sum$-symbol.
2. What happens in Python when you multiply a column vector to the left side of a 3x3 matrix?

```
Out[9]: <IPython.core.display.HTML object>
```