Project_02_exponential

September 20, 2023

0.1 Exponential growth

Exponential growth (or decay) is characterized by the rate equation $y' = ry$. Many things in the world experience exponential growth: populations of organisms, money invested in an account earning interest, radioactive elements, etc.

0.1.1 Problem 1

Spokane Teachers Credit Union (STCU) offers a certificate of deposit (CD) with an annual percent rate (APR) of 5.38% per year with a minimum investment of $2,000 (see **the STCU site**). 1. Use the APR of 5.38% to write a rate equation $y' = ry$ for the amount of money in the CD. 1. Suppose you invest $2,000. Assume interest is compounded yearly and calculate the value of the CD after 1, 2, and 5 years.

0.1.2 Answers:

1. $y' = 0.0538y$
2. Code below

[5]:
```
y0=2000
interest1=0.0538*2000

y1=y0+interest1
print(y1)

interest2=0.0538*y1
y2=y1+interest2
print(y2)
```

```
2107.6
2220.98888
```

[7]:
```
y=[2000]
for n in range(5):
    y.append(y[n]+0.0538*y[n])
print(y)
print('CD value after 1 year is', y[1])
print('CD value after 2 years is', y[2])
print('CD value after 5 years is', y[5])
```

```
[2000, 2107.6, 2220.98888, 2340.478081744, 2466.3958025418274,
2599.0878967185777]
CD value after 1 year is 2107.6
CD value after 2 years is 2220.98888
CD value after 5 years is 2599.0878967185777
```

0.1.3 Problem 2

Interest on STCU CDs is actually compounded monthly. Repeat your last calculations but use monthly compounding instead of yearly to calculate values for the CD. Do you earn more interest with monthly compounding or yearly compounding?

[9]:
```
y=[2000]
for n in range(12*5):
    y.append(y[n]+(0.0538/12)*y[n])
#print(y)
print('CD value after 1 year is', y[12])
print('CD value after 2 years is', y[24])
print('CD value after 5 years is', y[60])
```

```
CD value after 1 year is 2110.293290679847
CD value after 2 years is 2226.6688863441886
CD value after 5 years is 2615.7371965467464
```

0.1.4 Problem 3

Now calculate the value of your $2,000 initial investment after 30 years with: - yearly compounding - monthly compounding - daily compounding

How much interest did the CD earn in the 30th year?

[11]:
```
# Yearly compounding:
yy=[2000]
for n in range(30):
    yy.append(yy[n]+0.0538*yy[n])
print('CD value after 30 years with yearly compounding is', yy[30])

# Monthly compounding:
ym=[2000]
for n in range(12*30):
    ym.append(ym[n]+(0.0538/12)*ym[n])
print('CD value after 30 years with monthly compounding is', ym[12*30])

# Daily compounding:
yd=[2000]
for n in range(365*30):
    yd.append(yd[n]+(0.0538/365)*yd[n])
print('CD value after 30 years with daily compounding is', yd[365*30])
```

```
CD value after 30 years with yearly compounding is 9633.316358462405
CD value after 30 years with monthly compounding is 10009.55271264599
CD value after 30 years with daily compounding is 10044.530350889812
```

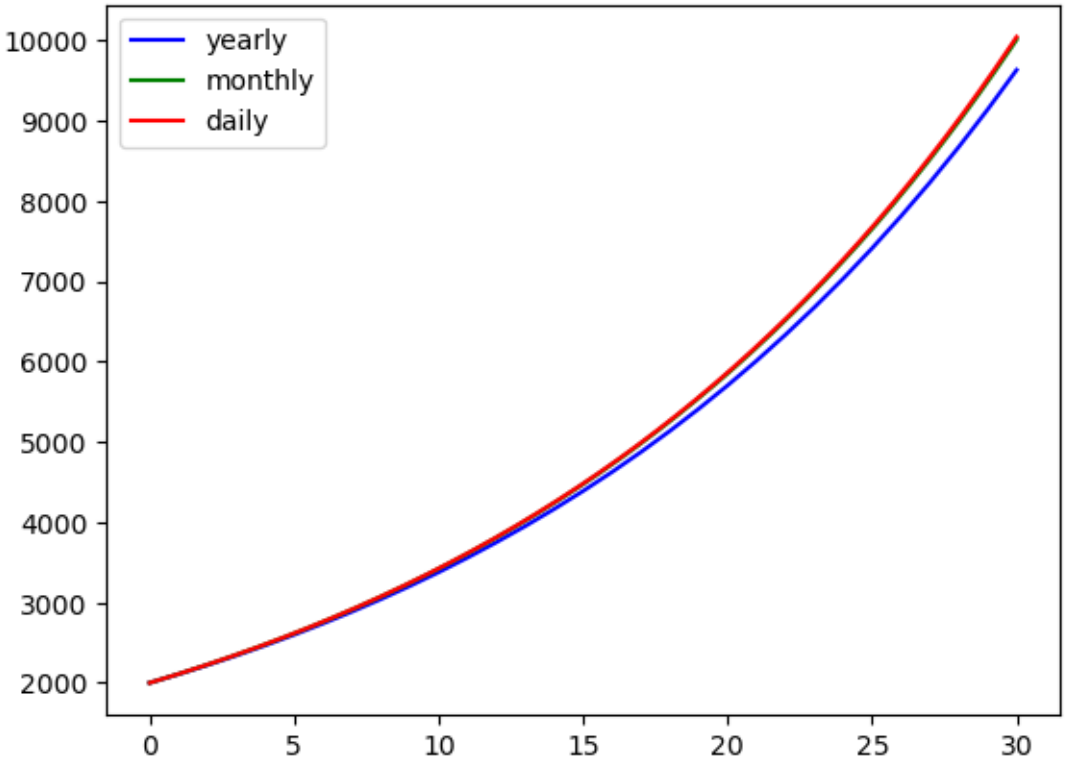
[16]:
```python
import matplotlib.pyplot as plt

xy=list(range(30))
xy.append(30)
#print(xy)

xm=[]
for i in range((12*30)+1):
    xm.append(i/12)
#print(xm)

xd=[]
for i in range((365*30)+1):
    xd.append(i/365)
#print(xd[365*30])

plt.plot(xy,yy, color='blue', label='yearly')
plt.plot(xm, ym, color='green', label='monthly')
plt.plot(xd,yd, color='red', label='daily')
plt.legend(loc='upper left')
plt.show()
```



[18]:
```
for m in [1, 12, 365, 1000, 10000]:
    y=[2000]
    for n in range(m*30):
        y.append(y[n]+(0.0538/m)*y[n])
    print('CD value after 30 years with compounding', m, 'times per year is',␣
 ↪y[m*30])
```

```
CD value after 30 years with compounding 1 times per year is 9633.316358462405
CD value after 30 years with compounding 12 times per year is 10009.55271264599
CD value after 30 years with compounding 365 times per year is
10044.530350889812
CD value after 30 years with compounding 1000 times per year is
10045.288971184706
CD value after 30 years with compounding 10000 times per year is
10045.68148243095
```

Most accounts aren't actually going to offer daily compounding (though this might be a reasonable model for a stock price in some situations). Frequent compounding makes more sense for things that change continuously: the population of an organism or the decay of a radioactive element are classic examples. In these cases frequent compounding is the same as small step size for the SIR model (that is, it increases the accuracy of our predictions).

0.1.5 Problem 4

Estimates for the current growth rate of the human population are about 0.88% per year (see **Worldometer**). 1. Use this to write a rate equation $P' = rP$ (where P is human population). 2. Use the current human population to predict the population in 2 years, 5 years, 50 years, and 100 years. Make sure to use frequent compounding (small step size). Are all the predictions equally reasonable? Why or why not? Address assumptions behind the model.

1. $P' = 0.0088P$
2. See the code below. Predictions over the short term (2 or 5 years) are probably reasonable. Over the long term we should expect the growth rate to change. Our model assumes a constant growth rate of 0.88%; all evidence points to declining growth rates. This is good since a population of 19 billion seems like it might stress our ability to feed and care for people. Note that there are two main ways to reduce the growth rate: reduce the number of births or increase the number of deaths.

[40]:
```
P=[8061799955]
for n in range(100):
    P.append(P[n]+0.0088*P[n])
print('Predicted population after 2 years (one step per year):', P[2])
print('Predicted population after 5 years (one step per year):', P[5])
print('Predicted population after 50 years (one step per year):', P[50])
print('Predicted population after 100 years (one step per year):', P[100])
```

```
P=[8061799955]
s=1000
for n in range(100*s):
    P.append(P[n]+(0.0088/s)*P[n])
print('Predicted population after 2 years (', s, 'steps per year):', P[2*s])
print('Predicted population after 5 years (', s, 'steps per year):', P[5*s])
print('Predicted population after 50 years (', s, 'steps per year):', P[50*s])
print('Predicted population after 100 years (', s, 'steps per year):', P[100*s])
```

```
Predicted population after 2 years (one step per year): 8204311939.996515
Predicted population after 5 years (one step per year): 8422817391.9711895
Predicted population after 50 years (one step per year): 12493545294.424696
Predicted population after 100 years (one step per year): 19361516645.7999
Predicted population after 2 years ( 1000 steps per year): 8204942967.930509
Predicted population after 5 years ( 1000 steps per year): 8424437070.653877
Predicted population after 50 years ( 1000 steps per year): 12517590750.38603
Predicted population after 100 years ( 1000 steps per year): 19436115888.35927
```

0.1.6 Problem 5

During the 1986 Chernobyl nuclear disaster, and estimated 27 kg of radioactive Cesium-137 was released into the environment. Cesium-137 has a half-life of about 30 years: this means it satisfies the rate equation $C' = -0.0231C$ (where C is the amount of Cesium-137). Estimate as precisely as you can the amount of this Cesium-137 remaining in the environment today.

[46]:
```
C=[27]
for n in range(37):
    C.append(C[n]-0.0231*C[n])
print('Predicion (one step per year):', C[37])

C=[27]
s=1000
for n in range(37*s):
    C.append(C[n]-(0.0231/s)*C[n])
print('Predicion (', s, 'steps per year):', C[37*s])
```

```
Predicion (one step per year): 11.371501670042617
Predicion ( 1000 steps per year): 11.485978080428058
```