A quick primer on Python

Zulfikar Najmudin

30th January 2012

1 Introduction

I will be writing (hopefully!) a few small python programs to go along with the second part of the 2nd year electromagnetism course. Since the course is mostly about electromagnetic waves, and em waves are (very!) dynamic, I think playing with these little scripts is a good way to help visualise and understand the subject.

Python mostly began life as a scripting language, but it's ease of use (no type-casting, relatively easy syntax) has made it very popular, especially amongst scientists wanting a quick and easy language in which to do "dirty" analysis and data manipulation.

Now python has been extensively enhanced (through the use of modules) to be able to do much more complicated tasks, such as visualisation (2/3D), linear alegbra, symbolic maths etc.

Here we are only going to scratch the surface; writing small scripts to demonstrate dynamic effects, and most of the work will be done through the use of two major modules *numpy*, a module for numerics (which includes arrays, matrices etc.) and *matplotlib* which is primarily a 2D plotting library, but which does some 3D, some animation and some analysis too. (Additionally you can call a front-end called *pylab* which lets the two behave in a matlab like way).

Big advantages of python are that it runs on all commonly used platforms (mac, pc, linux) and it is free! So let's get started!

2 Installation

The easiest way to get python on your computer is to run the install the Enthought Python Distribution from Enthought inc., which installs all the libraries you could probably ever want in one easy installer from:

http://enthought.com/products/epd.php

Their prepackaged installer is free, if you register with them from an academic email address. But the installer is > 250 MB.

You can also do the installation of the basic packages yourself, if you don't want to install so much. You can install the main python installer here: http://python.org/download/ (please install the latest 32-bit version of 2.7 to be compatible with the other packages we will install).

Then install *numpy* from:

http://sourceforge.net/projects/numpy/files/

(you want the latest version 1.6.1 for python 2.7) and *matplotlib* from:

http://sourceforge.net/projects/matplotlib/files/

again we want the latest version 1.1.0 for python 2.7.

3 Beginning

3.1 Hello

Now that we've installed we are ready to begin. (I'm on a mac, but I assume this translates to other platforms easily). Find the python folder (in your "Applications" folder), and open the application IDLE. From the file menu, open a **New Window**. In it type:

```
print 'hello world!'
```

For this file **Save as** "hello.py". Then from the **run** menu, **run module**. Hopefully that should show up in the python shell. That was easy, wasn't it! Now for something a little bit more challenging. Lets plot a simple graph.

3.2 Sines

We are going to plot a simple sine curve in matplotlib.

First we need to import matplotlib (the plotting module). There are a number of ways to do this, and you will see many of them in examples on the web. Two examples are:

```
import numpy as np
```

In that case, to use any functions from numpy, then you have to prefix them with np, e.g. np.arange(0,10,0.1). This is a bit like methods in a C++ class, identifying functions with the module that brought them to the party.

But we are just going to be lazy and do:

```
from numpy import *
```

This just loads all the functions from numpy, so that you could use them as if they were native commands. NB at any time you can try out python commands in the shell - just to test them. So do:

```
from numpy import *
```

followed by

```
arange(0,10,0.1)
```

You should see that this will return an array of values from 0 to 10 in 0.1 steps. Here is the complete script, it should be obvious what it does:

```
from pylab import * # this imports matplotlib and numpy

x=arange(0,10,0.1) # arange is a numpy fn. to create series in an array
plot(x,sin(x)) # plot is a matplotlib function and sin a numpy one
show() # this shows the graph!
```

Note that here we didn't need to import *numpy* explicity since *pylab* does it for us (as well as *matplotlib* which is the plotting package). In the figure window there are icons to let you zoom, move and save the figure. Close the window when finished. At this point, you have a cute little graphing calculator. Try doing some of the following:

1. Plot cos(x)
2. Plot sin(x)*cos(x) - (note the change in the scale)
3. Plot sin(x)/x
4. Plot sin(x)/x, changing the range of x from -10 to 10

If at any time you feel stuck, well you know how to google. Additionally there is a "cookbook" of examples which you can find here http://www.scipy.org/Cookbook/Matplotlib.

3.3 Arrays

As you saw, plot required x and y to be in put in the form of arrays (as was generated by the numpy command, arange. *pylab* offers a whole load of array manipulations, you can create them, do simple maths on them term by term, or as vectors, or as collections of complex numbers, do Fourier transforms etc. If you want a full list of the things you can do with *numpy* arrays, look here http://mathesaurus.sourceforge.net/numeric-numpy.html.

At any time, you can see what is in an array x just by typing print x in your script (very useful for debugging!). (If your testing in the shell window, then you just need to type x). Try this little script which tests some of the properties of arrays,

```
from numpy import *
from pylab import *

x = arange(0,10,0.01) #makes a series from 0 to 10 in 0.1 steps
y = ones(size(x)) #makes array of ones same size as x, "zeros" would make array of zeros
z= y/exp(x)
a = random((size(z))) # demonstration of creating random numbers
b = a * z**2
```

```
ax1=subplot(2,1,1)
plot(x,y,x,z,x,a) # plots y,z and a against x

ax2=subplot(2,1,2)
hist(a,50) #makes a histogram of a

show()

print x[5], x[-5], x[:5], x[-5:] #prints 5th,5th from last, first 5 and last 5 elements of x

A = array([[1,1], [0,1]] )
B = array([[2,0], [3,4]] ) # two 2x2 matrices are then multiplied
print A*B #elementwise multiplication
print dot(A,B) #matrix multiplication
```

3.4 Animate

With matplolib 1.1.0 there are some new routines which allow you to animate graphs and save them (see the online help), but we are just going do some very simple animations. Type this:

```
from pylab import *

ion()

x = arange(0,3*pi,0.02)  # x-array
line, = plot(x,sin(x))
for i in xrange(1,200,0.1):
    line.set_ydata(sin(x-i)) # update the data
    draw() # redraw the canvas
```

In this example ion() stands for interactive_on mode (you can turn it off with ioff()!), this is what lets us animate the graph (along with the forced redrawing with the draw() function). Note the use of the constant pi in the definition of the x array (it's defined in *numpy*), and the comma in "line," indicates that line too is an array holding the line data. Note the example of a for loop; the for loop can cycle through any array that it is given (for example an array of filenames), but in this case it is an array of numbers, which is called using xrange rather than arange because in this case, the whole array is not saved in memory as it would be for arange - only the current value of the index is retained, as in typical use of a counter in a for loop in other languages. The body of the loop is only denoted by the colon and the indentation of the following commands (and you must make sure that the indentation is always the same for all lines of the loop). Note also that the method line.set_ydata(sin(x-i)) is used to update the line in each loop, rather than just calling plot again (this is because when you call plot, the figure just retains every new line and soon becomes a memory and processor hog.).

Run the script, you should see a forward moving sine wave. Also try:

1. Changing the sign of the i in the sine (to plus), you should see the wave is now backward travelling.

2. Plot both forward and backward travelling waves, and their sum - what kind of wave does the sum represent? (NB you will need to generate more line objects).

USEFUL MATHEMATICAL FORMULAS FOR E& M.

in cartesian form are:

(1) $\mathbf{A} = A_x\mathbf{i} + A_y\mathbf{j} + A_z\mathbf{k}$ and $\mathbf{B} = B_x\mathbf{i} + B_y\mathbf{j} + B_z\mathbf{k}$.

where $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are the unit vectors in the x, y and z direction. The elementary vextor operations are given by:

(2) **Addition**: $\mathbf{A} + \mathbf{B} = (A_x + B_x)\mathbf{i} + (A_y + B_y)\mathbf{j} + (A_z + B_z)\mathbf{k}$.
subtraction has the similar obvious definition.

(3) **Scalar Product**: $\mathbf{A} \cdot \mathbf{B} = (A_xB_x) + (A_yB_y) + (A_zB_z) = |\mathbf{A}||\mathbf{B}| \cos\theta$
where θ is the angle between $\mathbf{A}$ and $\mathbf{B}$.

(4) **Vector or Cross Product**: $\mathbf{A} \times \mathbf{B} = (A_yB_z - A_zB_y)\mathbf{i} + (A_zB_x - A_xB_z)\mathbf{j} + (A_xB_y - A_yB_x)\mathbf{k}$.

VECTOR CALCULUS IN CARTESIAN COORDINATES.

We define various operations useful in physics. First for a time dependent vector $\mathbf{A}(t)$:

(1) $$\frac{d\mathbf{A}}{dt} = \frac{dA_x}{dt}\mathbf{i} + \frac{dA_y}{dt}\mathbf{j} + \frac{dA_z}{dt}\mathbf{k}$$

Scalars or vectors which depend on space (e.g. $f = f(x, y, z)$ or $\mathbf{A} = \mathbf{A}(x, y, z)$) are called respectively scalar or vector fields. We define the *gradient* or *Del* operator:

(2) $$\nabla = \mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z}$$

We define the *Laplacian* operator:

(3) $$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

For scalar f and vector $\mathbf{A}$ we define the operators:

(4) **Gradient** (grad$f$): $$\nabla f = \mathbf{i}\frac{\partial f}{\partial x} + \mathbf{j}\frac{\partial f}{\partial y} + \mathbf{k}\frac{\partial f}{\partial z}$$

(5) **Divergence** (div$\mathbf{A}$): $$\nabla \cdot \mathbf{A} = \frac{\partial A_x}{\partial x} + \frac{\partial A_y}{\partial y} + \frac{\partial A_z}{\partial z}$$

(6) **Curl** (curl$\mathbf{A}$): $$\nabla \times \mathbf{A} = \mathbf{i}(\frac{\partial A_z}{\partial y} - \frac{\partial A_y}{\partial z}) + \mathbf{j}(\frac{\partial A_x}{\partial z} - \frac{\partial A_z}{\partial x}) + \mathbf{k}(\frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y})$$

VECTOR IDENTITIES.

Notation: f, g, are scalars; $\mathbf{A}$, $\mathbf{B}$, etc., are vectors.

(1)$\mathbf{A}\cdot\mathbf{B}\times\mathbf{C}=\mathbf{A}\times\mathbf{B}\cdot\mathbf{C}=\mathbf{B}\cdot\mathbf{C}\times\mathbf{A}=\mathbf{B}\times\mathbf{C}\cdot\mathbf{A}=\mathbf{C}\cdot\mathbf{A}\times\mathbf{B}=\mathbf{C}\times\mathbf{A}\cdot\mathbf{B}$

(2)$\mathbf{A}\times(\mathbf{B}\times\mathbf{C})=(\mathbf{C}\times\mathbf{B})\times\mathbf{A}=(\mathbf{A}\cdot\mathbf{C})\mathbf{B}-(\mathbf{A}\cdot\mathbf{B})\mathbf{C}$

(3)$\mathbf{A}\times(\mathbf{B}\times\mathbf{C})+\mathbf{B}\times(\mathbf{C}\times\mathbf{A})+\mathbf{C}\times(\mathbf{A}\times\mathbf{B})=0$

(4) $(\mathbf{A}\times\mathbf{B})\cdot(\mathbf{C}\times\mathbf{D})=(\mathbf{A}\cdot\mathbf{C})(\mathbf{B}\cdot\mathbf{D})-(\mathbf{A}\cdot\mathbf{D})(\mathbf{B}\cdot\mathbf{C})$

(5) $(\mathbf{A}\times\mathbf{B})\times(\mathbf{C}\times\mathbf{D})=(\mathbf{A}\times\mathbf{B}\cdot\mathbf{D})\mathbf{C}-(\mathbf{A}\times\mathbf{B}\cdot\mathbf{C})\mathbf{D}$

(6) $\nabla(fg)=\nabla(gf)=f\nabla g+g\nabla f$

(7) $\nabla\cdot(f\mathbf{A})=f\nabla\cdot\mathbf{A}+\mathbf{A}\cdot\nabla f$

(8) $\nabla\times(f\mathbf{A})=f\nabla\times\mathbf{A}+\nabla f\times\mathbf{A}$

(9) $\nabla\cdot(\mathbf{A}\times\mathbf{B})=\mathbf{B}\cdot\nabla\times\mathbf{A}-\mathbf{A}\cdot\nabla\times\mathbf{B}$

(10)$\nabla\times(\mathbf{A}\times\mathbf{B})=\mathbf{A}(\nabla\cdot\mathbf{B})-\mathbf{B}(\nabla\cdot\mathbf{A})+(\mathbf{B}\cdot\nabla)\mathbf{A}-(\mathbf{A}\cdot\nabla)\mathbf{B}$

(11) $\mathbf{A}\times(\nabla\times\mathbf{B})=(\nabla\mathbf{B})\cdot\mathbf{A}-(\mathbf{A}\cdot\nabla)\mathbf{B}$

(12) $\nabla(\mathbf{A}\cdot\mathbf{B})=\mathbf{A}\times(\nabla\times\mathbf{B})+\mathbf{B}\times(\nabla\times\mathbf{A})+(\mathbf{A}\cdot\nabla)\mathbf{B}+(\mathbf{B}\cdot\nabla)\mathbf{A}$

(13) $\nabla^2 f=\nabla\cdot\nabla f$

(14) $\nabla^2\mathbf{A}=\nabla(\nabla\cdot\mathbf{A})-\nabla\times\nabla\times\mathbf{A}$

(15) $\nabla\times\nabla f=0$

(16) $\nabla\cdot\nabla\times\mathbf{A}=0$

Let $\mathbf{r}=\mathbf{i}x+\mathbf{j}y+\mathbf{k}z$ be the radius vector of magnitude$r$, from the origin to the point x,y,z. Then

(17) $\nabla\cdot\mathbf{r}=3$

(18) $\nabla\times\mathbf{r}=0$

(19) $\nabla r=\mathbf{r}/r$

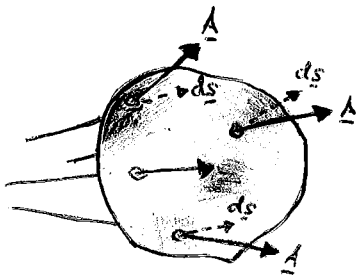
(20) $\nabla(1/r)=-\mathbf{r}/r^3$

(21) $\nabla\cdot(\mathbf{r}/r^3)=4\pi\delta(\mathbf{r})$

VECTOR INTEGRAL THEOREMS.

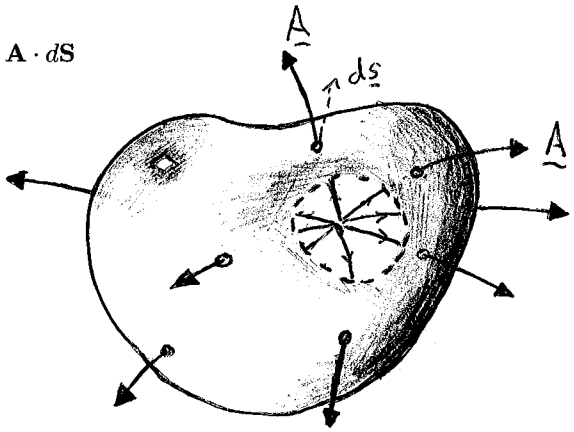
The flux Φ of a vector field $\mathbf{A}$ through a surface S is given by:

(1) $\Phi = \int_S \mathbf{A} \cdot d\mathbf{S}$



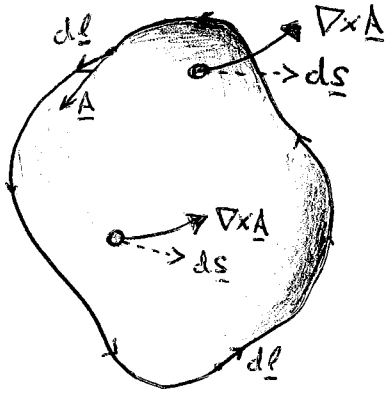
GAUSS'S THEOREM – DIVERGENCE THEOREM For a closed surface S enclosing a volume V with $d\mathbf{S}$ the differential area pointing outward and dV the differential volume element.

(2) $\int_V (\nabla \cdot \mathbf{A}) dV = \oint_S \mathbf{A} \cdot d\mathbf{S}$



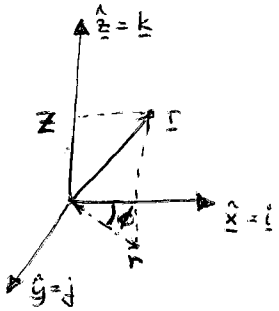
STOKE'S THEOREM – CURL THEOREM For any surface S spanning a closed loop l with $d\mathbf{l}$ the differential length around the loop and $d\mathbf{S}$ the differential area. The sign convention is that length increases in the anticlockwise direction if $d\mathbf{S}$ points towards us – the right hand rule.

(3) $\int_S (\nabla \times \mathbf{A}) \cdot d\mathbf{S} = \oint_l \mathbf{A} \cdot d\mathbf{l}$



DIFFERENTIAL OPERATORS IN CURVILINEAR COORDINATES[5]

Cylindrical Coordinates



$x = r\cos\phi$

$y = r\sin\phi$

$z = z$

Divergence $$\nabla \cdot \mathbf{A} = \frac{1}{r}\frac{\partial}{\partial r}(rA_r) + \frac{1}{r}\frac{\partial A_\phi}{\partial \phi} + \frac{\partial A_z}{\partial z}$$

Gradient $$(\nabla f)_r = \frac{\partial f}{\partial r}; \quad (\nabla f)_\phi = \frac{1}{r}\frac{\partial f}{\partial \phi}; \quad (\nabla f)_z = \frac{\partial f}{\partial z}$$

Curl $$(\nabla \times \mathbf{A})_r = \frac{1}{r}\frac{\partial A_z}{\partial \phi} - \frac{\partial A_\phi}{\partial z}$$

$$(\nabla \times \mathbf{A})_\phi = \frac{\partial A_r}{\partial z} - \frac{\partial A_z}{\partial r}$$

$$(\nabla \times \mathbf{A})_z = \frac{1}{r}\frac{\partial}{\partial r}(rA_\phi) - \frac{1}{r}\frac{\partial A_r}{\partial \phi}$$

Laplacian $$\nabla^2 f = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 f}{\partial \phi^2} + \frac{\partial^2 f}{\partial z^2}$$

Laplacian of a vector $$(\nabla^2 \mathbf{A})_r = \nabla^2 A_r - \frac{2}{r^2}\frac{\partial A_\phi}{\partial \phi} - \frac{A_r}{r^2}$$

$$(\nabla^2 \mathbf{A})_\phi = \nabla^2 A_\phi + \frac{2}{r^2}\frac{\partial A_r}{\partial \phi} - \frac{A_\phi}{r^2}$$

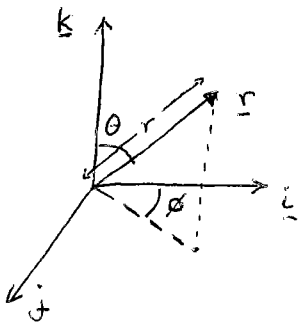
$$(\nabla^2 \mathbf{A})_z = \nabla^2 A_z$$

Components of $(\mathbf{A}\cdot\nabla)\mathbf{B}$ $$(\mathbf{A} \cdot \nabla \mathbf{B})_r = A_r\frac{\partial B_r}{\partial r} + \frac{A_\phi}{r}\frac{\partial B_r}{\partial \phi} + A_z\frac{\partial B_r}{\partial z} - \frac{A_\phi B_\phi}{r}$$

$$(\mathbf{A} \cdot \nabla \mathbf{B})_\phi = A_r\frac{\partial B_\phi}{\partial r} + \frac{A_\phi}{r}\frac{\partial B_\phi}{\partial \phi} + A_z\frac{\partial B_\phi}{\partial z} + \frac{A_\phi B_r}{r}$$

$$(\mathbf{A} \cdot \nabla \mathbf{B})_z = A_r\frac{\partial B_z}{\partial r} + \frac{A_\phi}{r}\frac{\partial B_z}{\partial \phi} + A_z\frac{\partial B_z}{\partial z}$$

Spherical Coordinates



$x = r\sin\theta\cos\phi$

$y = r\sin\theta\sin\phi$

$z = r\cos\phi$

Divergence $\nabla \cdot \mathbf{A} = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 A_r) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta A_\theta) + \frac{1}{r\sin\theta}\frac{\partial A_\phi}{\partial\phi}$

Gradient $(\nabla f)_r = \frac{\partial f}{\partial r}; \quad (\nabla f)_\theta = \frac{1}{r}\frac{\partial f}{\partial\theta}; \quad (\nabla f)_\phi = \frac{1}{r\sin\theta}\frac{\partial f}{\partial\phi}$

Curl $(\nabla \times \mathbf{A})_r = \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta A_\phi) - \frac{1}{r\sin\theta}\frac{\partial A_\theta}{\partial\phi}$

$$(\nabla \times \mathbf{A})_\theta = \frac{1}{r\sin\theta}\frac{\partial A_r}{\partial\phi} - \frac{1}{r}\frac{\partial}{\partial r}(rA_\phi)$$

$$(\nabla \times \mathbf{A})_\phi = \frac{1}{r}\frac{\partial}{\partial r}(rA_\theta) - \frac{1}{r}\frac{\partial A_r}{\partial\theta}$$

Laplacian $\nabla^2 f = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial f}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial f}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial\phi^2}$

Laplacian of a vector $(\nabla^2\mathbf{A})_r = \nabla^2 A_r - \frac{2A_r}{r^2} - \frac{2}{r^2}\frac{\partial A_\theta}{\partial\theta} - \frac{2\cot\theta A_\theta}{r^2} - \frac{2}{r^2\sin\theta}\frac{\partial A_\phi}{\partial\phi}$

$$(\nabla^2\mathbf{A})_\theta = \nabla^2 A_\theta + \frac{2}{r^2}\frac{\partial A_r}{\partial\theta} - \frac{A_\theta}{r^2\sin^2\theta} - \frac{2\cos\theta}{r^2\sin^2\theta}\frac{\partial A_\phi}{\partial\phi}$$

$$(\nabla^2\mathbf{A})_\phi = \nabla^2 A_\phi - \frac{A_\phi}{r^2\sin^2\theta} + \frac{2}{r^2\sin\theta}\frac{\partial A_r}{\partial\phi} + \frac{2\cos\theta}{r^2\sin^2\theta}\frac{\partial A_\theta}{\partial\phi}$$

Components of $(\mathbf{A}\cdot\nabla)\mathbf{B}$ $(\mathbf{A}\cdot\nabla\mathbf{B})_r = A_r\frac{\partial B_r}{\partial r} + \frac{A_\theta}{r}\frac{\partial B_r}{\partial\theta} + \frac{A_\phi}{r\sin\theta}\frac{\partial B_r}{\partial\phi} - \frac{A_\theta B_\theta + A_\phi B_\phi}{r}$

$$(\mathbf{A}\cdot\nabla\mathbf{B})_\theta = A_r\frac{\partial B_\theta}{\partial r} + \frac{A_\theta}{r}\frac{\partial B_\theta}{\partial\theta} + \frac{A_\phi}{r\sin\theta}\frac{\partial B_\theta}{\partial\phi} + \frac{A_\theta B_r}{r} - \frac{\cot\theta A_\phi B_\phi}{r}$$

$$(\mathbf{A}\cdot\nabla\mathbf{B})_\phi = A_r\frac{\partial B_\phi}{\partial r} + \frac{A_\theta}{r}\frac{\partial B_\phi}{\partial\theta} + \frac{A_\phi}{r\sin\theta}\frac{\partial B_\phi}{\partial\phi} + \frac{A_\phi B_r}{r} + \frac{\cot\theta A_\phi B_\theta}{r}$$