Homework (50 pts) …… Due date: 11/11/2016

(Chapter 6 –Lent book) Answer and submit in one script file the.

Write the program and outputs as directed in the questions below:

Q2- (5 pts)- Decaying circle

Q4- (5 pts)- Lissajous

Q7- (10 pts)- Epicycles

Q8- (10 pts)- A random walk in two dimensions

Q9- (10 pts)- Vertical motion and energy

Q10-(10 pts)- Two dimensional billiards

PROGRAMMING PROBLEMS

1. **Circle.** Write a program, `plotcirc.m`, that animates the plotting of the following parametrically defined motion with period T. In addition to plotting the circle, plot a line from the center of the circle to the point, for each point plotted.

$$r = r_0$$
$$x = r\cos(\omega t)$$
$$y = r\sin(\omega t)$$

2. **Decaying circle.** Write a program, `plotdecay.m`, that animates the plotting of the following parametrically defined motion characterized by T and τ. In addition to plotting the motion, plot a line from the origin to the point, for each point plotted.

$$r = r_0 e^{-\frac{t}{\tau}}$$
$$x = r(t)\cos(\omega t)$$
$$y = r(t)\sin(\omega t)$$

3. **Ellipse.** Write a program, `plotellipse.m`, that animates the plotting of the following parametrically defined motion characterized by T and ϕ.

$$x = r\cos(\omega t)$$
$$y = r\sin(\omega t + \phi)$$

4. **Lissajous.** Write a program, `plotLJ.m`, that plots the following parametrically defined motion characterized by T, a, b, and ϕ.

$$x = r\cos(a\omega t)$$
$$y = r\sin(b\omega t + \phi)$$

   Add the option to animate the motion (`animateON=true or false`) in a second figure (see help `figure` in the Command window) after pausing 2 seconds to see the complete graph.

5. **Random walk in one dimension.** Write a program, `rw1d.m`, that generates and plots random walks in one dimension. Let the walker start at $x = 0$ and move a unit step to the right or left, depending on a random number generated with `r=rand`. If $r <= 0.5$, the walker steps one unit to the right; if $r > 0.5$, the walker steps one unit to the left. Its position after each step depends on the previous position and the value of r.

   The key data structure is an array `x`. Let `x(istep)` be the position of the walker after `(istep-1)` steps. Initialize the x array to N zeros (start with $N = 100$ and let it get bigger). Create an array of step numbers, `steps=(1:N);`. Use a `for` loop to calculate the position after each step.

   - Since the walker starts at the origin, $x(1) = 0$.
   - A step to the right means `x(istep)=x(istep-1)+1;`.
   - A step to the left means `x(istep)=x(istep-1)-1;`.
   - Calculate the entire `x` array without plotting. Then plot and animate the motion and path of the walker.

Write the program in well-structured blocks:

a. *Header*. File name, description, author.

b. *Set walk parameters*. (e.g., number of steps)

c. *Set program parameters*. (*x* array, steps array)

d. *Calculate walk*.

e. *Plot and animate walk*.

6. **Roulette.** Write a program, `roulette.m`, that simulates betting red (or black, or even, or odd) each time for *N* spins of a roulette wheel. The wheel has 38 positions: 18 red, 18 black, 0 and 00 (green). Describe the game using a vector `cash; cash(ispin)` is the amount of cash you have after `ispin-1` spins.
   - The probability of winning on any spin is 18/38 = 47.37%.
   - Start with `cash(1)=100` dollars.
   - Set a fixed amount to bet each time, `betAmount`. If you have at least that much cash left, bet `C` dollars on the spin.
   - Payout is 1:1, so if you win, you add `betAmount` to your cash; if you lose, subtract it.
   - You can't bet money you don't have. If you don't have `betAmount` dollars left, you must bet 0 until the game ends. (Or use a `while` loop to play only until you're broke.)

   Plot cash as a function of spin number. As in the previous problem, calculate all *N* spins first, then animate the results, plotting the current value of your cash and the history of your bankroll.

7. **Epicycles.** Write a program, `epicycles.m`, to draw circular motion with one epicycle, as illustrated in Figure 6.4.

   The object moves around the origin with a period T_1 and around the center of the epicycle with a period T_2. The radius of the primary orbit is R and of the epicycle is r. The angular frequency of each orbit is:

$$\omega_1 = \frac{2\pi}{T_1} \qquad \omega_2 = \frac{2\pi}{T_2}$$

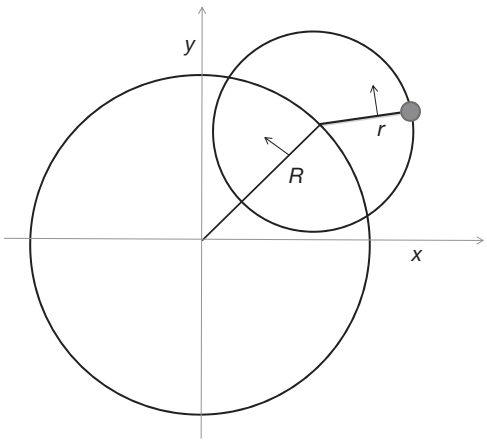

FIGURE 6.4 Epicycles.

The x and y coordinates of the motion are given by (convince yourself of this):

$$x(t) = r\cos(\omega_1 t) + r\cos(\omega_2 t)$$

$$y(t) = r\sin(\omega_1 t) + r\sin(\omega_2 t)$$

Let $R = 5$ and $r = 2$ initially. Let $T_1 = 1$ and $T_2 = T_1/$`tratio`. Start with `tratio = 9.25`. Let the number of large cycles be `Nc` (initially 10, say). Let the number of time points per cycle be `Ntpc` (initially 200). The total time is `Nc*T1` and the total number of times for which you calculate the position is `Nt=Nc*Ntpc`. Calculate the frequencies `omega1` and `omega2`. For each time in the interval `[0,Tf]` (say loop index `it` goes from 1 to `Nt`), calculate `x(it)` and `y(it)`. Plot the position of the object and a line tracing its path from the beginning to its present position.

a. Try various values of `tratio`. Try integers, simple rational numbers (e.g., 9.25) and non-simple numbers (e.g., 9.277).

b. Experiment with different r and R. What if $r > R$? Play around.

c. Add a logical switch, `doAnimation`, which lets you choose between an animation and a plot of the whole trajectory at once.

8. A random walk in two dimensions. Write a program, `rw2d.m`, to describe a walker making a random walk with unit steps in the x-y plane. The walker starts at the origin. Represent the walkers N positions using MATLAB arrays `x` and `y`. A good value to start with would be 1×10^3.

The walk starts with `x(1) = 0`, `y(1) = 0` (initialize both arrays to zero). Use a `for` loop to calculate the walker's moves. At each step the walker uses a random number to decide whether to take a step.

- North (y increases by 1; x stays the same).
- South (y decreases by 1; x stays the same).
- East (x increases by 1; y stays the same).
- West (x decreases by 1; y stays the same).

The walker always takes a step of length 1.

Notes:

a. Note that one can generate a random number in the set {1, 2, 3, 4} by using the command `idirection=randi(4);`.

b. Animate the plot by plotting at each step `istep`.

i. The starting position at the origin as an asterisk.

ii. The path the walker has taken up to now by a curve.

iii. The position of the walker at the present as an "o." E.g.,

```
plot(0, 0, 'r*', ...
       x(1:k), y(1:k), ...
       x(k), y(k), 'ro');
axis equal
draw now
```

c. Write the program in structures blocks labeled:

```
%% set walk parameters
...
%% set program parameters and
initialize arrays
...
%% calculate walk
...
%% plot walk
...
```

9. Vertical motion and energy. If an object is shot upward with an initial vertical velocity v_0, its height at any time t can be calculated by the equation:

$$h(t) = h_0 + v_0 t - \frac{1}{2} g t^2$$

where h_0 is the initial height of the object, and $g = 9.8\ m/s^2$, the acceleration due to gravity. The object's potential energy *PE* (its energy due to its position above the ground), kinetic energy *KE* (the energy associated with its movement) and total mechanical energy *TE* (potential energy + kinetic energy) can also be calculated at each time t, using the following equations:

$$\begin{aligned} PE &= mgh \\ KE &= 1/2mv^2 \\ TE &= PE + KE \end{aligned}$$

where m is the mass of the object, and v is the vertical velocity of the object. The vertical velocity of the object is the derivative of the height with respect to time, dh/dt.

Write a program, `verticalMotion.m`, for a 58 g tennis ball that is hit upward from an initial height of 0.8 m at a speed of 60 mph ($\approx 17.8\ m/s$), which computes for `Nt=500` timesteps and `Tfinal` = 4 s.

a. The height, h, of the ball.

b. The ball's vertical velocity, dh/dt.

c. Its potential energy (*PE*), kinetic energy (*KE*) and total energy (*TE*).

Using the `subplot` command (use `doc subplot` to get information on the `subplot` command) to plot and animate (with `Nstride=20`) the following three graphs:

a. The height, h, of the tennis ball.

b. The vertical velocity, dh/dt, of the ball vs. time.

c. The *PE*, *KE*, and *TE* of the ball vs. time. Use the `legend` command to create a legend that distinguishes the graphs from one another.

Each plot should be titled and each axis should be labeled.

Note: After the plot is complete, the legend can be dragged using the cursor to get a better view of the energy plot.

10. Two-dimensional billiards. Write a program `billard.m` that calculates and displays the motion of a point particle moving in two dimensions, x and y, with constant speed. The x-component of the velocity is v_x and the y-component is v_y. The velocities remain constant, so $v_x(t+\Delta t) = v_x(t)$ and $v_x(t+\Delta t) = v_x(t)$, unless there is a bounce off one of the walls at $x = 0$, $x = L_x$, $y = 0$, and $y = L_y$. Implement the bounce condition as follows:

- If $x(t+\Delta t) < 0$, flip the sign of $v_x(t+\Delta t)$ so that it is positive (moving to the right).
- If $x(t+\Delta t) > L_x$, flip the sign of $v_x(t+\Delta t)$ so that it is negative (moving to the left).
- If $y(t+\Delta t) < 0$, flip the sign of v_y so that it is positive (moving up).
- If $y(t+\Delta t) > L_y$, flip the sign of v_y so that it is negative (moving down).

Start the particle at $x = x_0$, $y = y_0$, $v_x = v_{x0}$, and $v_y = v_{y0}$. You will need to experiment with different time steps and simulation times.

a. Animate the path of the particle as a circle with a trailing line.

b. Set the playback speed using an animation stride, as in `AnimateEllipse.m`.

c. Change the program to accept an initial speed v_0 and an initial angle θ (with respect to the x-axis).

11. Handball (a two-dimensional billiard with gravity). Extend the previous program `billard.m` by adding a constant acceleration due to gravity, $a_y = -9.8$ m/s^2, $a_x = 0$. Call the new program `handball.m`. The equations of motion are given by equations (6.14) through (6.20). Good values for the parameters to start your explorations are $(x, y) = (0, 0)$, $(v_x, v_y) = (1.123, 3.8)$ m/s, `Tf=12` s, `Nt=12000`, and a stride value of 7.

a. As with the `billard.m` program, animate the motion with a trailing line.

b. Assume the mass of the particle is 0.05 kg. Calculate and plot, as functions of time, the kinetic energy, $\frac{1}{2}m(v_x^2 + v_y^2)$, the potential energy, mgy, and the total energy. The units of energy are joules.