# School year 2021-2022 - Semester 1

# Calculus 1 - Class CC15

# Group 7

**<u>Topic 7A: Differential Equations:</u> Direction Fields and Euler's Method, Exponential Growth and Decay**

**List of all members:**

-Nguyễn Văn Đức Long, 2153535, long.nguyencbct@hcmut.edu.vn

-Nguyễn Đăng Khoa, 2153465, khoa.nguyenmeeh11@hcmut.edu.vn

References

-Nguyễn Hoàng An, 2052824, an.nguyenunknown@hcmut.edu.vn

e.g.,
[1] J. Steward, Calculus. Concepts and Contexts, 7 th ed., Thomson Learning, 2012.

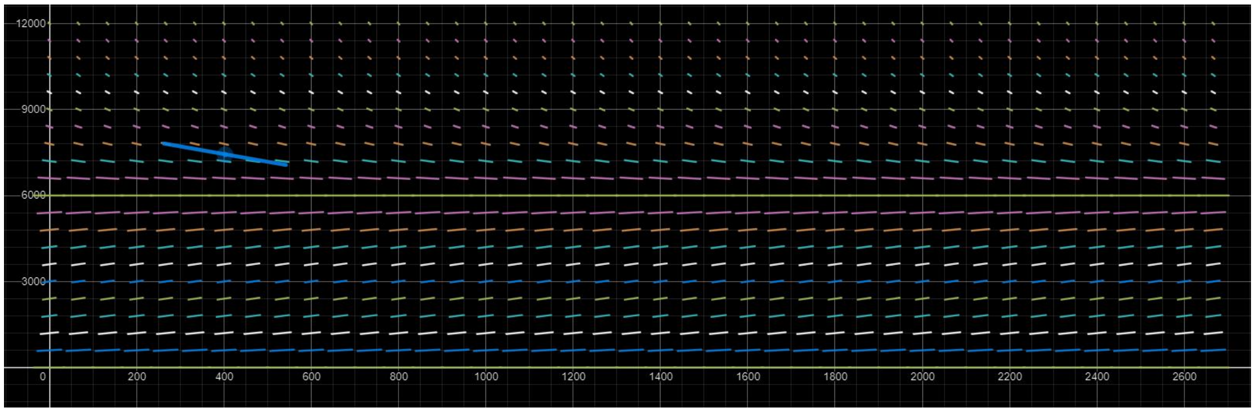
[2] Scilab

[3] Matlab

**Section 9.4, Exercise 2: Nguyễn Văn Đức Long**

a/ $\frac{dP}{dt} = 0.0015P(1 - \frac{P}{6000})$

b/

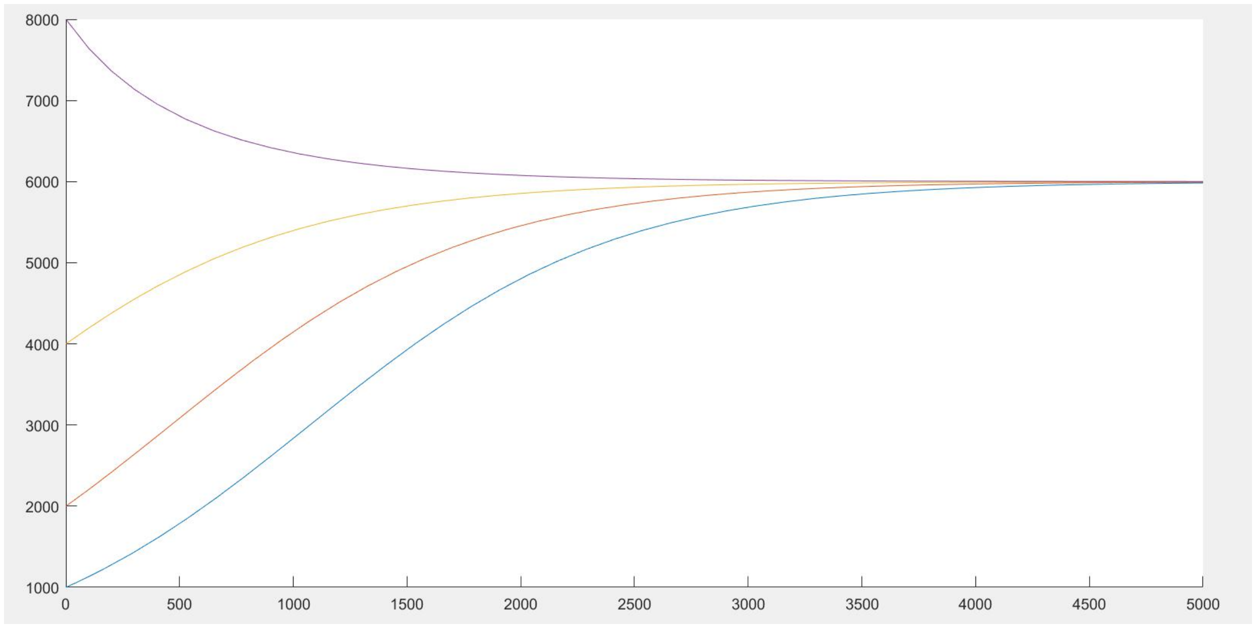


It tells us the slope of the curve when it passes through a certain point

C/

Matlab code

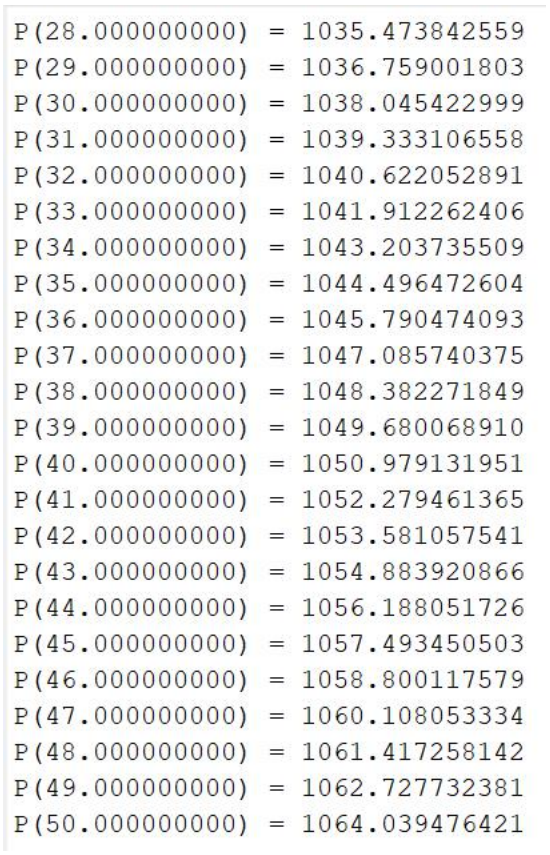
```
hold on
F = @(t,P)(0.0015*P.*(1-P./6000));
[t,P1] = ode45(F,[0 5000],1000);
plot(t,P1)
[t,P2] = ode45(F,[0 5000],2000);
plot(t,P2)
[t,P3] = ode45(F,[0 5000],4000);
plot(t,P3)
[t,P4] = ode45(F,[0 5000],8000);
plot(t,P4)
```



D/Matlab code:

```
f = @(t,P) 0.0015*P*(1-P/6000);
 t0 = 0;
 P0 = 1000;
 h = 1;
 tn = 50;
 n = (tn-t0)/h;
t(1)= t0; P(1) = P0;
for i=1:n
    P(i+1) = P(i)+ h*f(t(i),P(i));
    t(i+1) = t0 + i*h;
    fprintf('P(%.9f) = %.9f\n' ,t(i+1),P(i+1))
End
```

Output:



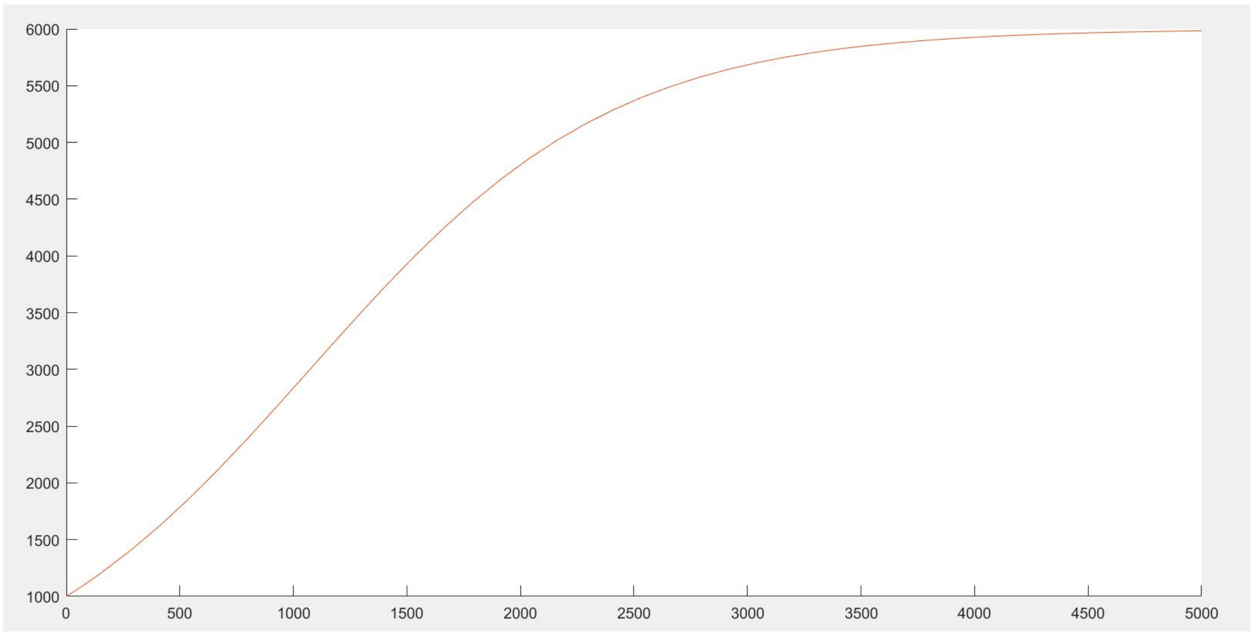
```
P(28.000000000) = 1035.473842559
P(29.000000000) = 1036.759001803
P(30.000000000) = 1038.045422999
P(31.000000000) = 1039.333106558
P(32.000000000) = 1040.622052891
P(33.000000000) = 1041.912262406
P(34.000000000) = 1043.203735509
P(35.000000000) = 1044.496472604
P(36.000000000) = 1045.790474093
P(37.000000000) = 1047.085740375
P(38.000000000) = 1048.382271849
P(39.000000000) = 1049.680068910
P(40.000000000) = 1050.979131951
P(41.000000000) = 1052.279461365
P(42.000000000) = 1053.581057541
P(43.000000000) = 1054.883920866
P(44.000000000) = 1056.188051726
P(45.000000000) = 1057.493450503
P(46.000000000) = 1058.800117579
P(47.000000000) = 1060.108053334
P(48.000000000) = 1061.417258142
P(49.000000000) = 1062.727732381
P(50.000000000) = 1064.039476421
```

E/P(t)= $\frac{6000}{1+5e^{-0.0015t}}$; P(50)=1064.0718

Conclusion: the answer of e is quite close to that of d.

F/

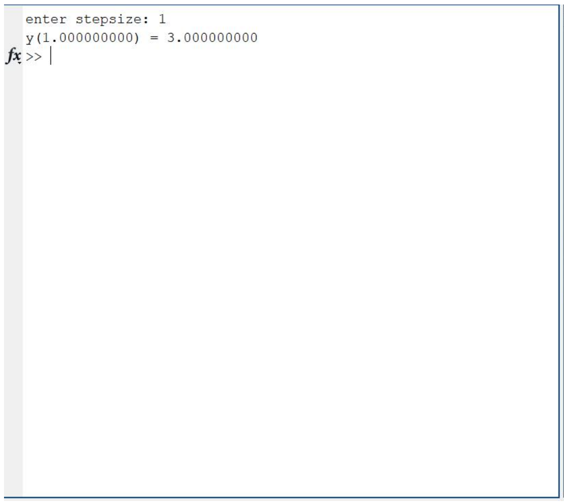
```
hold on
F = @(t,P)(0.0015*P.*(1-P./6000));
[t,P1] = ode45(F,[0 5000],1000);
plot(t,P1);
Pt=6000./(1+5.*exp(-0.0015.*t));
plot(t,Pt)
```


6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

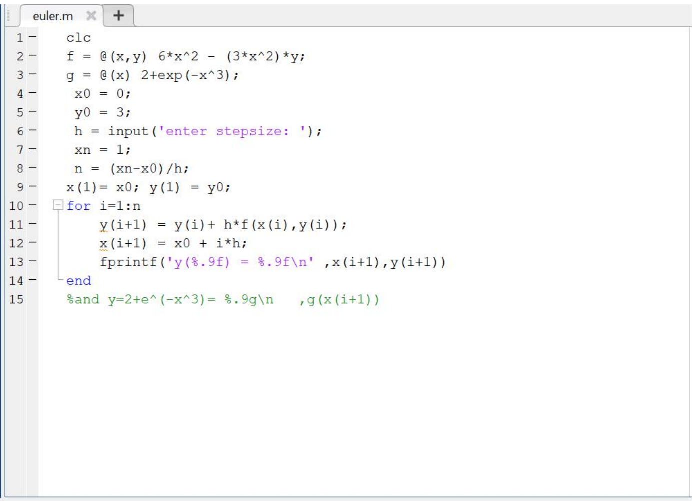
## Section 9.2, Exercise 25: Nguyễn Đăng Khoa

A/

Stepsize h=1

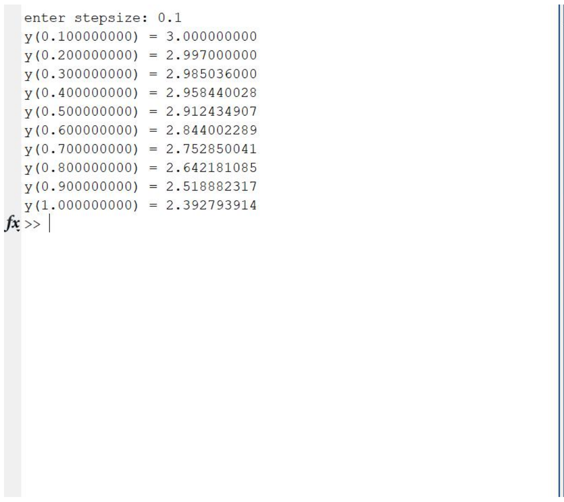


```
enter stepsize: 1
y(1.000000000) = 3.000000000
fx >> |
```

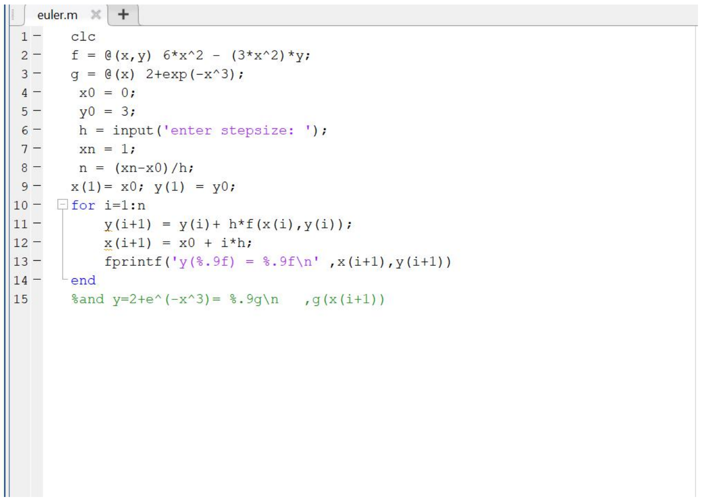


```
clc
f = @(x,y) 6*x^2 - (3*x^2)*y;
g = @(x) 2+exp(-x^3);
 x0 = 0;
 y0 = 3;
 h = input('enter stepsize: ');
 xn = 1;
 n = (xn-x0)/h;
x(1)= x0; y(1) = y0;
for i=1:n
    y(i+1) = y(i)+ h*f(x(i),y(i));
    x(i+1) = x0 + i*h;
    fprintf('y(%.9f) = %.9f\n' ,x(i+1),y(i+1))
end
%and y=2+e^(-x^3)= %.9g\n   ,g(x(i+1))
```

Stepsize h=0.1

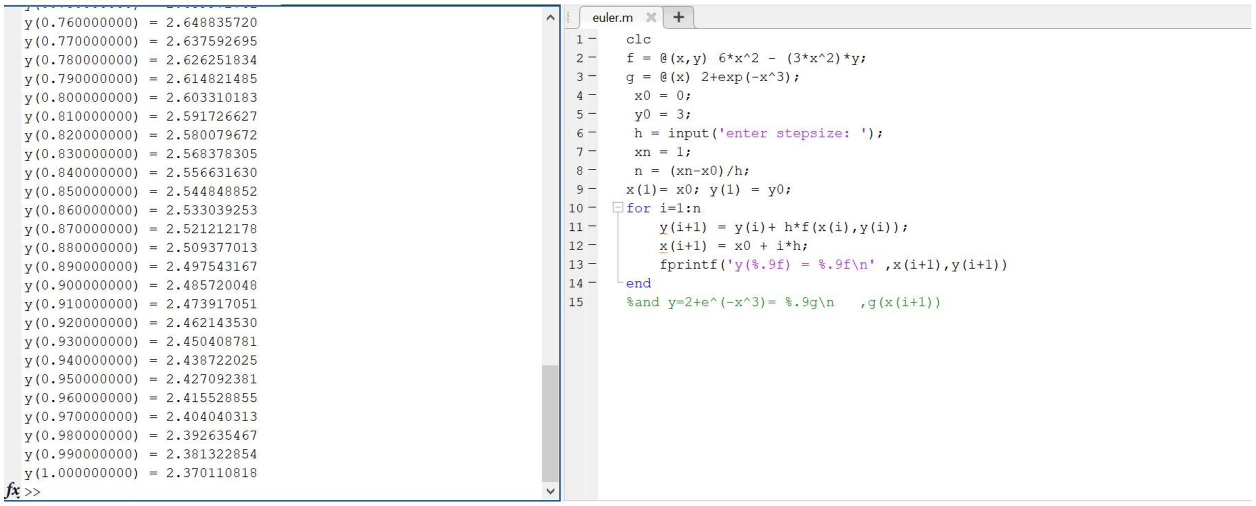


```
enter stepsize: 0.1
y(0.100000000) = 3.000000000
y(0.200000000) = 2.997000000
y(0.300000000) = 2.985036000
y(0.400000000) = 2.958440028
y(0.500000000) = 2.912434907
y(0.600000000) = 2.844002289
y(0.700000000) = 2.752850041
y(0.800000000) = 2.642181085
y(0.900000000) = 2.518882317
y(1.000000000) = 2.392793914
fx >> |
```



```
clc
f = @(x,y) 6*x^2 - (3*x^2)*y;
g = @(x) 2+exp(-x^3);
 x0 = 0;
 y0 = 3;
 h = input('enter stepsize: ');
 xn = 1;
 n = (xn-x0)/h;
x(1)= x0; y(1) = y0;
for i=1:n
    y(i+1) = y(i)+ h*f(x(i),y(i));
    x(i+1) = x0 + i*h;
    fprintf('y(%.9f) = %.9f\n' ,x(i+1),y(i+1))
end
%and y=2+e^(-x^3)= %.9g\n   ,g(x(i+1))
```

## Stepsize h=0.01

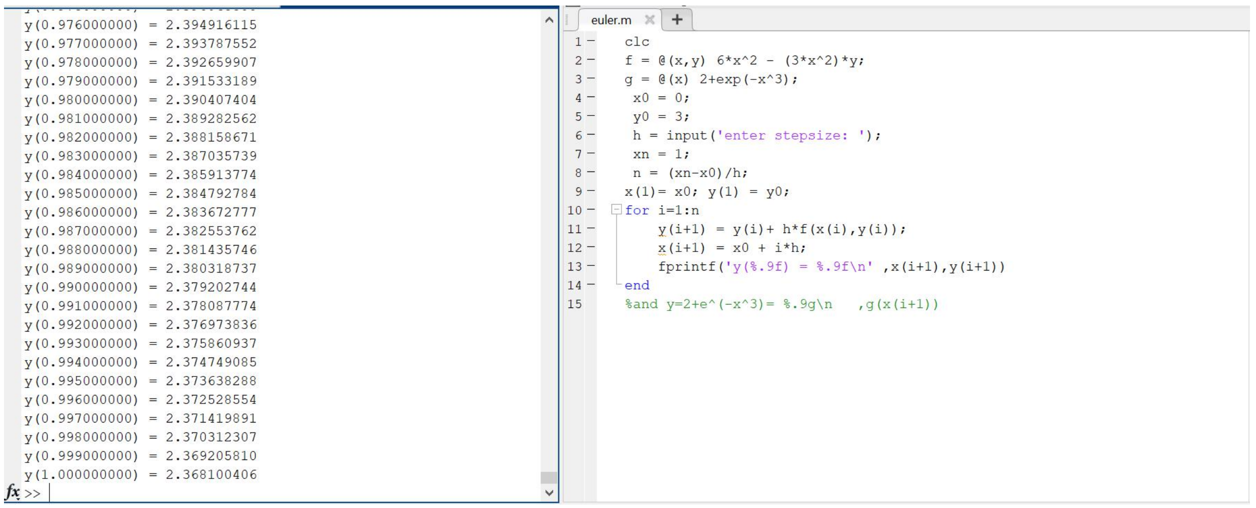


```
y(0.760000000) = 2.648835720
y(0.770000000) = 2.637592695
y(0.780000000) = 2.626251834
y(0.790000000) = 2.614821485
y(0.800000000) = 2.603310183
y(0.810000000) = 2.591726627
y(0.820000000) = 2.580079672
y(0.830000000) = 2.568378305
y(0.840000000) = 2.556631630
y(0.850000000) = 2.544848852
y(0.860000000) = 2.533039253
y(0.870000000) = 2.521212178
y(0.880000000) = 2.509377013
y(0.890000000) = 2.497543167
y(0.900000000) = 2.485720048
y(0.910000000) = 2.473917051
y(0.920000000) = 2.462143530
y(0.930000000) = 2.450408781
y(0.940000000) = 2.438722025
y(0.950000000) = 2.427092381
y(0.960000000) = 2.415528855
y(0.970000000) = 2.404040313
y(0.980000000) = 2.392635467
y(0.990000000) = 2.381322854
y(1.000000000) = 2.370110818
fx >>
```

euler.m

```
clc
f = @(x,y) 6*x^2 - (3*x^2)*y;
g = @(x) 2+exp(-x^3);
 x0 = 0;
 y0 = 3;
 h = input('enter stepsize: ');
 xn = 1;
 n = (xn-x0)/h;
x(1)= x0; y(1) = y0;
for i=1:n
    y(i+1) = y(i)+ h*f(x(i),y(i));
    x(i+1) = x0 + i*h;
    fprintf('y(%.9f) = %.9f\n' ,x(i+1),y(i+1))
end
%and y=2+e^(-x^3)= %.9g\n   ,g(x(i+1))
```

## Stepsize h=0.001

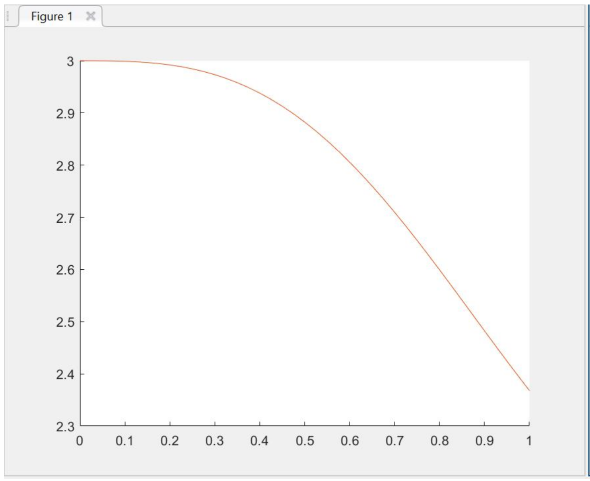


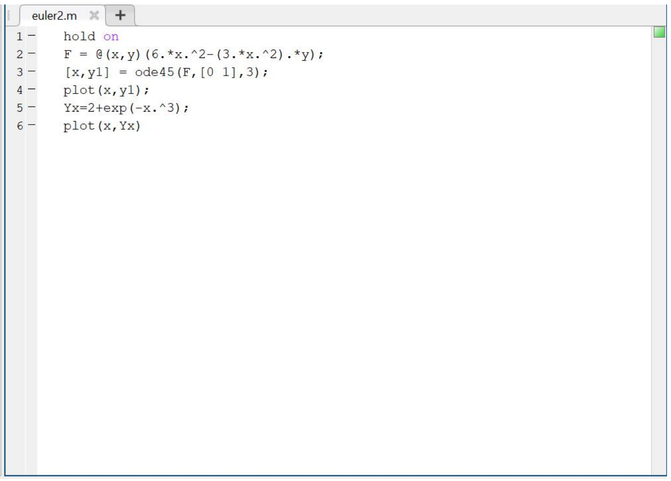
```
y(0.976000000) = 2.394916115
y(0.977000000) = 2.393787552
y(0.978000000) = 2.392659907
y(0.979000000) = 2.391533189
y(0.980000000) = 2.390407404
y(0.981000000) = 2.389282562
y(0.982000000) = 2.388158671
y(0.983000000) = 2.387035739
y(0.984000000) = 2.385913774
y(0.985000000) = 2.384792784
y(0.986000000) = 2.383672777
y(0.987000000) = 2.382553762
y(0.988000000) = 2.381435746
y(0.989000000) = 2.380318737
y(0.990000000) = 2.379202744
y(0.991000000) = 2.378087774
y(0.992000000) = 2.376973836
y(0.993000000) = 2.375860937
y(0.994000000) = 2.374749085
y(0.995000000) = 2.373638288
y(0.996000000) = 2.372528554
y(0.997000000) = 2.371419891
y(0.998000000) = 2.370312307
y(0.999000000) = 2.369205810
y(1.000000000) = 2.368100406
fx >>
```

euler.m

```
clc
f = @(x,y) 6*x^2 - (3*x^2)*y;
g = @(x) 2+exp(-x^3);
 x0 = 0;
 y0 = 3;
 h = input('enter stepsize: ');
 xn = 1;
 n = (xn-x0)/h;
x(1)= x0; y(1) = y0;
for i=1:n
    y(i+1) = y(i)+ h*f(x(i),y(i));
    x(i+1) = x0 + i*h;
    fprintf('y(%.9f) = %.9f\n' ,x(i+1),y(i+1))
end
%and y=2+e^(-x^3)= %.9g\n   ,g(x(i+1))
```

B/

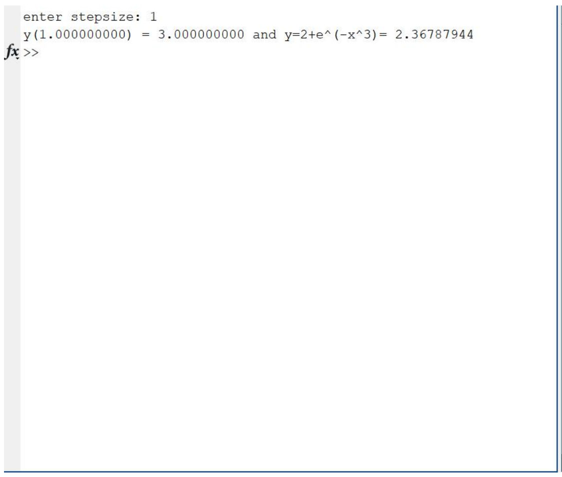




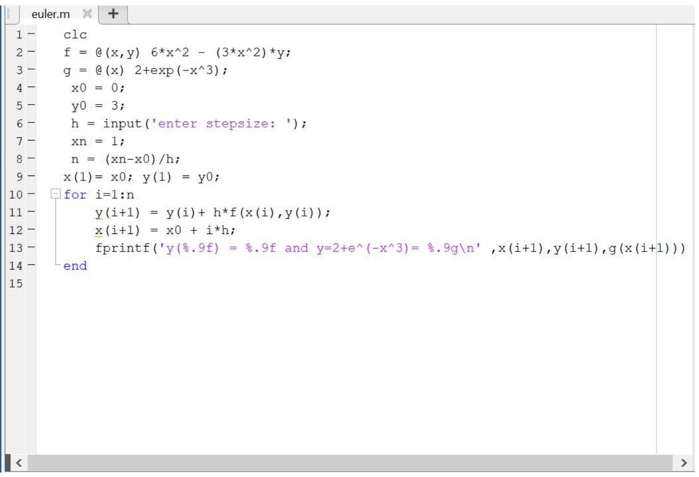
```
hold on
F = @(x,y)(6.*x.^2-(3.*x.^2).*y);
[x,y1] = ode45(F,[0 1],3);
plot(x,y1);
Yx=2+exp(-x.^3);
plot(x,Yx)
```

c/

## Stepsize h=1

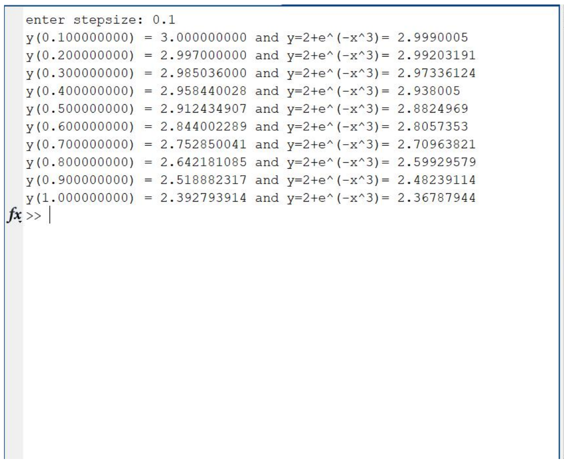


```
enter stepsize: 1
y(1.000000000) = 3.000000000 and y=2+e^(-x^3)= 2.36787944
fx >>
```

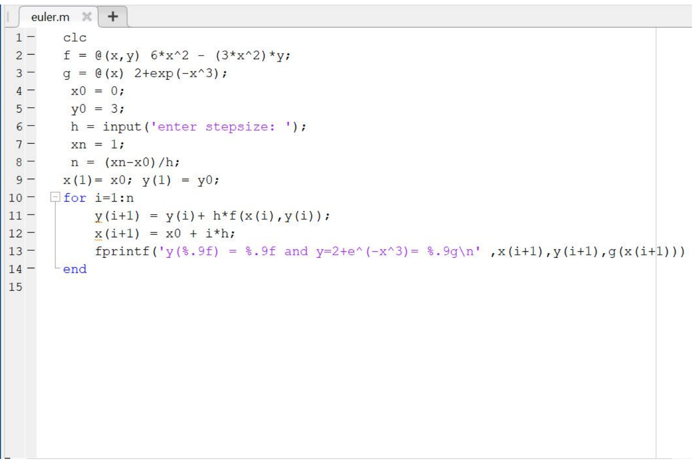


```
 clc
 f = @(x,y) 6*x^2 - (3*x^2)*y;
 g = @(x) 2+exp(-x^3);
  x0 = 0;
  y0 = 3;
  h = input('enter stepsize: ');
  xn = 1;
  n = (xn-x0)/h;
 x(1)= x0; y(1) = y0;
 for i=1:n
     y(i+1) = y(i)+ h*f(x(i),y(i));
     x(i+1) = x0 + i*h;
     fprintf('y(%.9f) = %.9f and y=2+e^(-x^3)= %.9g\n' ,x(i+1),y(i+1),g(x(i+1)))
 end
```

## Stepsize h=0.1

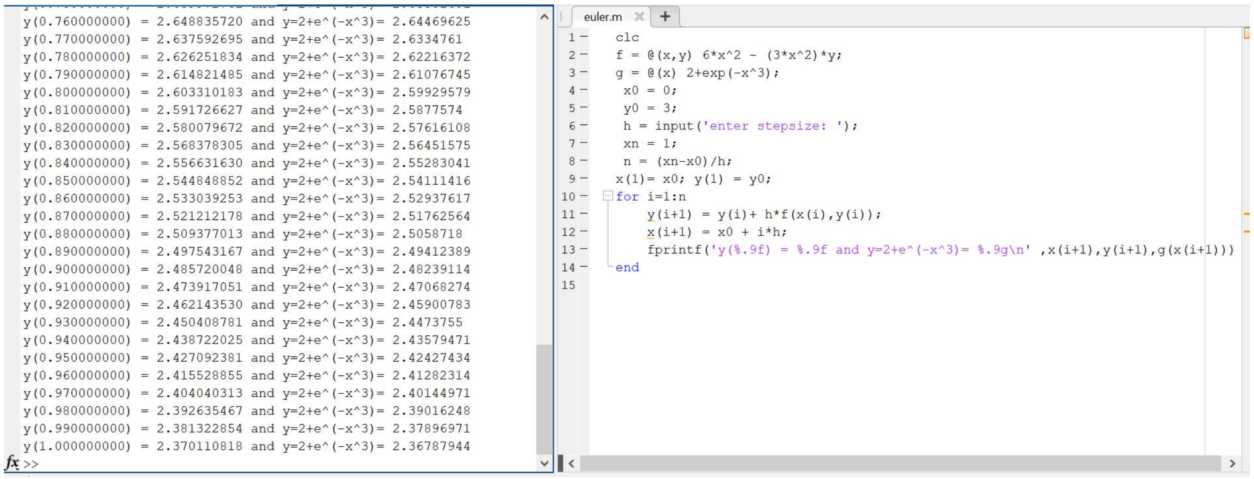


```
enter stepsize: 0.1
y(0.100000000) = 3.000000000 and y=2+e^(-x^3)= 2.9990005
y(0.200000000) = 2.997000000 and y=2+e^(-x^3)= 2.99203191
y(0.300000000) = 2.985036000 and y=2+e^(-x^3)= 2.97336124
y(0.400000000) = 2.958440028 and y=2+e^(-x^3)= 2.938005
y(0.500000000) = 2.912434907 and y=2+e^(-x^3)= 2.8824969
y(0.600000000) = 2.844002289 and y=2+e^(-x^3)= 2.8057353
y(0.700000000) = 2.752850041 and y=2+e^(-x^3)= 2.70963821
y(0.800000000) = 2.642181085 and y=2+e^(-x^3)= 2.59929579
y(0.900000000) = 2.518882317 and y=2+e^(-x^3)= 2.48239114
y(1.000000000) = 2.392793914 and y=2+e^(-x^3)= 2.36787944
fx >>
```



```
 clc
 f = @(x,y) 6*x^2 - (3*x^2)*y;
 g = @(x) 2+exp(-x^3);
  x0 = 0;
  y0 = 3;
  h = input('enter stepsize: ');
  xn = 1;
  n = (xn-x0)/h;
 x(1)= x0; y(1) = y0;
 for i=1:n
     y(i+1) = y(i)+ h*f(x(i),y(i));
     x(i+1) = x0 + i*h;
     fprintf('y(%.9f) = %.9f and y=2+e^(-x^3)= %.9g\n' ,x(i+1),y(i+1),g(x(i+1)))
 end
```

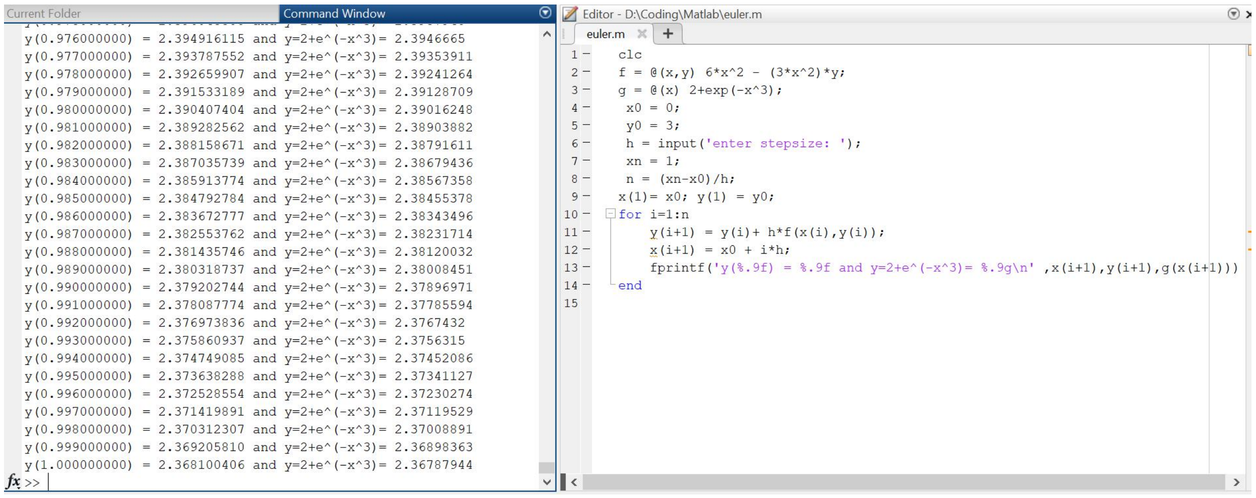
Stepsize h=0.01



```
y(0.760000000) = 2.648835720 and y=2+e^(-x^3)= 2.64469625
y(0.770000000) = 2.637592695 and y=2+e^(-x^3)= 2.6334761
y(0.780000000) = 2.626251834 and y=2+e^(-x^3)= 2.62216372
y(0.790000000) = 2.614821485 and y=2+e^(-x^3)= 2.61076745
y(0.800000000) = 2.603310183 and y=2+e^(-x^3)= 2.59929579
y(0.810000000) = 2.591726627 and y=2+e^(-x^3)= 2.5877574
y(0.820000000) = 2.580079672 and y=2+e^(-x^3)= 2.57616108
y(0.830000000) = 2.568378305 and y=2+e^(-x^3)= 2.56451575
y(0.840000000) = 2.556631630 and y=2+e^(-x^3)= 2.55283041
y(0.850000000) = 2.544848852 and y=2+e^(-x^3)= 2.54111416
y(0.860000000) = 2.533039253 and y=2+e^(-x^3)= 2.52937617
y(0.870000000) = 2.521212178 and y=2+e^(-x^3)= 2.51762564
y(0.880000000) = 2.509377013 and y=2+e^(-x^3)= 2.5058718
y(0.890000000) = 2.497543167 and y=2+e^(-x^3)= 2.49412389
y(0.900000000) = 2.485720048 and y=2+e^(-x^3)= 2.48239114
y(0.910000000) = 2.473917051 and y=2+e^(-x^3)= 2.47068274
y(0.920000000) = 2.462143530 and y=2+e^(-x^3)= 2.45900783
y(0.930000000) = 2.450408781 and y=2+e^(-x^3)= 2.4473755
y(0.940000000) = 2.438722025 and y=2+e^(-x^3)= 2.43579471
y(0.950000000) = 2.427092381 and y=2+e^(-x^3)= 2.42427434
y(0.960000000) = 2.415528855 and y=2+e^(-x^3)= 2.41282314
y(0.970000000) = 2.404040313 and y=2+e^(-x^3)= 2.40144971
y(0.980000000) = 2.392635467 and y=2+e^(-x^3)= 2.39016248
y(0.990000000) = 2.381322854 and y=2+e^(-x^3)= 2.37896971
y(1.000000000) = 2.370110818 and y=2+e^(-x^3)= 2.36787944
fx >>
```

```matlab
clc
f = @(x,y) 6*x^2 - (3*x^2)*y;
g = @(x) 2+exp(-x^3);
 x0 = 0;
 y0 = 3;
 h = input('enter stepsize: ');
 xn = 1;
 n = (xn-x0)/h;
x(1)= x0; y(1) = y0;
for i=1:n
    y(i+1) = y(i)+ h*f(x(i),y(i));
    x(i+1) = x0 + i*h;
    fprintf('y(%.9f) = %.9f and y=2+e^(-x^3)= %.9g\n' ,x(i+1),y(i+1),g(x(i+1)))
end
```

Stepsize h=0.001



```
y(0.976000000) = 2.394916115 and y=2+e^(-x^3)= 2.3946665
y(0.977000000) = 2.393787552 and y=2+e^(-x^3)= 2.39353911
y(0.978000000) = 2.392659907 and y=2+e^(-x^3)= 2.39241264
y(0.979000000) = 2.391533189 and y=2+e^(-x^3)= 2.39128709
y(0.980000000) = 2.390407404 and y=2+e^(-x^3)= 2.39016248
y(0.981000000) = 2.389282562 and y=2+e^(-x^3)= 2.38903882
y(0.982000000) = 2.388158671 and y=2+e^(-x^3)= 2.38791611
y(0.983000000) = 2.387035739 and y=2+e^(-x^3)= 2.38679436
y(0.984000000) = 2.385913774 and y=2+e^(-x^3)= 2.38567358
y(0.985000000) = 2.384792784 and y=2+e^(-x^3)= 2.38455378
y(0.986000000) = 2.383672777 and y=2+e^(-x^3)= 2.38343496
y(0.987000000) = 2.382553762 and y=2+e^(-x^3)= 2.38231714
y(0.988000000) = 2.381435746 and y=2+e^(-x^3)= 2.38120032
y(0.989000000) = 2.380318737 and y=2+e^(-x^3)= 2.38008451
y(0.990000000) = 2.379202744 and y=2+e^(-x^3)= 2.37896971
y(0.991000000) = 2.378087774 and y=2+e^(-x^3)= 2.37785594
y(0.992000000) = 2.376973836 and y=2+e^(-x^3)= 2.3767432
y(0.993000000) = 2.375860937 and y=2+e^(-x^3)= 2.3756315
y(0.994000000) = 2.374749085 and y=2+e^(-x^3)= 2.37452086
y(0.995000000) = 2.373638288 and y=2+e^(-x^3)= 2.37341127
y(0.996000000) = 2.372528554 and y=2+e^(-x^3)= 2.37230274
y(0.997000000) = 2.371419891 and y=2+e^(-x^3)= 2.37119529
y(0.998000000) = 2.370312307 and y=2+e^(-x^3)= 2.37008891
y(0.999000000) = 2.369205810 and y=2+e^(-x^3)= 2.36898363
y(1.000000000) = 2.368100406 and y=2+e^(-x^3)= 2.36787944
fx >>
```

```matlab
clc
f = @(x,y) 6*x^2 - (3*x^2)*y;
g = @(x) 2+exp(-x^3);
 x0 = 0;
 y0 = 3;
 h = input('enter stepsize: ');
 xn = 1;
 n = (xn-x0)/h;
x(1)= x0; y(1) = y0;
for i=1:n
    y(i+1) = y(i)+ h*f(x(i),y(i));
    x(i+1) = x0 + i*h;
    fprintf('y(%.9f) = %.9f and y=2+e^(-x^3)= %.9g\n' ,x(i+1),y(i+1),g(x(i+1)))
end
```

Conclusion: the smaller the stepsize, the more precise answer it gives compared to the exact value of y(1)

*Section 9.4, Exercise 21: Nguyễn Hoàng An*

## I/ THE THEORIES RELATED TO MODELS OF POPULATION GROWTH

### 1/ THE LAW OF NATURAL GROWTH

In general, if P(t) is the value of a quantity at time and if the rate of change of with respect to is proportional to its size P(t) at any time, then

(1) $$\frac{dP}{dt} = kP$$

The solution of the initial-value problem $\frac{dP}{dt} = kP \qquad P(0) = P_0$ is

(2) $$P(t) = P_0 e^{kt}$$

We can account for emigration (or "harvesting") from a population by modifying Equation (1): If the rate of emigration is a constant, then the rate of change of the population is modeled by the differential equation

(3) $$\frac{dP}{dt} = kP - m$$

### 2/ THE LOGISTIC MODEL

A population often increases exponentially in its early stages but levels off eventually and approaches its carrying capacity because of limited resources. If is the size of the population at time t, we assume that

$$\frac{dP}{dt} \approx kP - m \text{ if P is small}$$

This says that the growth rate is initially close to being proportional to size. In other words, the relative growth rate is almost constant when the population is small. But we also want to reflect the fact that the relative growth rate decreases as **the population P increases and becomes negative if P ever exceeds its carrying capacity M**, the maximum population that the environment is capable of sustaining in the long run. The simplest expression for the relative growth rate that incorporates these assumptions is

$$\frac{1}{P}\frac{dP}{dt}=k(1-\frac{P}{M})$$

Multiplying by P, we obtain the model for population growth known as the logistic differential equation:

(4) $$\boxed{\frac{dP}{dt}=kP(1-\frac{P}{M})}$$

The logistic equation (4) is separable and so we can solve it explicitly using the method of Section 9.3 Separable Equation. Since

(5) $$\boxed{\int\frac{dP}{P(1-\frac{P}{M})}=\int kdt}$$

Rewrite Equation (5), we have:

$$\int(\frac{1}{P}+\frac{1}{M-P})dP=\int kdt$$

$$\ln|P|-\ln|M-P|=kt+C$$

$$\ln\left|\frac{M-P}{P}\right|=-kt-C$$

$$\left|\frac{M-P}{P}\right|=e^{-kt-C}=e^{-C}e^{-kt}$$

(6) $\frac{M-P}{P}=Ae^{-kt}$ where $A=\pm e^{-C}$

Solving Equation 6 for P, we get

$$\frac{M}{P}-1=Ae^{-kt}$$

So $P = \dfrac{M}{1 + Ae^{-kt}}$

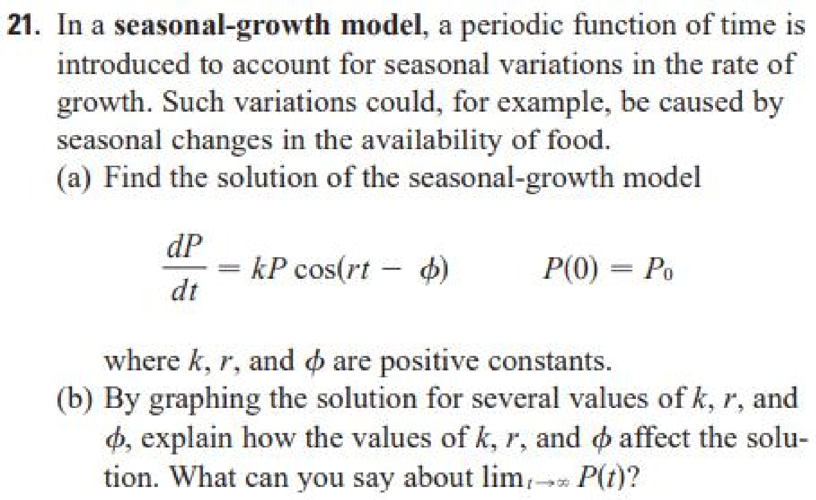
We find the value of A by putting t=0 in Equation 6. If t=0, then $P = P_0$ (the initial population), so $\dfrac{M - P_0}{P_0} = Ae^0 = A$

Thus the solution to the logistic equation is

(7) $$\boxed{P(t) = \frac{M}{1 + Ae^{-kt}} \quad \text{where} \quad A = \frac{M - P_0}{P_0}}$$

II/

Problem



**21.** In a **seasonal-growth model**, a periodic function of time is introduced to account for seasonal variations in the rate of growth. Such variations could, for example, be caused by seasonal changes in the availability of food.

(a) Find the solution of the seasonal-growth model

$$\frac{dP}{dt} = kP\cos(rt - \phi) \qquad P(0) = P_0$$

where $k$, $r$, and $\phi$ are positive constants.

(b) By graphing the solution for several values of $k$, $r$, and $\phi$, explain how the values of $k$, $r$, and $\phi$ affect the solution. What can you say about $\lim_{t\to\infty} P(t)$?

a/ The
solution of the seasonal-growth model by detailed solution

$$\frac{dP}{dt} = kP\cos(rt - \phi)$$

Rewrite equation, we have:

$$P'(t) = kP(t)\cos(rt - \phi)$$

Dividing both sides by P:

$$\frac{1}{P}dP = k\cos(rt - \phi)dt$$

$\Leftrightarrow \ln(P) = \int k\cos(rt-\phi)$

$\Leftrightarrow \ln(P) = \frac{k}{r}\sin(rt-\phi) + C$

$\Leftrightarrow P = e^{\frac{k}{r}\sin(rt-\phi)+C}$

$\Leftrightarrow P(0) = e^{\frac{k}{r}\sin(-\phi)+C} = P_0$

$\Leftrightarrow \ln(P_0) = \frac{k}{r}\sin(-\phi) + C$

$\Leftrightarrow C = \ln(P_0) - \frac{k}{r}\sin(-\phi)$

$\Rightarrow \boxed{P = e^{\frac{k}{r}\sin(rt-\phi)+\ln(P_0)-\frac{k}{r}\sin(-\phi)}}$

## Application of my topic in general

- In bringing together experimental and theoretical studies, the importance of long-term experimentation and the need for ecology to have model systems, and the value of population growth rate as a means of understanding and predicting population change.

- Some examples related to population growth model:

- Demographic history
- Demographic transition
- Density dependence
- Epidemiological transition
- Irruptive growth
- List of countries by population growth rate
- Population dynamics
- World population
- World population estimates

## Application of my topic in my question

Methods used to estimate the available value, to know whether a rising or falling variable has any effect on the overall trend.

## III/ CODE IN SCILAB

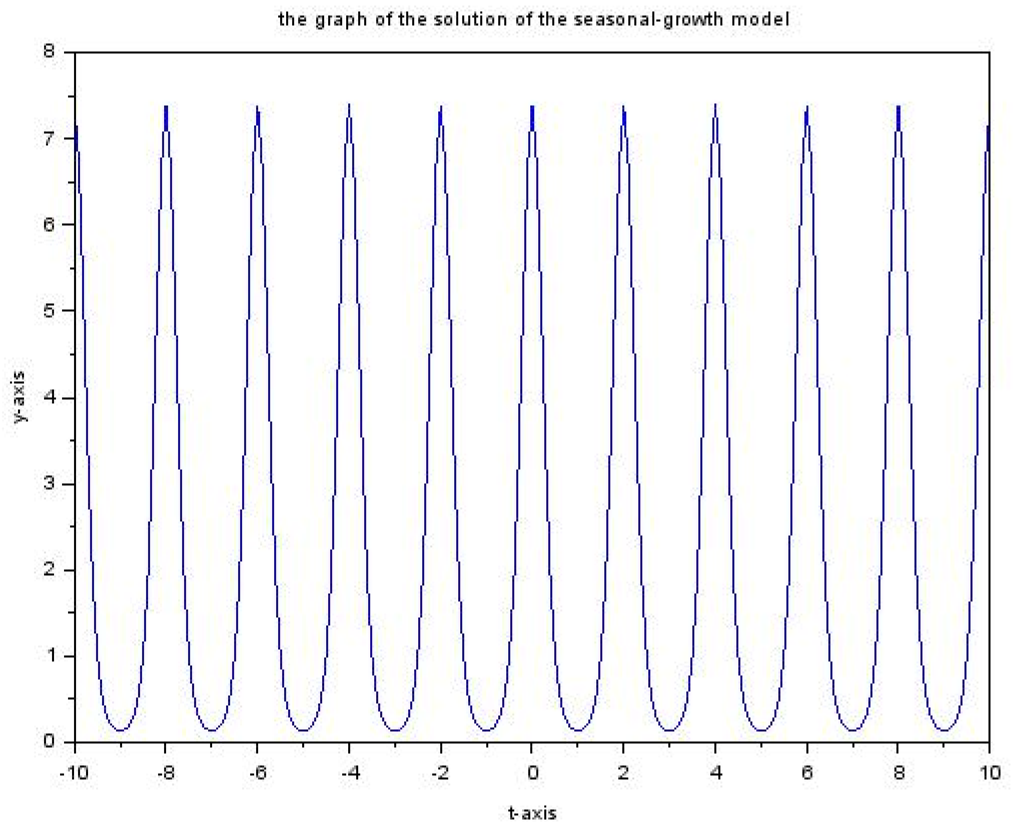
### b/ Graphing the solution

*We consider C=0 and suppose that Q = $\phi$ in scilab*

. $P(t) = e^{\frac{k}{r}\sin(rt-\phi)}$ when $k = 2\pi$ ; $r = \pi$ ; $\phi = \frac{3\pi}{2}$

Scilab code

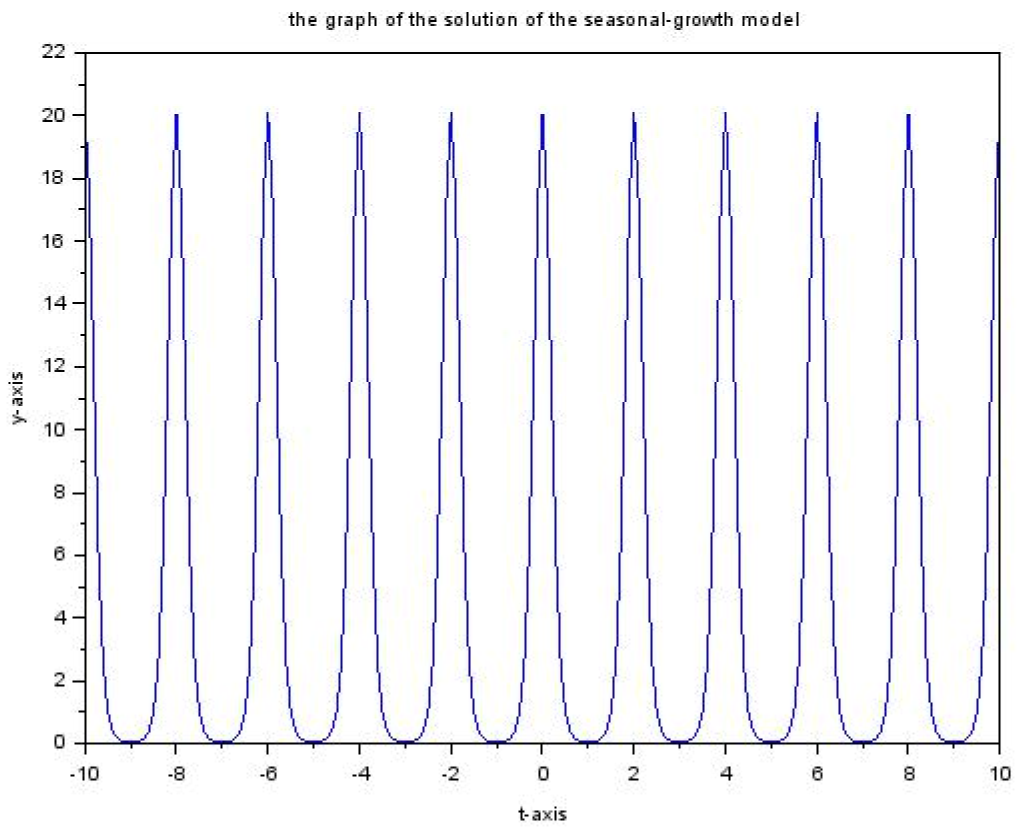
```
t=[-10:0.1:10];
k= 2*%pi;
r= %pi;
Q= 3*%pi/2;
y=%e^((k/r)*sin(r*t-Q))
plot(t,y)
xlabel('t-axis');
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```



. $P(t)=e^{\frac{k}{r}\sin(rt-\phi)}$ when $k=2\pi+\pi=3\pi$ ; $r=\pi$ ; $\phi=\frac{3\pi}{2}$

Scilab code

```
t=[-10:0.1:10];
k= 3*%pi;
r= %pi;
Q= 3*%pi/2;
y=%e^((k/r)*sin(r*t-Q))
plot(t,y)
xlabel('t-axis');
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```

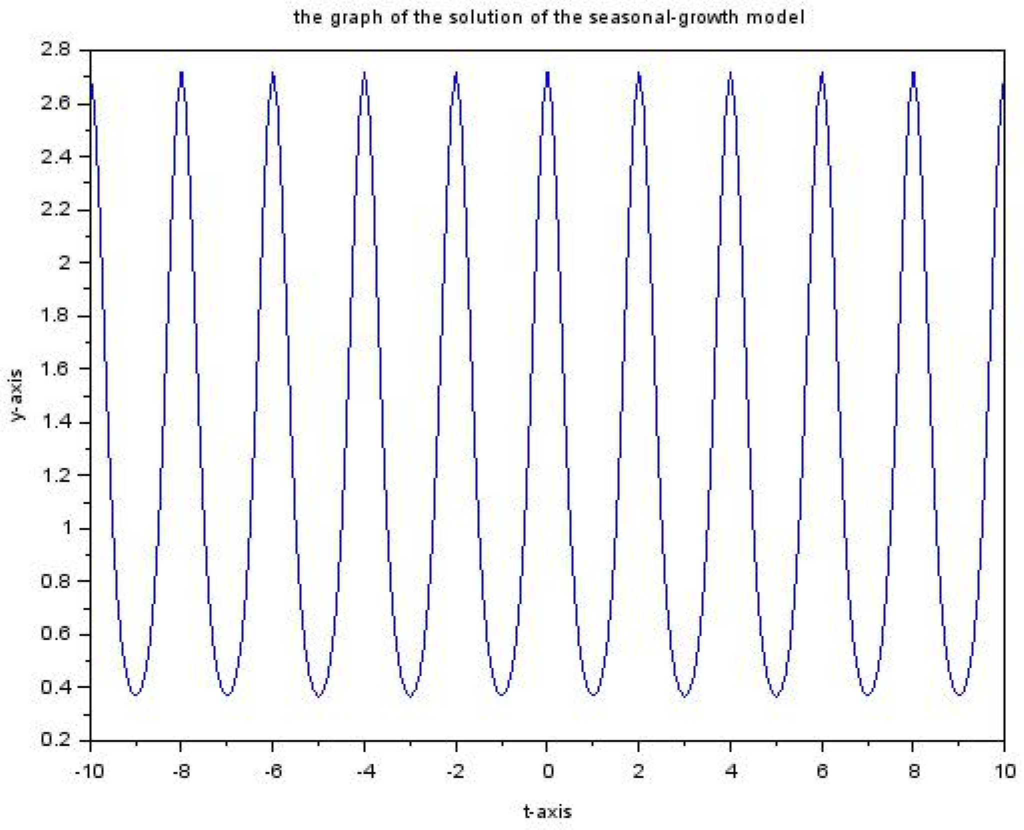


. $P(t) = e^{\frac{k}{r}\sin(rt-\phi)}$ when $k = 2\pi - \pi = \pi$ ; $r = \pi$ ; $\phi = \frac{3\pi}{2}$

Scilab code

```
t=[-10:0.1:10];
k= %pi;
r= %pi;
Q= 3*%pi/2;
y=%e^((k/r)*sin(r*t-Q))
plot(t,y)
xlabel('t-axis');
```

```
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```



*Comparison between 3 values k equals* $1\pi;2\pi;3\pi$ *. The size of the range of the function depend on magnitude of k*

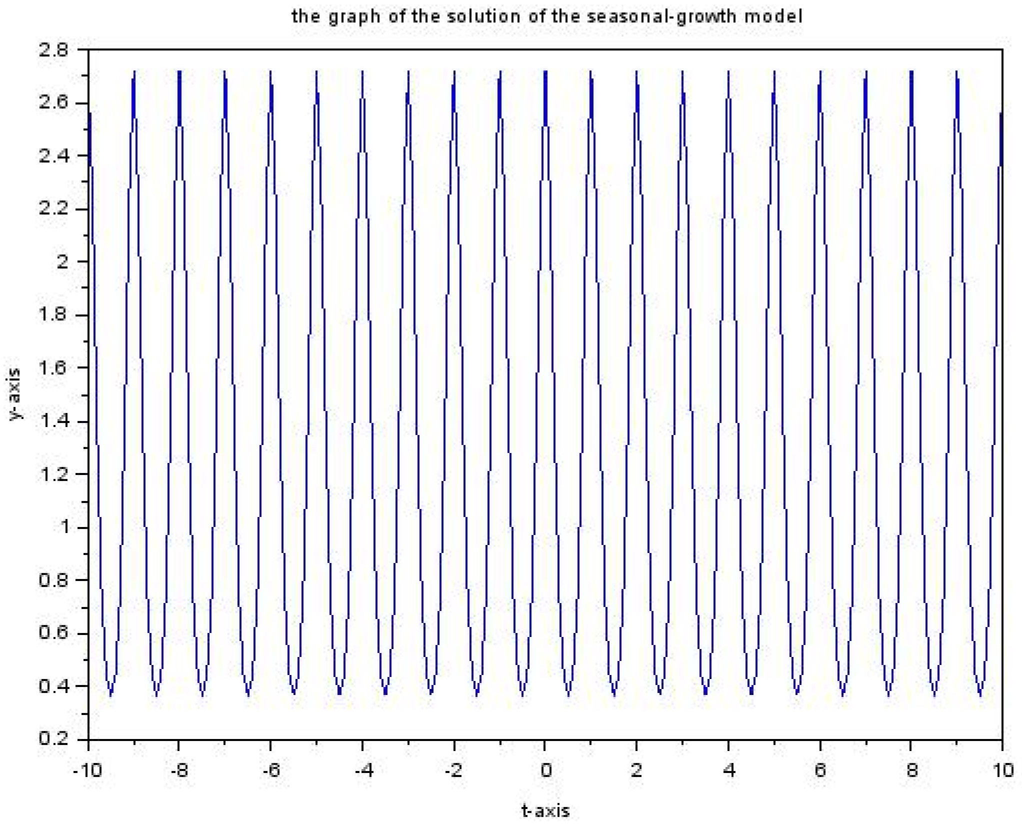
*Conclusion: k is proportional to the size of the range of the function*

. $P(t)=e^{\frac{k}{r}\sin(rt-\phi)}$ when $k=2\pi$ ; $r=\pi+\pi=2\pi$ ; $\phi=\dfrac{3\pi}{2}$

Scilab code

```
t=[-10:0.1:10];
k= 2*%pi;
r= 2*%pi;
Q= 3*%pi/2;
y=%e^((k/r)*sin(r*t-Q))
```

```
plot(t,y)
xlabel('t-axis');
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```

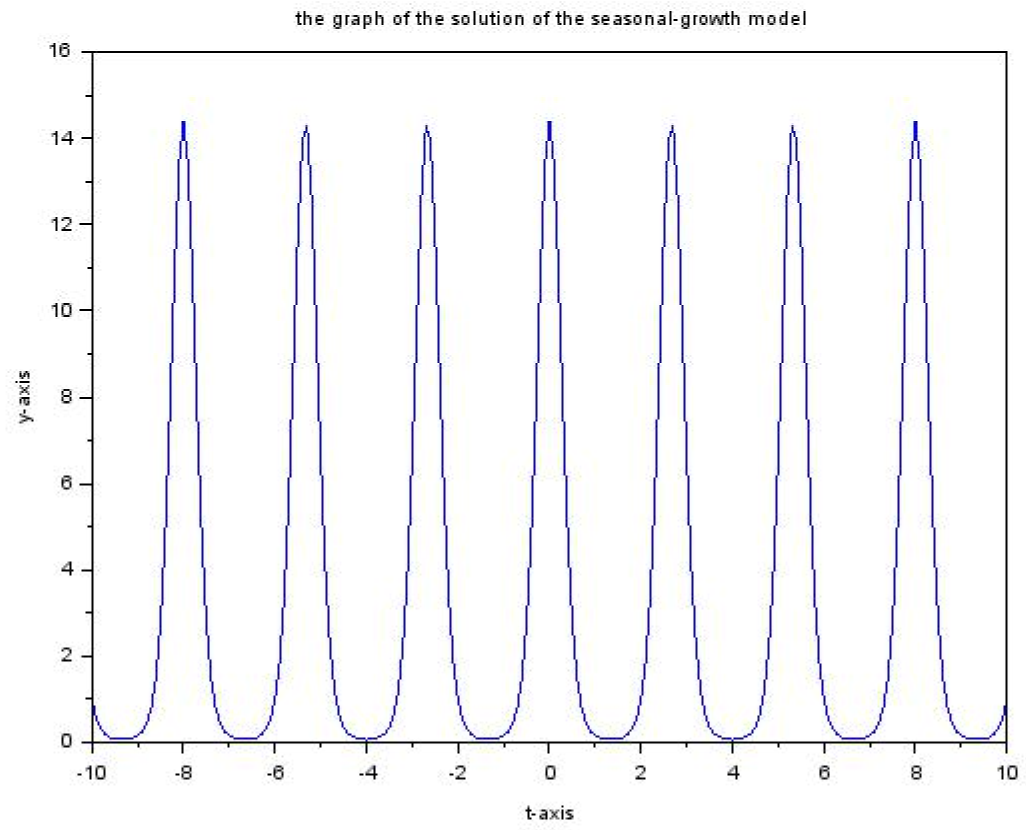


. $P(t) = e^{\frac{k}{r}\sin(rt-\phi)}$ when $k = 2\pi$ ; $r = \pi - \frac{\pi}{4} = \frac{3\pi}{4}$ ; $\phi = \frac{3\pi}{2}$

Scilab code

```
t=[-10:0.1:10];
k= 2*%pi;
r= 3*%pi/4;
```

```
Q= 3*%pi/2;
y=%e^((k/r)*sin(r*t-Q))
plot(t,y)
xlabel('t-axis');
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```



*Comparison between 3 values r equals* $\frac{3\pi}{4};\pi;2\pi$. *When r increases then the size of the range of the function decreases.*

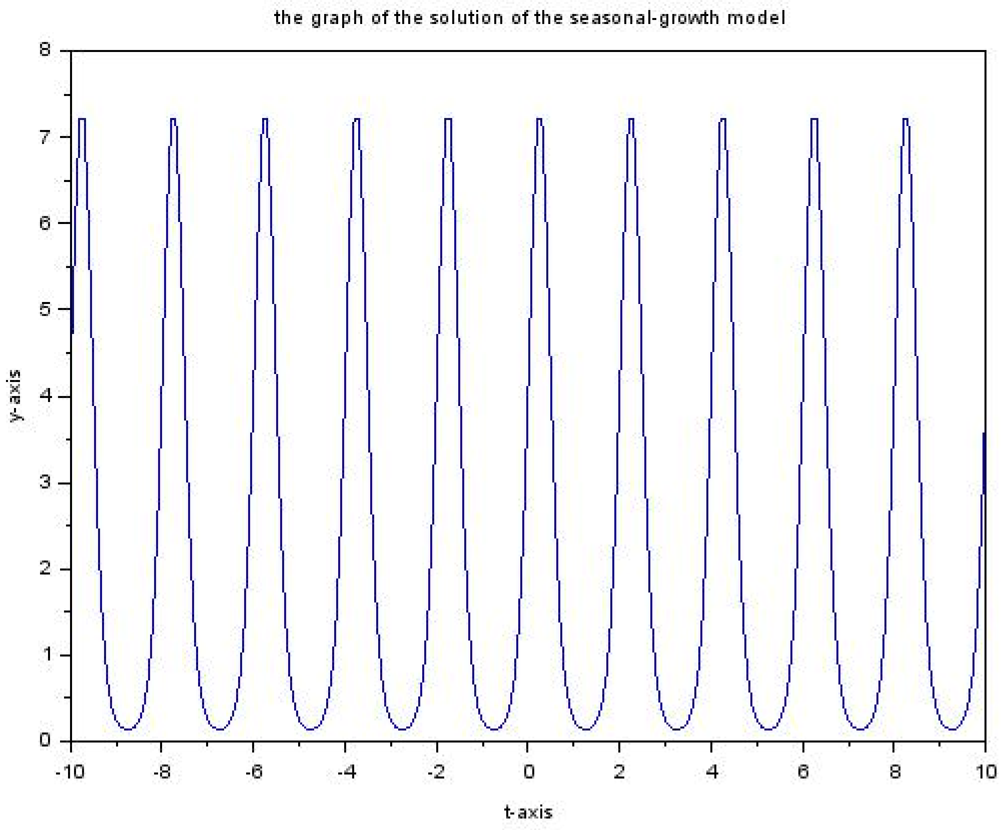
*Conclusion: r is inversely proportional to the size of the range of the function and proportional to the frequency of the function*

. $P(t)=e^{\frac{k}{r}\sin(rt-\phi)}$ when $k=2\pi$ ; $r=\pi$ ; $\phi=\frac{3\pi}{2}+\frac{\pi}{4}=\frac{7\pi}{4}$

Scilab Code

```
t=[-10:0.1:10];
```

```
k= 2*%pi;
r= %pi;
Q= 7*%pi/4;
y=%e^((k/r)*sin(r*t-Q))
plot(t,y)
xlabel('t-axis');
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```

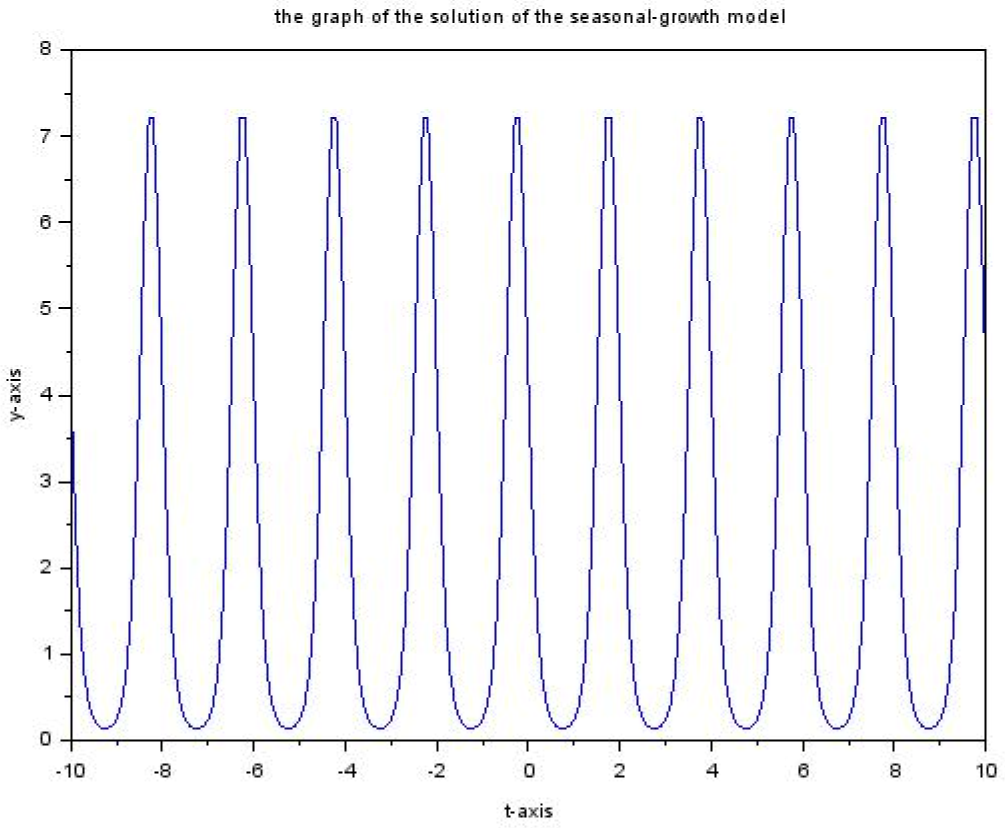


-

. $P(t) = e^{\frac{k}{r}\sin(rt-\phi)}$ when $k = 2\pi$ ; $r = \pi$ ; $\phi = \frac{3\pi}{2} - \frac{\pi}{4} = \frac{5\pi}{4}$

Scilab Code

```
t=[-10:0.1:10];
k= 2*%pi;
r= %pi;
Q= 5*%pi/4;
y=%e^((k/r)*sin(r*t-Q))
plot(t,y)
xlabel('t-axis');
ylabel('y-axis');
title('the graph of the solution of the seasonal-growth model');
```



*Conclusion: When φ increases the function moves to the right and moves to the left when φ decreases.*

$\lim_{t\to\infty} P(t) = DNE$ Because $\lim_{t\to\infty} \sin(rt-\phi) = DNE$

Explain the code

Code t=[a:h:b] means that I assign the value t from -10 to 10 with step size h equal 0.1

The next one, I assign several values k,r, $\phi$ equals ( x*%pi) , ( y*%pi), ( z*%pi)Then I code the function y(t)= $P(t) = e^{\frac{k}{r}\sin(rt-\phi)}$

With code plot(t, y), I have the graph of y(t)

I use xlabel and ylabel to set the name the horizontal and vertical axis are t-axis and y-axis

Title code to set the name of the function y(t).

## Remark the code

With the solution by code, I could apply for the same exercises with the substitution of inputs to get the results quickly and accurately.

## Comparison between the detailed solution and the solution by code

The solution by code is easier than the detailed solution in some ways:

- In the detailed solution, you should write down values of t with step size equal 0.1 after you assign values k,r, $\phi$ while you can write a quickly code to solve it. - I see that draw the graph by code is more detail and easier than draw by hand because after I enter the code that the graph will appear but the most important thing is that I must enter it correctly. While follow the detailed solution, I will spend a lots time to write the values and draw on x-y axis can not as detailed and accurate as code.

## Reference:

[1] Stewart James (2013). Calculus: Early Transcendentals, 7th ed.

[2] https://en.wikipedia.org/wiki/Population_growth: Wikipedia, Population growth.