Scientific Computing in Python 4 – Solving Differential Equations

Differential equations describe how something changes over time, this is the most efficient way to model our changing parameters.

Model parameters can then feed into each other provided we understand the underlying relationships between them.

First of all we are going to learn how to solve differential Equations in python.

Realistically we would just use an ODE solver, there are many built into various python modules but for the sake of practicing our maths and seeing how they work we will solve one by hand and then iterate it through a series of values.

Lets look at a differential equation. How about our red panda one.

dN/dt = rN

Unfortunately there isn't a function for rearranging equations we can quickly make so we will do that bit ourself.

1/N dN = r dt

Ok we have a separated variable differential equation. Now we have to integrate both sides.

That would be a function that might be a bit complex for us to write. So lets use a function already made

```
from sympy import *

import sympy.integrals as integ
```

note we are using a module called sympy here which allows us to integrate indefinite integrals. Normally we'd just use scipy and an ODE solver.

First of all we have to create mathematical symbol objects for sympi

```
N = Symbol('N')

r = Symbol('r')

t = Symbol('t')
```

Now we have that we can specify our Integrals

```
solvN = inte.integrate(1/N,N)

solvr = inte.integrate(r,t)
```

Now we have that we have our solution quickly simplify it by exp both sides.

```
solvN = exp(solvN)

solvr = exp(solvr)
```

now we have N = e^rt, however this is missing out constant c, in this case we just have to add it in now we want to iterate through. Remember that constant c in this case will be your starting population size. Lets write a function to iterate through 100 timesteps of a population with a starting size of 2 and a 0.1 growth rate.

```
def readpandas(timesteps,starting,growth):
    popsize= []
    step = []
    for i in range(timesteps):
        popsize.append(starting * exp(growth * i))
        step.append(i)
    plt.plot(step,popsize)
    return(popsize)


readpandas(100,2,0.1)
```

And there we solved a differential equation from 0 to 100 timesteps of an increasing population of red pandas.

Lets try and do it using an ODE solver already made by someone in scipy.

```
import scipy
from scipy.integrate import solve_ivp


def pandagrowth(t,y):
    return(0.1 * y)


sol = solve_ivp(pandagrowth, [0,100], [2],t_eval=range(0,100))
print(sol.y)
tstep = []
tstep.extend(range(0,100))
plt.plot(tstep,sol.y[0])
```

And there we go, we get the exact same plot and answers without doing all that maths ourselves, this is much easier.

Problem 1: By hand solve our panda equation with a carrying capacity, then compare it to our ODE solver.

Challenge Problem

Problem 2: The carrying capacity of our population is due to food limitation not just a general carrying capacity. The current carrying capacity is determined by the amount of food. It takes a relationship between the number of cabbages that are available to eat. Cabbages have their own fixed growth rate and population size, but this is dependent on the number of red pandas.

Design an ODE that describes the population size of cabbages for a fixed population of red pandas.

We start with 10000 cabbages.

Fixed 100 red pandas.

Cabbages grow at a rate of 0.3 cabbages per cabbage per timestep.

Each panda eats 2 cabbages per timestep.

Soil nutrients limit the possible number of cabbages that can be grown to maximum of 15000 cabbages.

Design this ODE and then run it over 1000 timesteps. Plot it out, tell me if the population of red pandas of that size is sustainable. Find the sustainable point which maximises number of Red Pandas.