ENSC 180 - Assignment 1

Shengcong Zhou
sza111@sfu.ca

1. a) Wildcard
-Acceptable

b) WILDCARD
-Acceptable. Matlab is case sensitive thus this is a different variable from a).

c) *Wildcard
-Not acceptable because a variable name may only consists of alphanumeric characters and the underscore (_). In this case the asterisk (*) is not acceptable.

d) 2Wildcard
-Not acceptable because a variable name must start with a letter.

e) Wild_card
-Acceptable.

f) Wildcard!!
-Not acceptable because of the exclamation mark (!).

g) wild_card
-Acceptable.

2.

Command Window

```
>> x = [1,2,3,4,5];
>> y = [7,8,9];
>> C1 = [x;x;x;x;x]; %5x5
>> C2 = [x;x]; %2x5
>> C3 = C2'; %5x2
>> C4 = [y;y;y]; %3x3
>> C5 = y'; %3x1
>> C6 = [x y;x y]; %2x8
>> C7 = [y y;y y;y y;y y;y y;y y;y y]; %7x6
>> whos
  Name      Size            Bytes  Class     Attributes

  C1        5x5               200  double
  C2        2x5                80  double
  C3        5x2                80  double
  C4        3x3                72  double
  C5        3x1                24  double
  C6        2x8               128  double
  C7        7x6               336  double
  x         1x5                40  double
  y         1x3                24  double
```

3.

a) -2.4493e-16
b) 2.1109
c) -1.1108
d) 4.6234

Command Window

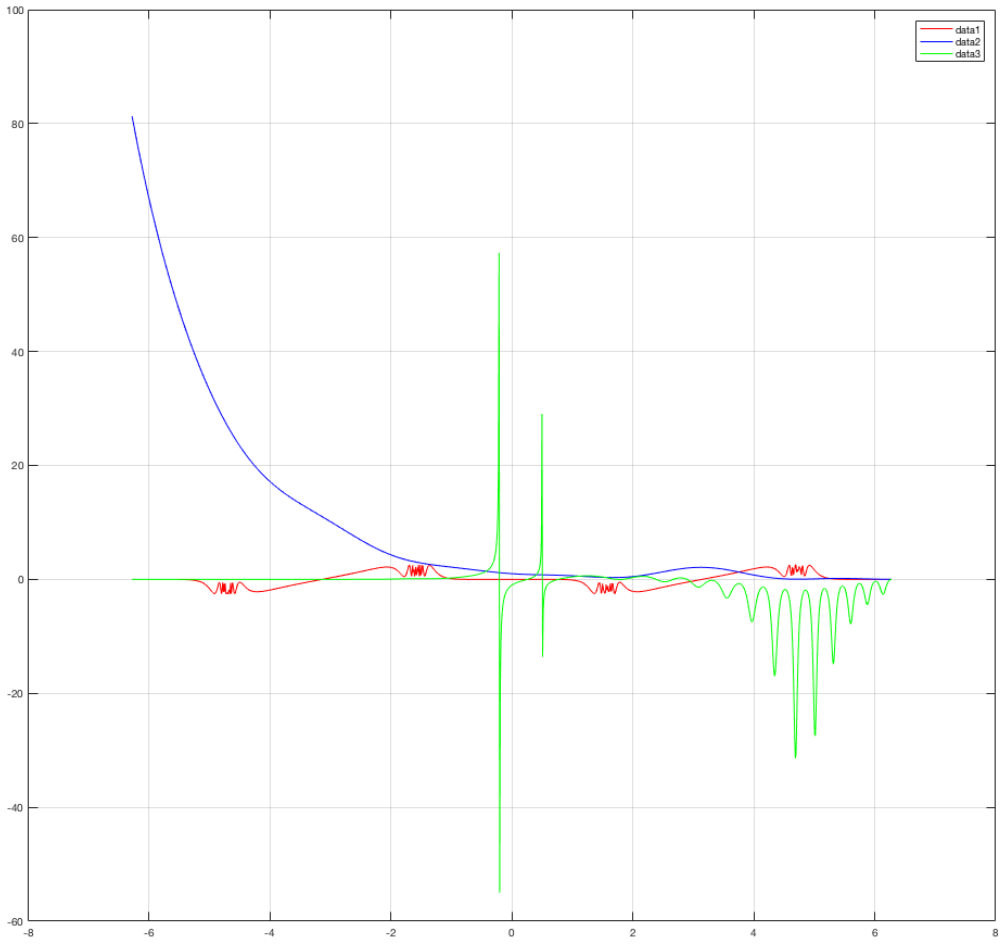
```
>> x = pi;
>> a = sin(tan(x)) - tan(sin(x))

a =

  -2.4493e-16

>> b = exp(-0.7*x) + (1 - cos(x))/(1.0 + (tan(x))^2)

b =

    2.1109

>> c = (1 + x/(x - 0.5))/(1 + (3.1*x*exp(-x) + 2)/(sin(x) - (cos(x^2))^2))

c =

   -1.1108

>> d = 3.0^0.25 + log(2.1^3.7) + atan(0.63)

d =

    4.6234
```

4.

a(x) plotted in RED
b(x) plotted in BLUE
c(x) plotted in GREEN

Command Window

```
>> x = -2*pi:0.01:2*pi;
>> a = sin(tan(x)) - tan(sin(x));
>> b = exp(-0.7.*x) + (1 - cos(x))./(1.0 + (tan(x)).^2);
>> c = (1 + x./(x - 0.5))./(1 + (3.1.*x.*exp(-x) + 2)./(sin(x) - (cos(x.^2)).^2));
>> plot(x,a,'r',x,b,'b',x,c,'g')
```



5.

size = 1x8

minimum value = 2.1

maximum value = 8.2

mean value = 5.25

median value = 5.4

standard deviation = 2

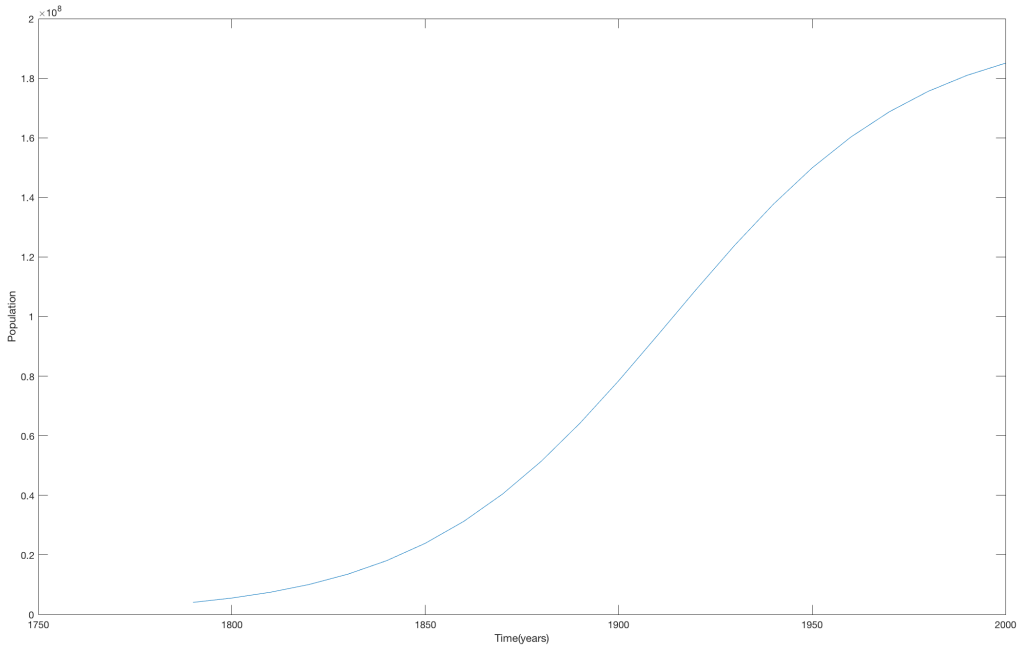
sorted array = [2.1 3.1 4.6 5.0 5.8 6.2 7.0 8.2]

```
>> array = [3.1,5.8,6.2,2.1,7.0,5.0,8.2,4.6];
>> size(array)

ans =

     1     8

>> min(array)

ans =

    2.1000

>> max(array)

ans =

    8.2000

>> mean(array)

ans =

    5.2500

>> median(array)

ans =

    5.4000

>> std(array)

ans =

    2.0000

>> sort(array)

ans =

    2.1000    3.1000    4.6000    5.0000    5.8000    6.2000    7.0000    8.2000

>>
```

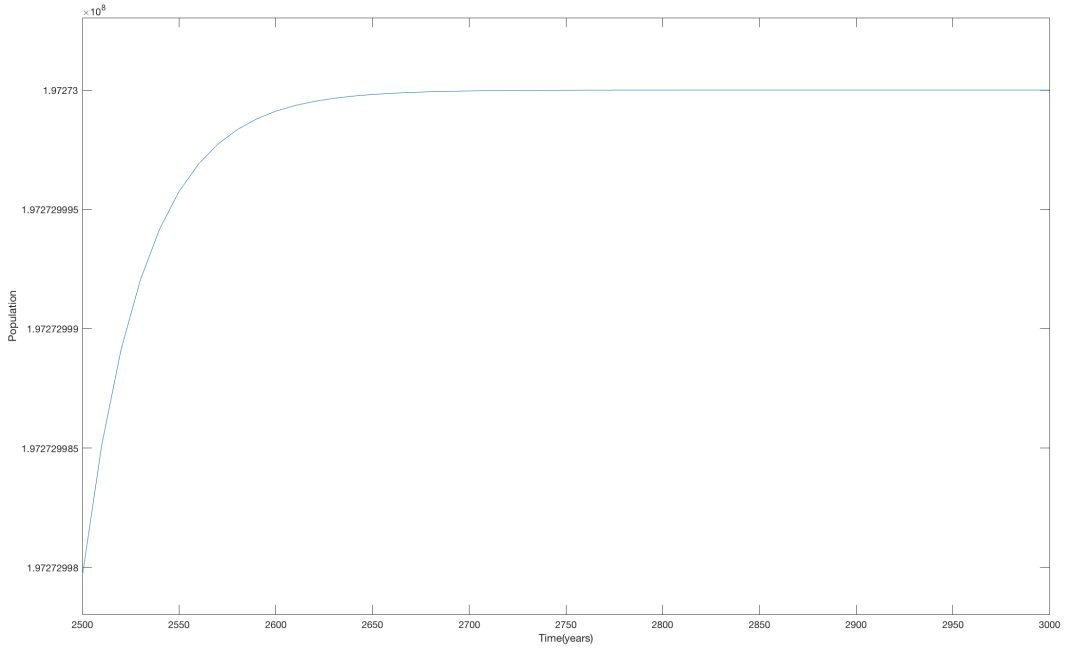
6.

Command Window

```
>> t = 1790:10:2000; %t is a variable of time in years
>> P = 197273000./(1 + exp(-0.03134*(t - 1913.25))); %Population as a function of t
>> disp(' '); disp('       t        P'); format shortG; disp([t' P']);

       t        P
    1790   4.0598e+06
    1800   5.5124e+06
    1810   7.4645e+06
    1820   1.0072e+07
    1830   1.3525e+07
    1840   1.8047e+07
    1850   2.3886e+07
    1860   3.1283e+07
    1870   4.0437e+07
    1880    5.144e+07
    1890    6.421e+07
    1900   7.8446e+07
    1910   9.3618e+07
    1920   1.0903e+08
    1930   1.2395e+08
    1940   1.3772e+08
    1950   1.4989e+08
    1960   1.6025e+08
    1970   1.6877e+08
    1980    1.756e+08
    1990   1.8095e+08
    2000   1.8507e+08
```



In order to see if the population ever reaches steady state, shift the graph to much larger values of t. If the graph seems to be a horizontal line meaning there is no change to the population occurring, the population is in steady state.



As shown in the graph over (2500<t<3000), the line becomes horizontal roughly >year 2700. Therefore it is reasonable to assume the population does reach steady state.