

MCS 320 Project 3: A simple model of billiards

The goal of this project is to gain experience with functions and plots in Matlab to define a simple model of billiards. Despite the simplicity of our model, we will see trajectories normally not seen on a pool table.

To start this project download `pooltable.m`, `getCoord.m` and `evenBounces.m` from the course web-page.

1. Defining and visualizing trajectories

Imagine a square pool table. The length of each side is one. A billiard ball moving on the pool table starts at some position $\mathbf{p} = (x_0, y_0)$ and moves with constant speed $\mathbf{v} = (v_x, v_y)$.

The function `pooltable` contained in `pooltable.m` draws the table together with the trajectory of the ball in the time interval $[0, t]$. To do that it assumes that the ball moves in a straight line *modulo* the reflections from the boards of the table. The adjustment to coordinates is made by function `getCoord` contained in `getCoord.m`.

Make sure you understand how `getCoord` works.

Assignment One: Square pool tables are very rare. Modify the model to work with any rectangular pool table of dimensions a and b .

1. Create a function `recttable` that takes two additional arguments: the dimensions a and b . It should return the plot of a rectangular pool table with sides of length a and b together with the trajectory of the ball determined by these and the other parameters.
2. Test the function: `recttable([0.4,0.5],[0.09,0.015],1000,3,2)` should return the plot of a 3-by-2 rectangular pool table with the correct trajectory of the ball.
3. (extra 5 points) Given a and b , find all $\mathbf{v} = (v_x, v_y)$ such that the trajectory of the ball is cyclic, i.e. repeats itself starting at some point in time. Explain.

2. Computing special trajectories

On the 1-by-1 pool table, we start at $\mathbf{p}_0=(x_0, y_0)$ and wish to reach $\mathbf{p}_1=(x_1, y_1)$ at time $t = 1000$ hitting the vertical sides m times and the horizontal sides n times.

Let $\mathbf{p}_1=(0.4, 0.5)$, $\mathbf{p}_2=(0.8, 0.2)$, $m = 4$, $n = 4$. The function `evenBounces` produces the desired trajectory for this case.

This works only when m and n are both even.

Assignment Two: Write the function `oddBounces` that performs the task for the same data as above except with $m = 3$, $n = 5$. Make sure the ball ends up at the point \mathbf{p}_2 .

3. The deadline is Wednesday December 1, at 2PM.

Hand in the printout of the M-files containing the functions that you create, as well as the plots of the trajectories you have obtained for each assignment (hand-drawn sketches are acceptable).

To avoid any misunderstanding, this project must be solved *individually*. If you have questions, comments, or difficulties, feel free to come to my office for help.