Object-Oriented Programs

1 Basic Concepts

The data in the sequence represent some relevant property of the model and are expressed as a variable $s$. An individual value of variable $s$ is known as a term in the sequence and is denoted by $s_n$. A sequence with $m$ data values or terms, is written as follows:

$$\langle s_1, s_2, s_3, s_4, s_5, \ldots, s_m \rangle$$

In models with geometric growth, the data increase (or decrease) by an equal percentage or growth factor in equal intervals of time. The difference equation that represents the pattern of geometric growth has the general form:

$$s_{n+1} = c \cdot s_n$$

(1)

In Equation 1, the parameter $c$ is constant and represents the growth factor, and $n$ identifies an individual value such that $n \leq m$. With geometric growth, the data increases or decreases by a fixed factor in equal intervals.

1.1 Increasing Data with Geometric Growth

The data in a sequence will successively increase in value when the value of the growth factor is greater than 1. For example, consider a data sequence that exhibits geometric growth with a growth factor of 1.45 and a starting value of 50.0. The sequence with 8 terms is:

$$\langle 50.0, 72.5, 105.125, 152.43, 221.02, 320.48, 464.70, 673.82 \rangle$$

Figure 1 shows a graph of the data with geometric growth. Note that the data increases rapidly.

1.2 Decreasing Data with Geometric Growth

The data in a sequence will successively decrease in value when the value of the growth factor is less than 1. For example, consider a data sequence that exhibits geometric growth with a growth factor of 0.65 and a starting value of 850.0. The sequence with 10 terms is:

$$\langle 850.0, 550.0, 362.5, 238.125, 157.03125, 100.87, 66.589, 43.597375, 28.958859375, 19.137765625 \rangle$$
Figure 1: Data with geometric growth.

\[ \langle 850.0, 552.5, 359.125, 233.43, 151.73, 98.62, 64.10, 41.66, 27.08, 17.60 \rangle \]

Figure 2 shows a graph of the data with geometric growth. Note that the data decreases rapidly because the growth factor is less than 1.0.

### 1.3 Case Study 1

The population of a small town is recorded every year; the increases per year are shown in Table 1, which gives the data about the population during the years from 1995 to 2003. The table also shows the population growth factor.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop.</td>
<td>81</td>
<td>90</td>
<td>130</td>
<td>175</td>
<td>206</td>
<td>255</td>
<td>288</td>
<td>394</td>
<td>520</td>
</tr>
<tr>
<td>Fac.</td>
<td>-</td>
<td>1.111</td>
<td>1.444</td>
<td>1.346</td>
<td>1.177</td>
<td>1.237</td>
<td>1.129</td>
<td>1.368</td>
<td>1.319</td>
</tr>
</tbody>
</table>

Note that although the growth factors are not equal, the data can be considered to grow in a geometric pattern. The values of the growth factor shown in the table are sufficiently close and the average growth factor calculated is 1.2667.

© 2015 J. M. Garrido
After a brief analysis of the data in the problem, the following tasks are to be computed: (1) create the data lists (arrays) of the sequence $s$ with the values of the original data in Table 1; (2) compute the average growth factor from the data in the table; (3) compute the values of a second data list $y$ using 1.267 as the average growth factor and the difference equation $y_{n+1} = 1.267 \ y_n$, and (4) plot the graphs.

Listing 1 shows the Python program that performs these tasks and is stored in file: `popstown.py`. The factors of the population data are computed in line 19 with a call to function `factors` and average growth factor is computed in line 22 of the program. The computed data is calculated in lines 25–27.

Listing 1: Program that computes the population average growth factor.

```python
1 # program: popstown.py
2 # This program computes the growth factor per year
3 # of the population.
4 # Uses NumPy
5 # J Garrido 09-2-2014
6
7 import numpy as np
8
9 def factors(marray): # Compute factors in marray
10     n = marray.size
11     mf = np.zeros(n-1) # array of factors
12     for j in range(n-1):
13         c
```

© 2015 J. M. Garrido
```
12    mf[j] = marray[j+1]/marray[j]
13    return mf
14
15    N = 9
16    s = np.array([81.0, 90.0, 130.0, 175.0, 206.0, 255.0, 288.0, 394.0, 520.0])
18    print "Program to compute average growth factor"
19    f = factors(s) # compute array of factors
20    print "Factors in array s: \n"
21    print f
22    meanv = np.mean(f)
23    print "Mean factor: ", meanv
24    cs = np.zeros(N)
25    cs[0] = s[0]
26    for j in range(N-1):
27        cs[j+1] = meanv * cs[j];
28    print "Given and Computed values of population"
29    for j in range(N):
30        print x[j], s[j], cs[j]
```

When the Python interpreter processes the program, the following listing is produced.

Program to compute average growth factor

Factors in array s:

```
[ 1.11111111 1.44444444 1.34615385 1.17714286 1.23786408
  1.12941176 1.36805556 1.31979695]
```

Mean factor: 1.26674757639

Given and Computed values of population

1995  81.0  81.0
1996  90.0  102.606553687
1997 130.0  129.976603205
1998 175.0  164.647547097
1999 206.0  208.566881243
2000 255.0  264.201591329
2001 288.0  334.676725494
2002 394.0  423.950930893
2003 520.0  537.038814216
Figure 3: Population of a small town for 1995–2003.

Figure 3 shows a graph with two curves; one with the population in the town by year from 1995 through 2003 taken directly from Table 1. The other curve shown in the graph of Figure 3 is the computed data applying Equation 1 with 1.267 as the growth factor, using the difference equation \( s_{n+1} = 1.267 \cdot s_n \).

A very similar python program is stored in file popstown2.py, which reads the population data from a text file and produces the same results.

1.4 Case Study 2

In a water treatment process, every application of solvents removes 65% of impurities from the water to make it more acceptable for human consumption. This treatment has to be performed several times until the level of purity of the water is adequate for human consumption. Assume that when the water has less than 0.6 parts per gallon of impurities, it is adequate for human consumption.

In this problem, the data given is the contents of impurities in parts per gallon of water. The initial data is 405 parts per gallon of impurities and the growth factor is 0.35.

Listing 2 shows a Python program that computes the impurities of water after each application of the solvent. The program declares the data lists (arrays) of the sequence \( s \) with the data of the contents of impurities in parts per gallon of water. The output data produced by executing the program and GnuPlot is used to plot the graphs. The program is stored in file watertr.py.
Listing 2: Program that computes the impurities of water.

```python
# program: watertr.py
# This program computes the impurities in water
given the growth factor.
# Uses NumPy
# J Garrido 9-2-2014
import numpy as np

N = 10  # Number of applications
f = 0.35  # constant factor
s = np.zeros(N)  # create array with zeros
s[0] = 405.0  # initial impurity of water
print "Program to compute growth of impurities in water\n"
napp = np.arange(N)
napp = napp + 1  # application number
print "Initial impurity in water: ", s[0]
print "Number of applications: ", N
print "Factor: ", f
for j in np.arange(N-1):  # compute impurities
    s[j+1] = f * s[j]
print "Impurities after each application \n"
for j in np.arange(N):
    print napp[j], s[j]
```

When the Python interpreter processes the program, the following output listing is produced. Figure 4 shows the graph of the impurities in parts per gallon of water for several applications of solvents.

Program to compute growth of impurities in water

Initial impurity in water: 405.0
Number of applications: 10
Factor: 0.35

Impurities after each application

```
1 405.0
2 141.75
3 49.6125
4 17.364375
5 6.07753125
6 2.1271359375
7 0.744497578125
```
Functional Equations in Geometric Growth

From the difference equation for models with geometric growth, Equation 1, the first few terms of sequence $s$ can be written as:

$$s_2 = cs_1$$
$$s_3 = cs_2 = c(cs_1)$$
$$s_4 = cs_3 = c(c(cs_1))$$
$$s_5 = cs_4 = c(c(c(cs_1))))$$
$$s_6 = cs_5 = c(c(c(c(cs_1))))$$
$$\cdots$$
$$s_n = c^{n-1} s_1$$

Equation 1 was referenced by substituting $s_{n-1}$ for its difference equation, and continuing this procedure up to $s_n$. In this manner, a functional equation can be derived. Recall that a functional equation gives the value of a term $s_n$ without using the previous value, $s_{n-1}$. The following mathematical expression is a general functional equation for geometric growth models.
Equation 2 gives the value \( s_n \) as a function of \( n \) for a geometric growth model, with \( n \geq 1 \). Note that this functional equation includes the fixed value \( s_1 \), which is the value of the first term of the data sequence.

A functional equation such as Equation 2 is an example of an exponential function because the independent variable, \( n \), is the exponent. This type of growth in the data is also known as exponential growth.

Functional equations can be used to answer additional questions about a model. For example: what will the population be 12 years from now? What amount of impurities are left in the water after 8 repetitions of the application of solvents?

When the growth factor does not correspond to the desired unit of time, then instead of \( n \), a more appropriate variable can be used. For example the first population data in Case Study 1, Section 1.3, the variable \( n \) represents number of years. To deal with months instead of years, a small substitution in the functional equation is needed. Variable \( t \) will represent time, and the starting point of the data is at \( t = 0 \) with an initial value of \( y_0 \). This gives meaning to the concept of a continuous model. Because one year has 12 months and using the same growth factor \( c \) as before, the following is a modified functional equation can be applied when dealing with months.

\[
y(t) = y_0 c^{t/12}
\]  

(3)