Spring 2018 Math 541

Homework Solutions – MatLab Programming Due Tues. 2/01/18

1. (**Each part 1 pt**) If I give you the array

```
X = linspace(0,1,5);
```

   (a) How many points are in the array?

   5

   (b) What is the spacing between the points?

   $\frac{1}{4}$

   (c) What code would you write to double the number of points in the array?

   X = linspace(0,1,10);

   (d) What code would you write to have equally spaced points in the interval $[0, 2]$, but with the same spacing between points as the original array?

   X = linspace(0,2,9);

   (e) What code would I write to turn `X` into a column vector?

   X'

   (f) What array would the code `X.^2` produce?

   0 0.0625 0.2500 0.5625 1.0000 1.5625 2.2500 3.0625 4.0000

   The code returns the square of each term.

   (g) What array would the code `X(end:-1:1)` produce?

   2.0000 1.7500 1.5000 1.2500 1.0000 0.7500 0.5000 0.2500 0

   The code returns the array with the horizontal position inverted.

   (h) Using vectorization, what code would I write to efficiently plot 2003 equally spaced points of the function $\sin(x^3 + 2x)$ over the interval $[-3.7, 4.2]$ in the color blue with a linewidth of 2? Note, your answer should be two lines. One to define an array of points, say `X`, and one to make the plot.

```
     x=linspace(-3.7, 4.2, 2003);
    plot(x,sin(x.^3 + 2*x), 'b-','Linewidth', 2)
```

2. (**3 pts**) Using a for loop based approach, write a program which finds

$$\sum_{j=1}^{n} (j^3 + 4j^2),$$

for any n. Give the answers for $n = 10$, 43, and 72.

Solution:

```
function tot = sumfun(nstop)

tot = 0;
for jj=1:nstop
     tot = tot +  (jj.^3 + 4.*jj.^2);
end
end
```

n	$\sum$
10	4565
43	1004652
72	7414464

3. (**3 pts**) Extend the program for generating Fibonacci numbers to satisfy the recursion relationship:

$$p_n = ap_{n-1} + bp_{n-2}, \; p_0 = s_0, \; p_1 = s_1.$$

Your program should take as input the values a, b, s_0, s_1, and n, and it should return p_n.

Solution:

```
function pn = gen_fib(a,b,s0,s1,n)

    p = zeros(n,1);
    p(1) = s0;
    p(2) = s1;
    for jj = 2:n
        k = jj + 1;
        p(k) = a.*p(k - 1) + b.*p(k - 2);
    end
    format long;
    pn=p(k);
end
```

For $a = 3.2$, $b = -2$, $s_0 = 3$, $s_1 = 0$, what is p_{10}? What is p_{50}?

$$p_{10} = -8.705920450560010 \times 10^3$$

$$p_{50} = -5.891661895145126 \times 10^{18}$$

4. (**4 pts**) Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

   1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

   By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms. Note, the use of the Matlab command `mod` is going to be critical.

   Solution:

   Note to grader: Students are not required to show their program, but it is preferred if they do.

```
function tot = fibonaccisum

    f(1) = 1;
    f(2) = 2;
    tot=2;
    k=2;
    while  f(k)<4000000
        k=k+1;
        f(k) = f(k-1) + f(k-2);
        if f(k)<4000000
            if mod(f(k), 2)==0;
                tot=tot+f(k);
            end
        end
    end
end
```

   The sum of even valued terms is 4613732.

5. (**4 pts**) If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6, and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.

   Solution:

```
function tot = natsum

tot=0;
for  n=1:999
    if ((mod(n,3)==0)||(mod(n,5)==0))
        tot=tot+n;
    end
end
end
```

   The sum is 233168.

6. (**3 pts**) Using Matlab and the Maclaurin series for $\sin(x)$, write a code, which computes $\sin(1)$. Explain how you choose a stopping criteria, and determine the maximum accuracy you are able to achieve.

Solution:

The Maclaurin series for $\sin(1)$ is

$$\sin(1) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}$$

Our recurrence formula shows that each iteration has the previous term in the series multiplied by $\frac{-1}{(2n+1)(2n)}$, which easily included in a program similar to the one from class for computing e^1. The MatLab code is as follows:

```
function y = sin1(tol)
%Approximation of sin(1) to a given tolerance
y = 1;
term = y;
n = 1;
while (abs(term) ≥ tol)
    term = term.*(-1)./((2*n+1)*(2*n));
    y = y + term;
    n = n + 1;
end
end
```

The value computed to maximum accuracy (double precision) is

$$\sin(1) = 0.8414709848078 97$$

This value is obtained from the program by selecting double precision tolerance or $tol = 10^{-16}$. (In fact, this accuracy is obtained because of the strong convergence whenever $tol < 10^{-13}$.) The program will continue in the `while` loop until the last term added to the series is less than the tolerance level, which by the Taylor Remainder Theorem gives the degree of accuracy desired. (Note to grader: Some students may not have understood that maximum accuracy means precision arithmetic (next section of study), so allow any discussion saying what accuracy they observed and how they obtained it.)

7. (**5 pts**) Create a Matlab function of the Maclaurin series for $\cos(x)$, which depends on `x` and a tolerance, `tol`. Explain how you choose a stopping criteria, and determine the maximum accuracy you are able to achieve. Create another MatLab function, which plots the function for $-L_x \leq x \leq L_x$ where L_x is a user specified input. Overlay a plot of the MatLab defined function `cos(x)`, using dashed lines.

Solution:

The Maclaurin series for $\cos(x)$ is

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$$

We follow the class example for e^x and obtain the cos series:

```matlab
function y = cosx(x,tol)
%Approximation to cos(x) to a given tolerance
y = 1;
term = 1;
k = 1;
while (max(abs(term)) ≥ tol)
    term = term.*(-x.^2)./((2*k)*(2*k-1));
    y = y + term;
    k = k + 1;
end
end
```

As before, the maximum accuracy would be double precision, so choosing a tolerance, $tol = 10^{-16}$. In this case, we allow the vector x to be entered, so the stopping criterion chosen is for the maximum in absolute value of the next term in the Maclaurin series, which depends on the x value, so is a vector of values. When the largest (in absolute value) of the terms is less than the prescribed tolerance, then the program terminates. Clearly, the larger the x value and the smaller the tolerance, the more iterations that the program will need to use. (Note to grader: Some students may not have understood that maximum accuracy means precision arithmetic (next section of study), so allow any discussion saying what accuracy they observed and how they obtained it.)

The program below generates the desired graph, and we use the template for graphing $y = e^x$. (We note that since this is a graph, we do NOT need double precision accuracy.)

```matlab
function cos_plot(Lx,res,tol)
% Create a plot using the exp_sum function
xx = linspace(-Lx,Lx,res);  % x points of evaluation
yy = cosx(xx,tol);        % run exp series program
yy1 = cos(xx);             % evaluate e^x with MatLab

figure(101)                   % assign a figure number

plot(xx,yy,'r-','LineWidth',1.5);  % plot the series
hold on                            % add more graphs
plot(xx,yy1,'b--','LineWidth',1.5);% plot e^x
grid;                        % provide gridlines
legend('Series cosine','MatLab cosine');

% Set up fonts and labels for the Graph
fontlabs = 'Times New Roman';
xlabel('$x$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
ylabel('$\cos(x)$','FontSize',16,'FontName',fontlabs, ...
    'interpreter','latex');
mytitle = 'Series for $\cos(x)$';
title(mytitle,'FontSize',16,'FontName', ...
    'Times New Roman','interpreter','latex');
set(gca,'FontSize',16);

print -depsc cos_gr.eps    % Create EPS file (Figure)
end
```

The graph is below in Figure 1.

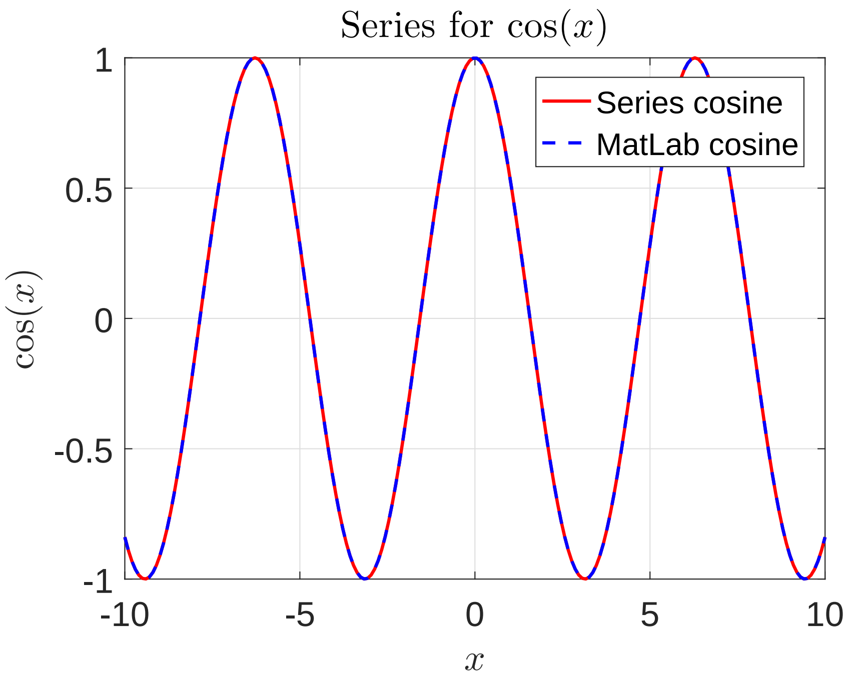


Figure 1: Plot of $y = \cos(x)$ for $x \in [-10, 10]$.

8. (**10 pts**) An important differential equation in mathematical physics is Airy's equation which is given by

$$y'' - xy = 0.$$

Two solutions to this equation can be found via the power series solutions

$$y_1(x) = 1 + \sum_{m=1}^{\infty} \frac{x^{3m}}{(2\cdot 3)(5\cdot 6)\cdots((3m-1)\cdot 3m)}$$

and

$$y_2(x) = x + \sum_{m=1}^{\infty} \frac{x^{3m+1}}{(3\cdot 4)(6\cdot 7)\cdots(3m\cdot(3m+1))}$$

Write a while based code which ultimately plots the function for $-L_x \leq x \leq 0$ and $0 \leq x \leq L_x$ where L_x is a user specified input. So, first, you would write a code which found the solutions to Airy's equation. Then, you would want a code, which plots this function.

Explain how you choose a stopping criteria and the accuracy you are able to achieve with your program. How large can you make L_x before your series solutions are unreliable? Provide plots that justify your answer. Describe the difference between the behavior of the solutions for $x < 0$ and $x > 0$. (Hint: Refer to the class notes on Bessel functions to help design these programs.)

Solution:
Using the code for generating the Bessel functions as a model, we can write one program to generate both functions via the code

```
function [y1x,y2x] = airy_maker(x,tol)
vprod = x.^3;
y1x = vprod./(2*3); y2x = vprod./(3*4);
term1 = y1x; term2 = y2x;
count = 2;
while(max(abs(term1))≥tol)
    term1 = term1.*vprod./((3*count-1)*(3*count));
    term2 = term2.*vprod./((3*count)*(3*count+1));
    y1x = y1x + term1;
    y2x = y2x + term2;
    count = count + 1;
end
y1x = 1 + y1x;
y2x = x.*(1 + y2x);
end
```

To make pretty pretty plots, I modify the `bessel_plotter` code and write

```
function airy_plotter(Llx,res)
tol=0.000001;
pos = linspace(0,Llx,res);
neg = linspace(-Llx,0,res);

[py1x,py2x] = airy_maker(pos,tol);
[ny1x,ny2x] = airy_maker(neg,tol);

figure(1)
plot(pos,py1x,'r--',pos,py2x,'b','LineWidth',1.5)
grid;
h=legend('$y_1(x)$','$y_2(x)$','Location','northwest');
set(h,'Interpreter','latex');
set(gca,'FontSize',18,'FontName','Times New Roman')
xlabel('$x$','FontName','Times New Roman','FontSize',18,...
    'Interpreter','latex')
print -depsc airy_p1.eps

figure(2)
plot(neg,ny1x,'r--',neg,ny2x,'b','LineWidth',1.5)
grid;
h=legend('$y_1(x)$','$y_2(x)$','Location','southwest');
set(h,'Interpreter','latex');
set(gca,'FontSize',18,'FontName','Times New Roman')
xlabel('$x$','FontName','Times New Roman','FontSize',18,...
    'Interpreter','latex')
print -depsc airy_n1.eps
end
```

Note, in order to get nice plots, I split my interval $-L_x \leq x \leq L_x$ up into negative and positive parts. Using `tol = 1e-6`, and $L_x = 5$, I then produced the plots in Figure 2. Note, I also generated these figures using a tolerance of `tol = 1e-8`, but there was no noticeable difference in the plots, and thus we can argue that we have converged sufficiently well with the lower tolerance. As we can see from the figures, the nature of the solution is markedly different if x is negative or positive. When $x < 0$, we see both functions oscillate as in Figure 2 (a), while for $x > 0$, both functions grow exponentially rapidly as in Figure 2 (b).

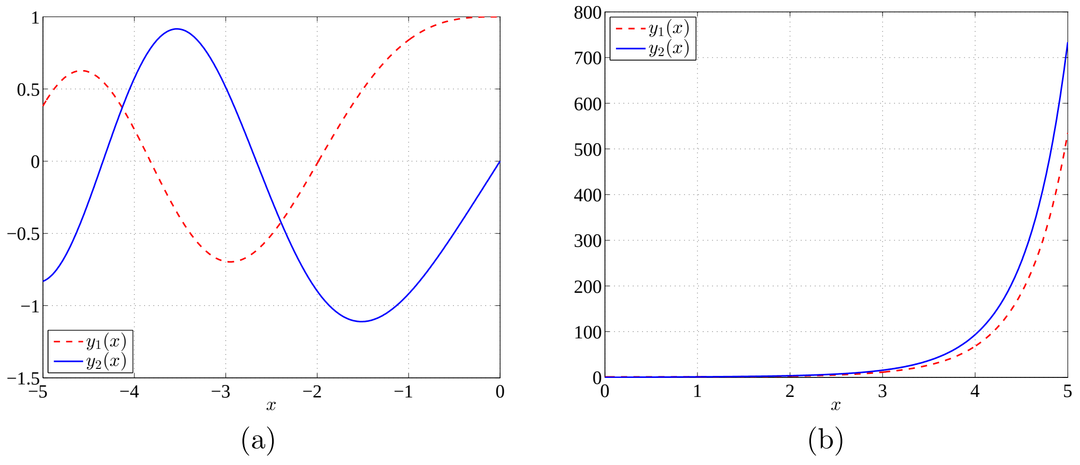


Figure 2: Plots of the solutions to the Airy equation $y_1(x)$ and $y_2(x)$ for $-5 \leq x \leq 0$ (a) and $0 \leq x \leq 5$ (b). `tol = 1e-6` for both figures.

Given that if x is positive and large we expect unbounded growth, we can get a better measure for how valid the series approach is by focusing on the case that $x < 0$. We now choose L sufficiently large so that we see a break down in the smoothness of the plotted solutions. As can be seen in Figure 3, the reliability of the series for giving an accurate representation breaks down near $x = -14$. It is easy to show that using the series itself is inherently a bad decision, since increasing the tolerance by six orders of magnitude does nothing to alleviate the problem. (Figure not shown.)

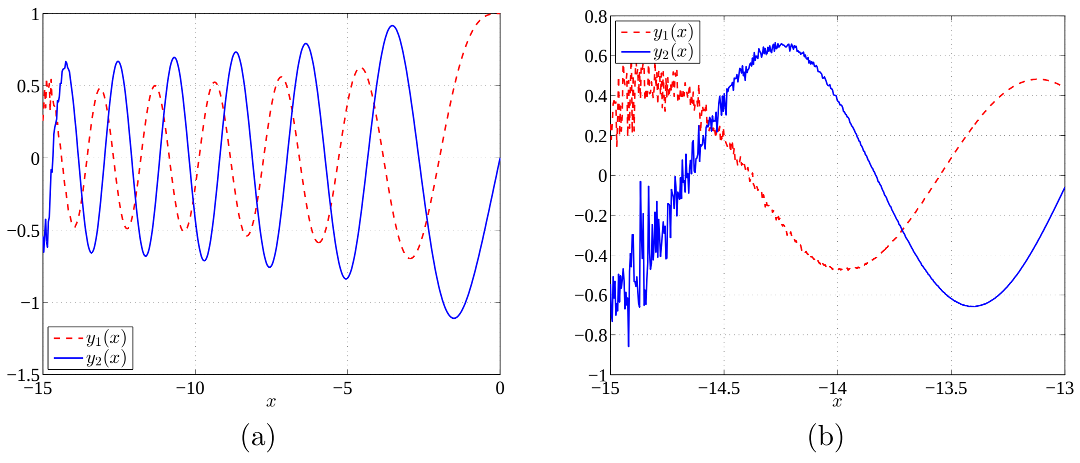


Figure 3: Plots of the solutions to the Airy equation $y_1(x)$ and $y_2(x)$ for $-15 \leq x \leq 0$ (a) and $-15 \leq x \leq -13$ (b). `tol = 1e-6` for both figures.

Results from the WeBWorK homework

1. (**4 pts**) If c_n represents the concentration of the inert gas argon (Ar) in the lungs, then a mathematical model for breathing is given by the discrete dynamical model:

$$c_{n+1} = (1-q)c_n + q\gamma,$$

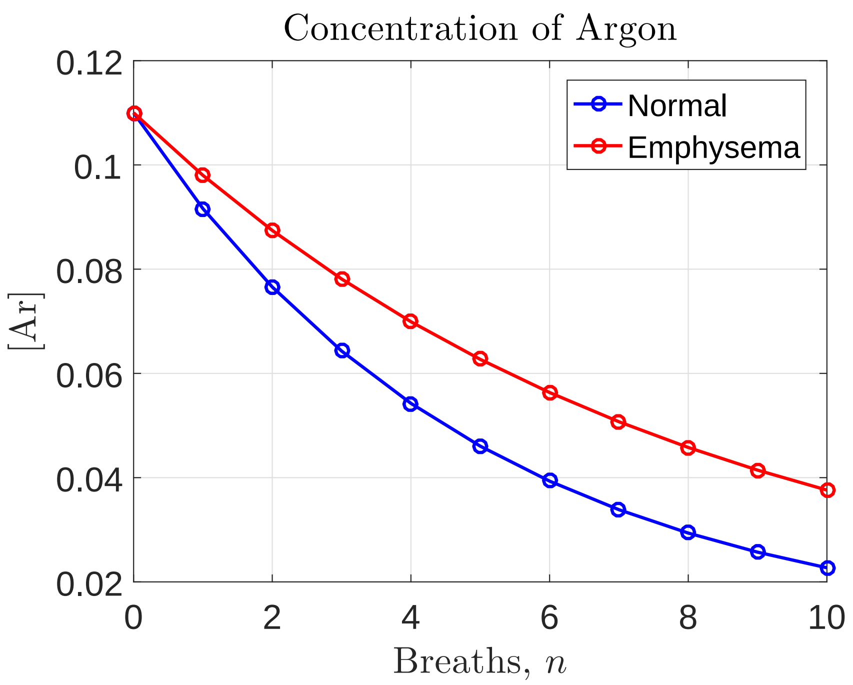
where q is the fraction of the lung volume exchanged with each breath and $\gamma = 0.0093$ (fraction of Ar in dry air) is the concentration of Ar in the atmosphere.

Create a MatLab program that produces a graph showing the concentration of Ar in the first 10 breaths of both subjects (normal and emphysema). Your line style should be data points connected by lines, e.g., `bo -` creates the appropriate blue open circle connected by a blue line. Also, create a legend to label the different subjects. Provide a title and axis labels that are appropriate for the graph produced.

Solution:

The values of q are computed in WeBWorK for both subjects and are used in the program below. The random number generator means that students will have different initial conditions and different values of q. However, the programs and graphs should be very similar.

```
function concentration(n)
%concentration of argon
%with graph
n=n+1;
q = 120/1007;
qa = 525/(2350+525);
k=1;
c(1)=0.11;
ca(1) =0.11;
    for k=2:n
       ca(k) = (1-qa).*ca(k-1)+qa*0.0093;
    c(k) = (1-q).*c(k-1)+q*0.0093;
    end
x=linspace(0,k-1,k);
plot(x,ca(x+1),'bo-', 'LineWidth', 1.5);
hold on;grid;
plot(x,c(x+1),'ro-', 'LineWidth', 1.5);
fontlabs = 'Times New Roman';
xlabel('Breaths, ...
    $n$','FontSize',16,'FontName',fontlabs,'interpreter','latex');
ylabel('[Ar]','FontSize',16,'FontName',fontlabs, 'interpreter','latex');
mytitle = 'Concentration of Argon';
title(mytitle,'FontSize',16,'FontName','Times New ...
    Roman','interpreter','latex');
set(gca,'FontSize',16);
legend('Normal','Emphysema','Location','northeast');
print -depsc problem1.eps
end
```



To solve Part b and determine the number of breaths before dropping below $[Ar] \leq 0.01$, a related program with a `while` loop is constructed.

```
function conc2
%concentration of argon with while
q = 120/1007;
k=1;
c(1)=0.11;
    while c(k)>0.01
        k=k+1;
    c(k) = (1-q).*c(k-1)+q*0.0093;
    end
k-1
format long; c
end
```

2. (**4 pts**) The population of the United States was about 39.88 million in 1870 and 50.18 million in 1880. Let 1870 be represented by P_0 and assume that its population is growing according to the Malthusian growth law,

$$P_{n+1} = (1 + r)P_n,$$

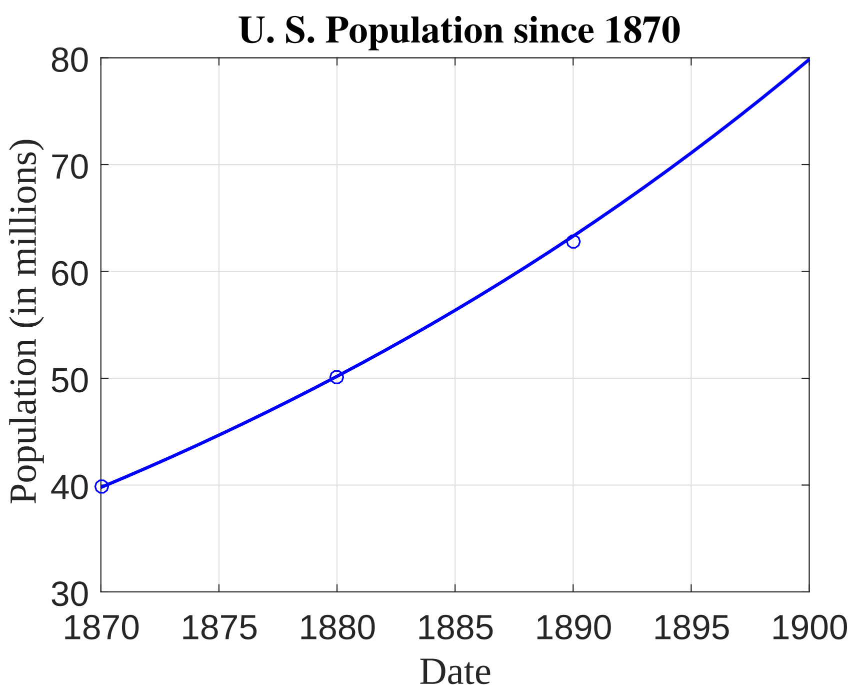
where n is in years.

Use MatLab to create a graph for your written HW of this model and the data. Plot the graph of the Malthusian model over 30 years from 1870, using a line for the model. Use open circles to graph all of the census data (3 points) given above. Provide a title and axis labels that are appropriate for the graph produced.

Solution: A program is designed to graph the Malthusian growth model, the we include the 3 data points for actual census data. (Note: The starting year and various given populations vary between students, but programs should have similar structures.)

```matlab
function popul2
% population graph
r=(50.18/39.78)^(0.1)-1;
k=1;
P(1)= 39.78;
for k = 1:30
    P(k+1)=(1+r)*P(k);
end
x=linspace(1870,1900,31);
hold off;
plot(x,P,'b-', 'LineWidth', 1.5);
hold on; grid;
plot([1870 1880 1890],[39.88 50.18 62.86],'bo');
fontlabs = 'Times New Roman';
xlabel('Date','FontSize',16,'FontName',fontlabs);
ylabel('Population (in millions)','FontSize',16,'FontName',fontlabs);
mytitle = 'U. S. Population since 1870';
title(mytitle,'FontSize',16,'FontName','Times New Roman');
set(gca,'FontSize',16);

print -depsc problem2.eps
end
```



To solve Part c and determine the doubling time for this population, a related program with a `while` loop is constructed, showing how long until the population doubles from the population in 1870.

```
function popul1
% population doubling
r=(50.18/39.88)^(0.1)-1
P(1)= 39.88;
k = 1;
while (P(k) ≤ 2*39.88)
    k = k+1;
    P(k)=(1+r)*P(k-1);
end
k-1
format long; P
end
```

3. (**7 pts**) This problem studies the behavior of the discrete logistic growth model as the growth parameter varies. For certain parameter values, it is possible for this discrete model to exhibit chaotic behavior. The discrete logistic growth model satisfies

$$P_{n+1} = f(P_n) = P_n + rP_n\left(1 - \frac{P_n}{M}\right).$$

This problem explores some of the complications that can arise as the parameter r varies.

Create a MatLab graph over 50 generations showing the simulations for $r = 0.84$ and 1.64. Create another graph showing the simulations for $r = 2.21$ and 2.47. Finally, create a graph showing the simulation for $r = 2.57$. Simply use line segments to connect the points of your simulation. Add a legend to label the different r values. Provide a title and axis labels that are appropriate for the graph produced.

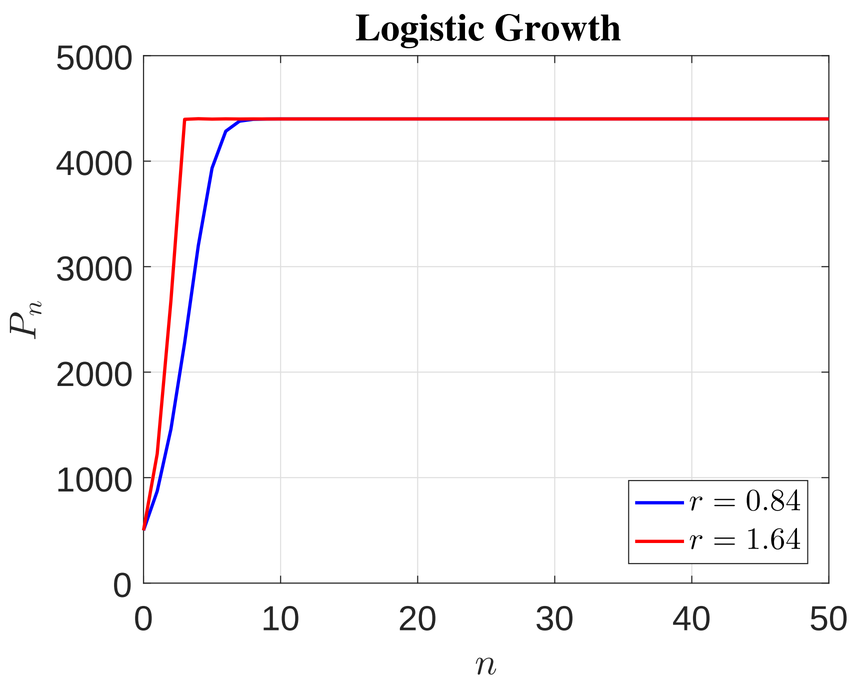
Solution: A program is designed to graph the discrete Logistic growth model. Our example (different for different students) started $P_0 = 500$. The carrying capacity (nonzero equilibrium) is $M = 4400$. Simulations ran for 50 iterations with varying r values.

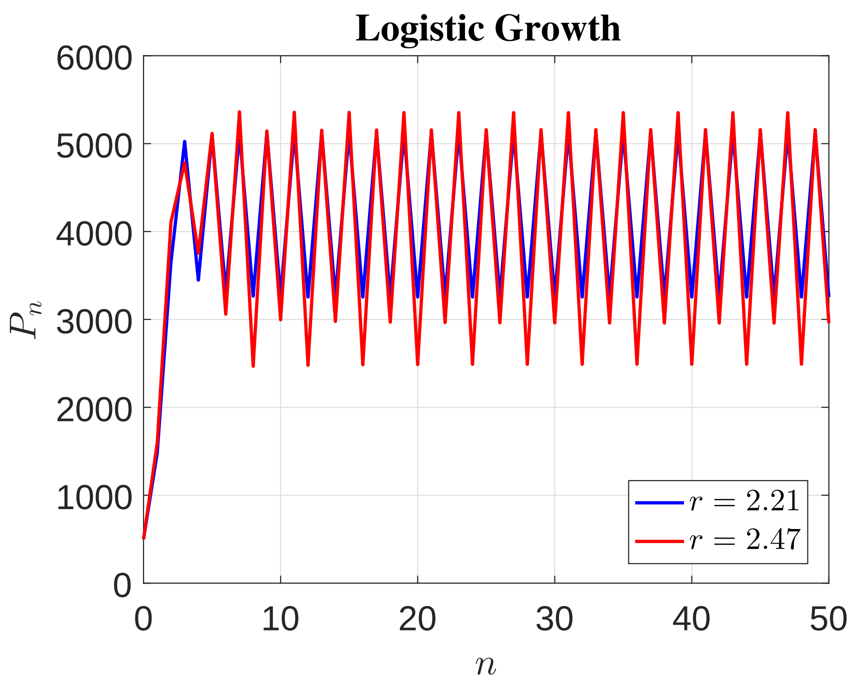
A version of the graphing program is seen below.

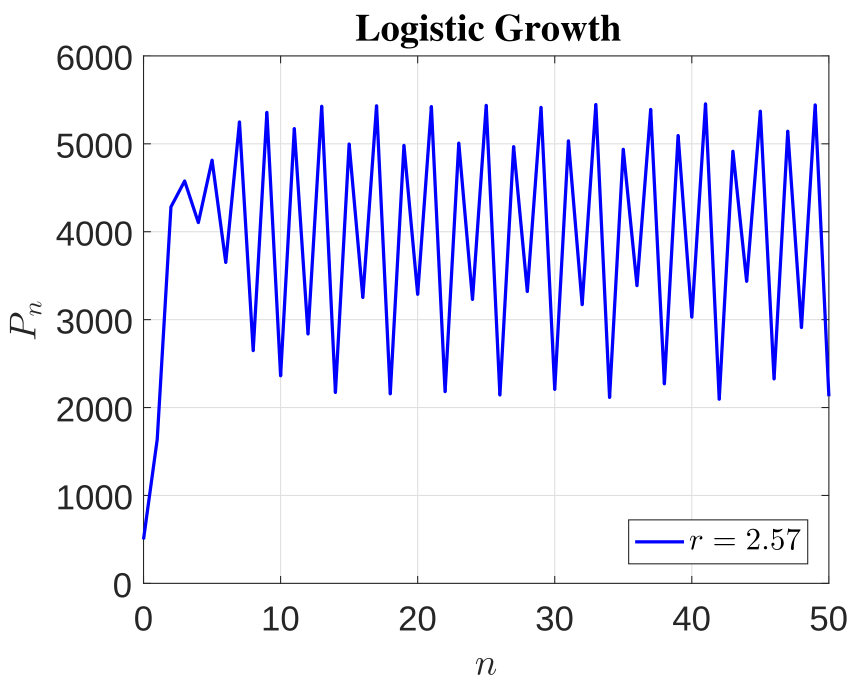
```
function discrete(n, P0)
%WW question 3 discrete dynamics
ra =0.84;
rb =1.64;
M=4400;
Pa(1)= P0;
Pb(1)=P0;
for k=2:n+1
    Pa(k)=Pa(k-1)+ra*Pa(k-1)*(1-Pa(k-1)/M);
    Pb(k)=Pb(k-1)+rb*Pb(k-1)*(1-Pb(k-1)/M);
end
x=linspace(0,k-1,k);
plot(x,Pa(x+1),'b-', 'LineWidth', 1.5);
hold on; grid;
plot(x,Pb(x+1),'r-', 'LineWidth', 1.5);
fontlabs = 'Times New Roman';
xlabel('$n$','FontSize',16,'FontName',fontlabs,'interpreter','latex');
```

```
ylabel('$P_n$','FontSize',16,'FontName',fontlabs, 'interpreter','latex');
mytitle = 'Logistic Growth';
title(mytitle,'FontSize',16,'FontName','Times New Roman');
set(gca,'FontSize',16);
h=legend('$r = 0.84$','$r = 1.64$','Location','southeast');
set(h,'interpreter','latex');
print -depsc discrete3a.eps
end
```

Graphic output for the 5 values of r are shown below:






Logistic Growth
6000
5000
4000
3000
2000
1000
0
P_n
0
10
20
30
40
50
n
$r = 2.57$