

<b>Name: Andrew Brown</b>	<b>Lab Time T 12:00</b>
---------------------------	-------------------------

<b>Names of people you worked with:</b>
<ul style="list-style-type: none"><li>• Kyle Magness</li><li>• Kevin Dellenbach</li></ul>

<b>Websites you used:</b>
<ul style="list-style-type: none"><li>•</li></ul>

<b>Approximately how many hours did it take you to complete this assignment (to nearest whole number)?</b>	12
--	----

By writing or typing your name below you affirm that all of the work contained herein is your own, and was not copied or copied and altered.

Andrew Brown

---

**Note: Failure to sign this page will result in a 50-point penalty. Failure to list people you worked with may result in no grade for this homework. Failure to fill out hours approximation will result in a 10-point penalty.**

**Turn .zip files to Canvas or your assignment will not be graded**

**WARNING:** For most of you this homework will be substantially more difficult than the previous two have been. Start early.

### Learning Objectives:

You should...

- Understand how to use **for** loops to avoid replicating code and for iterating over arrays.
- Know how to use relational operators to ask questions like greater than/less than/the same about data.
- Know how to use **if** statements to control program flow.

### Homework Guidelines:

#### 1) Loops:

- a. You can start by manually writing the first step of the loop, then writing the second step.
- b. Once those steps work, put the **for** loop around the line(s)
  - i. How many times should you loop? When should you stop? How do you have to change the line so that it works with the loop variable?
- c. You can also try downloading example scripts from online, choosing those closest to your desired script and editing them (understand them first).

#### 2) If statements/relationals:

- a. Write it out in pseudo code first.

#### 3) Indenting:

- a. Maintain correct indenting at all times – this will help you “see” your code structures and check if your ends are in the right place (see Lab 3 for more detail).
- b. You can use this green button to do it for you:







#### 4) Debugging:

- a. Before running your code, put a break point at the first line of every **for** loop and in every **if** statement (click on the number on the left). Then you can “watch” to see if a) you go into the **if** statement when you expect to and b) if you are doing the right thing in your **for** loop.
- b. Put your cursor over the variables (eg, **k**, **open** etc) to see what values they take on. You can also check what they are by typing them in the command line (eg, type **dayToSell(k)** in the command window).

```
13 % Now print out days to sell
14 - for k = 1:length(dayToSell) % k will go from 1 to the length of the vector, ir
15     % Could also do close(k) > open(k)
16     % need close(k) (not just close) because want to compare only one elemen
17     % of close to one element of open
18     if dayToSell(k) % Check the value for this value of k
19         % close(k) gives the kth element in close
20         fprintf('Profit on day %d is $ %2.2f\n', k, close(k) - open(k) );
21     end
22 - end
23
```

**Specific Coding Notes:**

- 1) In general, it is better form to separate calculations from printing out the values of the calculations, if possible.
- 2) Use a loop index variable (eg, **k = 1:length(xs)**) rather than having the loop variable take on the values (eg, **x = xs**). This is useful for separating out where you are in the vector (**k**) from the value in the vector (**xs(k)**).
- 3) Don't hard-wire values in - eg, use **length(xs)** instead of **5** for creating index variables. Declare variables for constants.

 NO 	 YES 
<pre>xs = zeros(1, 5);  % k is not based on xs length for k = 1:5     x(k) = k*2; end</pre>	<pre>xs = zeros(1, 5);  % k IS based on xs length for k = 1:length(xs)     x(k) = k*2; end</pre>

**Grading Checkpoints**

Criteria	Component	No	Yes
<b>[30%] Comments and Pseudocode</b>	Declared units on all variables?		
	English description of problem at top?		
	Comments outlining your steps? *Write out what your loops and if statements are doing*		
<b>[20%] Output formatting</b>	Used fprintf() to make complete sentences (when required)?		
	Correct units on answers?		
	Correct number of decimal places?		
<b>[50%] Functionality</b>	Script computes/plots correct value(s)?		
	Correctly converted units in script when needed?		
	Use for loop and if statement correctly?		
	Use for loop instead of duplicating code?		
	Ensure loops work for arbitrary vectors (i.e. use length (x) instead of hard-wiring number of elements)? *See specific coding notes*		

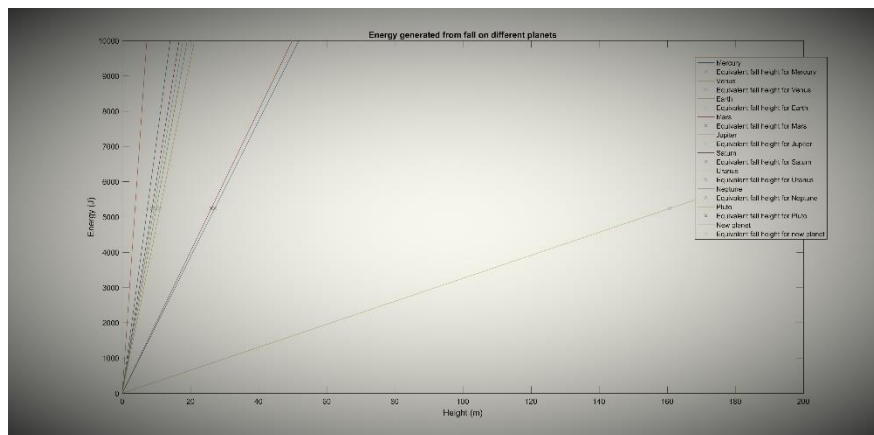
## Problem 1

This problem is a continuation of HW2, Problem 3.

- Update your code to include a variable 'gValues' which is an 1x8 array containing the eight g values for each of the solar system planets.
- For each of the 8 planets, **plot the energy** generated by the fall of a 120-pound object over the surface of that planet. Your plot should show the energy generated between values heights of 0 to 100 meters. You must use loops to get credit.
- Identify **on your graph and command window** the harmful heights that will generated the same energy generated by a 10-meter fall on earth. You must use loops to get credit.
- EXTRA CREDIT [100 points]** Ask the user to input a g-value for any newly discovered planet. If g is not equal to any of the existing 8 planets, then **do steps (b) and (c) for this planet over the same figure**. If g is equal to any of the 8 existing planets, then just **print a message** like this one:

"The size of this planet is identical to *Mars*"

Self check: (Your axes may be different)



### Comments for grader/additional information (if any)

I did the extra credit

### Script File

```
%Andrew Brown Homework 3 Problem 1

clc
clear
close all

%Physics of a harmful fall on different planets using loops
```

```

%Define given constants
massPounds=120; %mass in pounds
massKg=massPounds/2.246; %mass in kilograms
PlanetNames={'Mercury','Venus','Earth','Mars','Jupiter','Saturn','Uranus','Neptune','Pluto'}; %array of planet names
newPlanet=input('New Planet Gravity:'); %user input for a new planet's gravity in m/s^2
gValues=[3.61, 8.83, 9.81, 3.75, 26.0, 11.2, 10.5, 13.3, 0.61]; %accelerations due to gravity in m/s^2

%If the new planet is identical in size to an existing planet, say so, if not, add the new planet to gValues and run it through the rest of the code
if newPlanet~=gValues
    gValues(10)=newPlanet; %adds the new planet gravity to a new index in gValues
else
    EqualPlanet=find(newPlanet==gValues); %stores the new planet as the array index of its equal planet in gValues
    fprintf('The size of your planet is identical to %s\n',PlanetNames{EqualPlanet(1)}) %Prints the planet that is equal to the new planet input
end
h=linspace(1,200,200); %height off the ground in meters

%Calculate the energy given off by falling on each planet from variable heights
EPlanetStore=zeros(1,length(h)); %Pre-allocate the energy for the fall on each planet
for i=1:length(gValues)
    EPlanet=massKg*gValues(i)*h; %energy generated from the fall in joules on earth
    EPlanetStore(i, :)=EPlanet;
end

%Calculate the height of an equivalent fall on different planets
PlanetHeightStore=zeros(1,length(h)); %Pre-allocate the individual planet heights
for k=1:length(gValues)
    PlanetHeight=EPlanetStore(3,10)/(gValues(k)*massKg);
    PlanetHeightStore(k)=PlanetHeight;
end

%Plotting the graph for energy generated from a fall on Earth and the equivalent falls on different planets
for j=1:length(gValues)
    hold on
    plot(h,EPlanetStore(j, :))
    plot(PlanetHeightStore(j),EPlanetStore(3,10),'r*')
end

%Misc. graph labels, legend, and defining axes.
title('Energy Generated from a fall on Different Planets')
xlabel('Height (m)')
ylabel('Energy (J)')
axis([0,200,0,10^4])

```

```
%Make array for the legend labels so we can exclude the new planet labels
%if they do not exist
legendLabels={'Mercury','Equivalent fall for Mercury','Venus',...
    'Equivalent fall for Venus','Earth','Equivalent fall for Earth',...
    'Mars','Equivalent fall for Mars','Jupiter',...
    'Equivalent fall for Jupiter','Saturn','Equivalent fall for Saturn',...
    'Uranus','Equivalent fall for Uranus','Neptune',...
    'Equivalent fall for Neptune','Pluto','Equivalent fall for Pluto',...
    'New Planet','Equivalent fall for New Planet'};
legend(legendLabels(1:length(gValues)*2)) %make the legend relying off the
length of gValues
```

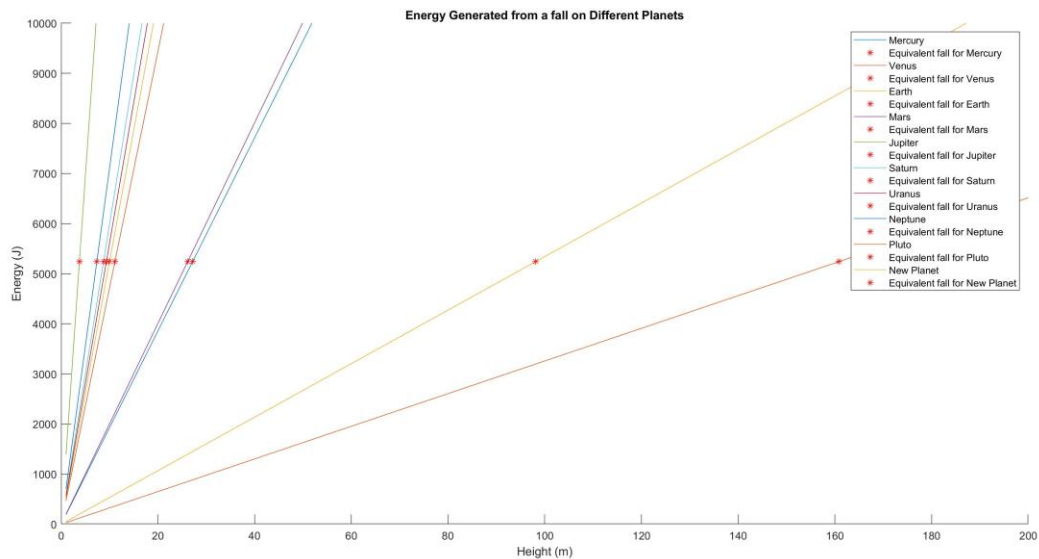
### Command Window Output

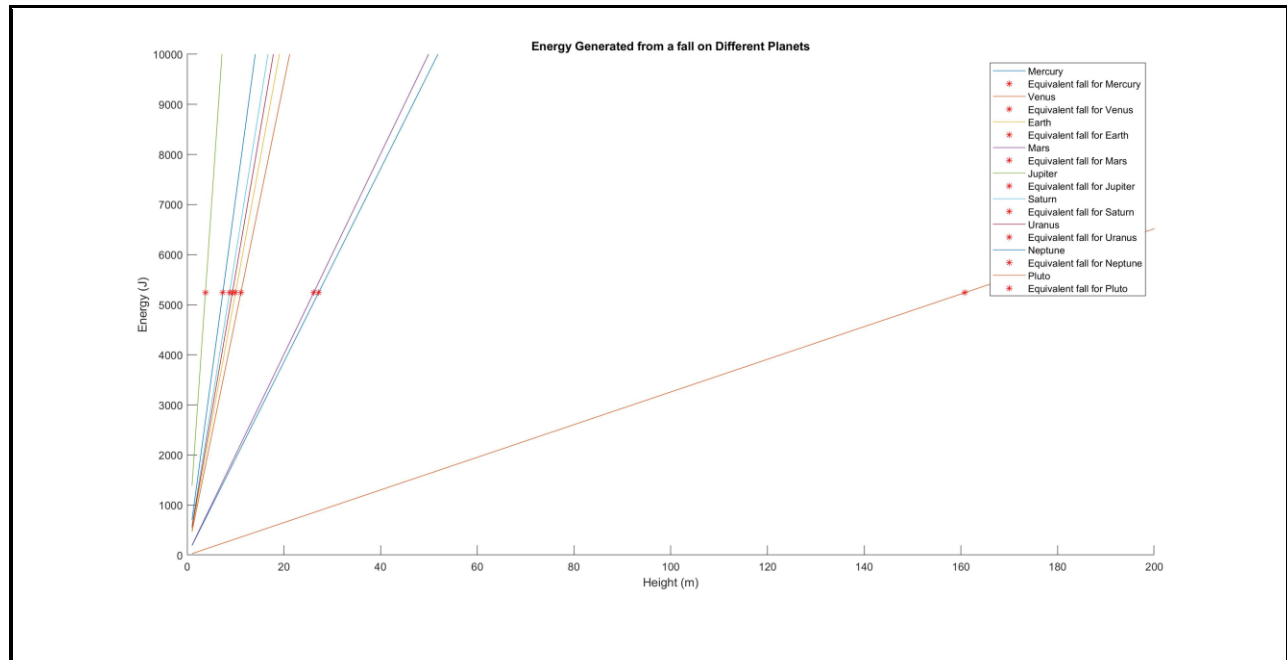
New Planet Gravity:9.81  
 The size of your planet is identical to Earth

OR

New Planet Gravity:1.0

### Image Output







## Problem 2

Continuation of HW2, Problem 2: Total Solar Eclipse (TSE) budgeting.

This problem will help your program be more usable and flexible. Apply the following improvements to your code so it does the following:

- a) **Ask the user for the required budget.** (In HW2, we assumed that it takes \$3,000 to travel to a solar eclipse. Some people would like to buy camera and gear, so we want them to define their own required budget)
- b) **Ask the user to input their daily savings rate** starting from 1-1-2018.
- c) **Ask the user to input their initial savings.** (How much money you start with on 1-1-2018. Note that you started with a balance of zero dollars in the past homework.)
- d) Using **loops and if statements**, identify all future total solar eclipse events that you will be able to afford to go to.

Test your code using these parameters:

- Required budget= \$4000
- Daily Savings (starting from January 1<sup>st</sup> 2018= \$4
- Initial savings= \$200

**How many eclipse events can you choose to attend out of the next 6 upcoming eclipses?**

- e) **[Extra credit +25pts]** Using an if statement, adjust your program to print an appropriate response if the savings budget won't allow you to attend any of the upcoming eclipses. Note if you did this in the **Comment for grader** box.

Self check: the output should look like this:

```
With a required budget of $4000, you can attend the following total solar
eclipse(s) :
A total solar eclipse that occurs at 10xx days with a savings balance of
$4x1x
A total solar eclipse that occurs at 14xx days with a savings balance of
$5xx6
A total solar eclipse that occurs at 1xxx days with a savings balance of
$7xx8
A total solar eclipse that occurs at 2xxx days with a savings balance of
$9xx0
A total solar eclipse that occurs at 3x4x days with a savings balance of
$12xxx
```

Self-check for the EC:

What is your required budget?-->\$9999

How much will you save per day?-->\$1

Do you have an initial savings amount that you want to put aside?-->\$0

You will not be able to attend any eclipse in the next 3147. In this number of days, you will be having \$3147 only in your savings

**Comments for grader/additional information (if any)**

Extra credit

**Script File**

```
%Andrew Brown Homework 3 Problem 2

clc
clear
close all

%Continuation of HW2 Problem 2 TSE budgeting with practice using loops, if
%statements and user inputs with arrays.

%Total Solar Eclipse Part 1

%Part A: Create three variables: Year, Month, and Day. These three
%variables should contain a vector of the corresponding values of the
%TSE dates.

%Defining the three time-based variables with the given data.
day=[2,14,4,20,8,12]; %Days of the future TSE
month=[7,12,12,4,4,8]; %months of the future TSE
year=[2019,2020,2021,2023,2024,2026]; %years of the future TSE

%Part B: Calculate how many days between January 1st, 2018 and each of
%the upcoming TSE dates. Use the assumption that the Gregorian (western)
%solar calendar month, on average has 30.44 days. And every year has on
%average 365.2425 days.

%Do the needed year and month conversions into days
daysInAMonth=30.44; %Average number of days in a month
daysInAYear=365.2425; %Average number of days in a year

%Set up the arrays into only days for the end calculation
daysSubtracted=736695; %Days before Jan. 1, 2018
monthsToDays=(month-1).*daysInAMonth; %convert months to days
yearsToDays=(year-1).*daysInAYear; %convert years to days

%Calculation of days between Jan 1, 2018 and the given TSEs
daysUntilTSE=yearsToDays+monthsToDays+day-daysSubtracted;
roundedDaysUntilTSE=round(daysUntilTSE);
%Part C: Calculate the daily saving balance that you will accumulate in
%your account starting from day one until and ending on day 3,200.
%(Assume that you are adding $2/day into your account. No interest is
%added to your saving account). Assign the daily savings of all 3,200
%values into one array.

%Ask the user for the required budget
budget=input('Required Budget: $');

%PART 2 START
```

```

%Ask the user to input their daily and initial savings starting from 1-1-2018
savings=input('Daily savings starting 1-1-2018: $'); %user input savings in $
initialSavings=input('Intial savings on 1-1-2018: $'); %user input initial
savings in $
dailySavingsBalance=initialSavings:savings:11^4; %Array for the daily savings
that counts by your dailysavings and starts at your initial savings

%Create a counter for i that breaks when you can afford a trip or when you
cannot afford any trips
i=1;
while roundedDaysUntilTSE(i)*savings+initialSavings<budget
    i=i+1; %increase i when you cannot afford the trip
    if i==6
        break %if you cannot afford any trip, break the if statement and
while loop
    end
end

%Produces the correct statement based on whether or not you can afford any
eclipse
if i==6
    fprintf('Your savings budget will not allow you to attend any of the
upcoming eclipses.\n'); %print when you cannot afford any trip
else
    fprintf('With a required budget of $%.2f you can attend the following
total solar eclipse(s):\n',budget); %print when you can afford a trip
end

%Prints out the needed values for the eclipses that you can afford.
for j=1:6
    if roundedDaysUntilTSE(j)*savings+initialSavings>=budget %if your savings
are more than or equal to your budget
        fprintf('A total solar eclipse that occurs on %0.0f days with a
savings balance of
$%.0f\n',roundedDaysUntilTSE(j),roundedDaysUntilTSE(j)*savings+initialSavings)
        %print out which trips you can afford
    end
end

%Part 1 D: Generate a plot that shows all of the following:
%1. The value of your daily savings.
%2. Plot a marker on the plot for each TSE and your savings that day.
%3. Plot a dashed horizontal line for savings = $3000
%4. Print out how many days are left until the first TSE you can afford and
%their savings balance that day.

countDays=1:length(dailySavingsBalance); %Array to count the days in order

%Plot the Solar Eclipse Budget Simulation
plot(countDays,dailySavingsBalance,'b')
title('Solar Eclipse Budget Simulation')
xlabel('Days')
ylabel('Money Saved ($)')
hold on

```

```
%Plot a horizontal line at y=3000 using the lines function
horizontalLine=ones(1,length(countDays));
plot(countDays,budget.*horizontalLine,'--k')

%Plot each solar eclipse date and savings.
plot(roundedDaysUntilTSE(1),dailySavingsBalance(roundedDaysUntilTSE(1)),'xr',
'MarkerSize',10) %2019 eclipse
plot(roundedDaysUntilTSE(2),dailySavingsBalance(roundedDaysUntilTSE(2)),'*r',
'MarkerSize',10) %2020 eclipse
plot(roundedDaysUntilTSE(3),dailySavingsBalance(roundedDaysUntilTSE(3)),'+r',
'MarkerSize',10) %2021 eclipse
plot(roundedDaysUntilTSE(4),dailySavingsBalance(roundedDaysUntilTSE(4)),'sg',
'MarkerSize',10) %2023 eclipse
plot(roundedDaysUntilTSE(5),dailySavingsBalance(roundedDaysUntilTSE(5)),'dg',
'MarkerSize',10) %2024 eclipse
plot(roundedDaysUntilTSE(6),dailySavingsBalance(roundedDaysUntilTSE(6)),'og',
'MarkerSize',10) %2026 eclipse

%Create the legend in the SW corner
legend('Savings','Required Budget','2019 Eclipse','2020 Eclipse','2021
Eclipse','2023 Eclipse','2024 Eclipse','2026 Eclipse','location','SE')
```

### Command Window Output

```
Required Budget: $4000
Daily savings starting 1-1-2018: $4
Initial savings on 1-1-2018: $200
With a required budget of $4000.00 you can attend the following total solar
eclipse(s):
A total solar eclipse that occurs on 1078 days with a savings balance of
$4512
A total solar eclipse that occurs on 1434 days with a savings balance of
$5936
A total solar eclipse that occurs on 1937 days with a savings balance of
$7948
A total solar eclipse that occurs on 2290 days with a savings balance of
$9360
A total solar eclipse that occurs on 3146 days with a savings balance of
$12784

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
EXTRA CREDIT

Required Budget: $4000
Daily savings starting 1-1-2018: $1
Initial savings on 1-1-2018: $0
Your savings budget will not allow you to attend any of the upcoming
eclipses.
```

### Answers to question(s) asked in the homework (if any)

d.) How many eclipse events can you choose to attend out of the next 6 upcoming eclipses?  
5 (with the given data)

## Problem 3

### Part 3 TSE

Estimating all Total Solar Eclipse (TSE) dates in the 21<sup>st</sup> century.

Write a program that estimates all total solar eclipse dates that happened and will happen in the 21<sup>st</sup> century. Use the following:

- Use the sample of 6 TSE dates to calculate the average difference in days between any two consecutive TSEs. (Hint: To calculate the difference between elements in an array `x`, you can use the function `diff(x)`. Use MATLAB help to learn more about the function)
- Starting from the last TSE in the last century (August 11, 1999), calculate the next 150 TSE dates. (Hint: You have to use a `for` loop). Save these dates into a variable in an array form. **Do not print them to the command window.**
- Count all TSE's that occur between years 2001 and 2100. **Print on the command window how many TSE's do you expect to occur in the 21<sup>st</sup> century.** *An answer within 10% accuracy is ok, as long as you justify your answer in part (d)*
- Answer this question: According to [Eclipsewise.com](https://eclipsewise.com), This century will have 68 total solar eclipses and 7 hybrid eclipses (75 total). Compare your answer in part (c) to this number. How does it compare? Is there an error? If so, what assumptions in this method or in your calculations might be the cause of that error?
- [Extra credit: +25]** What suggestions would you offer to increase the accuracy of this program? No need to write the code, just answer in the appropriate box below.

#### Comments for grader/additional information (if any)

#### Script File

```
%Andrew Brown Homework 3 Problem 3

clc
clear
close all

%Continuation of HW3 Problem 2 TSE with practice using loops, if
%statements, user inputs with arrays and other matlab operators.

%Part 1A: Create three variables: Year, Month, and Day. These three
%variables should contain a vector of the corresponding values of the
%TSE dates.

%Defining the three time-based variables with the given data.
day=[2,14,4,20,8,12]; %Days of the future TSE
month=[7,12,12,4,4,8]; %months of the future TSE
year=[2019,2020,2021,2023,2024,2026]; %years of the future TSE
```

```

%Part B: Calculate how many days between January 1st, 2018 and each of
%the upcoming TSE dates. Use the assumption that the Gregorian (western)
%solar calendar month, on average has 30.44 days. And every year has on
%average 365.2425 days.

%Do the needed year and month conversions into days
daysInAMonth=30.44; %Average number of days in a month
daysInAYear=365.2425; %Average number of days in a year

%Set up the arrays into only days for the end calculation
daysSubtracted=736695; %Days before Jan. 1, 2018
monthsToDays=(month-1).*daysInAMonth; %convert months to days
yearsToDays=(year-1).*daysInAYear; %convert years to days

%Calculation of days between Jan 1, 2018 and the given TSEs
daysUntilTSE=yearsToDays+monthsToDays+day-daysSubtracted;
roundedDaysUntilTSE=round(daysUntilTSE);

%PART 3

%Calculate the average of the difference between consecutive TSEs
avgDif=mean(diff(daysUntilTSE)); %in days

%Calculate 150 TSE dates that will happen after Aug. 11, 1999
lastTSE=(1998*daysInAYear)+(7*daysInAMonth)+11; %days before Aug. 11, 1999
saveDate=zeros(1,150); %Preallocate savedate
for i=1:150
    tseDate=avgDif.*i+lastTSE;%calculate the date
    saveDate(i)=tseDate; %save the date to a preallocated array
    if saveDate(i)<=(2099*daysInAYear) %Calculate the TSE that will happen
between 2001 and 2100
        saveBetterDate(i)=saveDate(i); %Save values into new array
    end
end

%Count all the TSEs that occur between 2001 and 2100
numTSE=length(saveBetterDate); %caluclate the length of the saveBetterDate
array
disp(numTSE) %display the length in days

```

### Command Window Output

70

### Answers to question(s) asked in the homework

- e) According to [Eclipsewise.com](http://Eclipsewise.com), This century will have 68 total solar eclipses and 7 hybrid eclipses (75 total). Compare your answer in part (c) to this number. How does it compare? Is there an error? If so, what assumptions in this method or in your calculations might be the cause of that error?

The cause of the error is the assumption that the TSEs happen at even intervals of approximately 519.4 days, when in reality, the eclipses occur due to the orbits of the moon, earth, and sun.

f) **Extra Credit:** What suggestions would you offer to increase the accuracy of this program?

To increase the accuracy of this program, I would instead program (or download off the internet) a simulation of the orbits of the sun, earth, and moon, and count a TSE s when they line up in the form of a TSE. This would be the most accurate way of doing this calculation.

## Problem 4

Epidemic problem 3 [Multiple time steps]

### Comments for grader/additional information (if any)

### Script File

```
% Andrew Brown Homework 3 Problem 4

clc
clear
close all

% Epidemic Part 3: more practicing with if and for loops. In particular, I
% practiced with matrixes in this code.

%Initial Values
a=10; %the contact rate: the average # of people a person comes in contact
with.
b=1.25; %the amount of time in days that a person is infectious
S0=0:500:2000; %Susceptibles, those who have never had the illness and can
catch it.
I0=100; %Infectives, those who are infected and are contagious.
R0=0; % Recovered, those who already had the illness and are immune.
h=0.05; %timestep in days
ts=1:140; %timesteps

%Preallocate all necessary matrices
dS=zeros(length(ts)-1,length(S0)); %Preallocate dS
dI=dS; %Preallocate dI
dR=dS; %Preallocate dR
S=zeros(length(ts),length(S0)); %Preallocate S
I=S; %Preallocate I
R=S; %Preallocate R

%Set first rows to the given values S0 and I0
S(1, :)=S0; %Set first row of S to the S0 array
I(1, :)=I0; %Set the first row of R to all equal the constant R0

%Run all 140 timesteps with a for loop with the given equations
for i=1:(length(ts)-1)
    %Equations used for calculations of the first timestep.
    N=S0+I0+R0; %Total population
    dS(i, :)=(-a.*S(i, :).*I(i, :))./N; %How S changes
    dI(i, :)=((a.*S(i, :).*I(i, :))./N)-(I(i, :)./b); %How I changes
    dR(i, :)=I(i, :)./b; %How R changes

    %Calculations for values over time using the previous timestep.
    S(i+1, :)=S(i, :)+(h.*dS(i, :)); %# of Sick people after time h.
    I(i+1, :)=I(i, :)+(h.*dI(i, :)); %# of Infected people after time h.
```



```

        R(i+1, :)=R(i, :)+(h.*dR(i, :)); %# of Recovered people after time h.
    end

%Print out ending values after 140 timesteps
for j=1:length(S0)
    fprintf('Start S=%0.0f, Ending after %0.2f Days: S = %0.0f, I = %0.0f, R
= %0.0f, total %0.0f\n',
S(j),h*length(ts),S(length(ts),j),I(length(ts),j),R(length(ts),j),N(j))
end

%Plot S, I, R, and Totals vs time
for k=1:length(S0)
    %Plot S vs Time
    hold on %keep all on same plot
    subplot(1,4,1) %plot in first position
    plot(ts,S(:, k),'xk') %plot s vs time
    title('Susceptibles vs. Time') %title the plot
    xlabel('Time') %X label for the plot
    ylabel('Number of People') %Y label for the plot
    axis([0,140,0,2200]) %Set proper axes

    %Plot I vs Time
    hold on %keep all on same plot
    subplot(1,4,2) %plot in second position
    plot(ts,I(:, k),'xk') %plot I vs time
    title('Infected vs. Time') %title the plot
    xlabel('Time') %X label for the plot
    ylabel('Number of People') %Y label for the plot
    axis([0,140,0,2200]) %Set proper axes

    %Plot R vs Time
    hold on %keep all on same plot
    subplot(1,4,3) %plot in third position
    plot(ts,R(:, k),'xk') %plot R vs time
    title('Recovered vs. Time') %title the plot
    xlabel('Time') %X label for the plot
    ylabel('Number of People') %Y label for the plot
    axis([0,140,0,2200]) %Set proper axes

    %Plot Total vs Time
    hold on %keep all on same plot
    subplot(1,4,4) %plot in fourth position
    plot(ts,N(k),'xk') %plot total vs time
    title('Total vs. Time') %title the plot
    xlabel('Time') %X label for the plot
    ylabel('Number of People') %Y label for the plot
    axis([0,140,0,2200]) %Set proper axes
end

```

### Command Window Output

```

Start S=0, Ending after 7.00 Days: S = 0, I = 0, R = 100, total 100
Start S=0, Ending after 7.00 Days: S = 0, I = 2, R = 598, total 600
Start S=0, Ending after 7.00 Days: S = 0, I = 5, R = 1095, total 1100
Start S=0, Ending after 7.00 Days: S = 0, I = 7, R = 1593, total 1600

```

Start  $S=0$ , Ending after 7.00 Days:  $S = 0$ ,  $I = 10$ ,  $R = 2090$ , total 2100

- a.) None
- b.) 500, 1000, 1500, 2000
- c.) It decreases towards zero

### Image Output

