Project 1: A Dilemma!

Project *Comp. Methods Physics* ASU PHY494 (2018)[*]

February 8, 2018 – Feb 22, 2018

Abstract You will investigate the decisions that a driver can make when braking for a traffic light that switches from green to yellow by writing code in Python to model the car's behavior for different decisions and parameters. You will write a short report to communicate, discuss and summarize your reasoning and your results.

Due Thursday, February 22, 2018, 11:59pm.

- Each student works on their own project.

- **Admissible Collaboration:** Students are allowed to talk to other students in the class about the project and exchange ideas and tips. However, sharing/copying reports or full code solutions is not allowed. **Help from other students must be acknowledged in an Acknowledgments section**. Direct help from outside the class is not allowed (except instructor/TA), e.g., you cannot ask for solutions (online or in person) but you can use books and the internet to solve problems.

- Each student should commit their *own* report (see Section 3) in **PDF** format to their own **GitHub repository**; alternatively, combining report and code in a Jupyter notebook is also possible as long as the notebook can be read like a report (i.e., not just bullet points or short comments).

- Each student should commit and push **all code** (see Section 4) that is required to reproduce the results in the report to their own **GitHub repository**. Include a text file `README.txt` that describes the commands to run calculations. The code must run in the standard anaconda-based environment used for the class. If it is a Jupyter notebook then it should be possible to *Kernel → Restart & Run All* and to produce all the required figures and output.

---

[*]Current version of this document: February 9, 2018. See Appendix C for a list of changes since v1 from Feb 06, 2018.

Grading will take the following into consideration:

- code runs and produces correct output

- report clearly and succinctly describes the question, approach, and results and contains sufficient evidence that the requirements (see below) have been met

- thorough attribution of code and help

- any additional work that you want to include in an appendix to the report or additional simulations for the main report will be treated as bonus material

1. Submission instructions

Submission is now to your **private GitHub repository**. Follow the link provided to you by the instructor in order for the repository to be set up: It will have the name *ASU-CompMethodsPhysics-PHY494/assignments-2018*-YourGitHubUsername and will only be visible to you and the instructor/TA. Follow the instructions below to submit this (and all future) homework.

Read the following instructions carefully. Ask if anything is unclear.

1. `git clone` your assignment repository (change *YourGitHubUsername* to your GitHub user name)

   ```
   repo="assignments-2018-YourGitHubUsername.git"
   git clone https://github.com/ASU-CompMethodsPhysics-PHY494/${repo}
   ```

   or, if you already have done so, `git pull` from within your assignments directory.

2. run the script `./scripts/update.sh` (replace *YourGitHubUsername* with your GitHub user name):

   ```
   cd ${repo}
   bash ./scripts/update.sh
   ```

   It should create three sub-directories[1] `project_1/Submission`, `project_1/Grade`, and `project_1/Work`.

3. You can try out code in the `project_1/Work` directory but you don't have to use it if you don't want to. Your grade with comments will appear in `project_1/Grade`.

4. Create your solution in `project_1/Submission`. Use Git to `git add` files and `git commit` changes.

---

[1]If the script fails, file an issue in the Issue Tracker for PHY494-assignments-skeleton and just create the directories manually.

You can create a PDF file or Jupyter notebook inside the `project_1/Submission` directory as well as Python code (if required). **Name your files `project1.pdf` or or `project1.ipynb`**, depending on how you format your work. Files with code (if requested) should be named exactly as required in the assignment.

5. When you are ready to submit your solution, do a final `git status` to check that you haven't forgotten anything, commit any uncommitted changes, and `git push` to your GitHub repository. Check on *your* GitHub repository web page[2] that your files were properly submitted.

    You can push more updates up until the deadline. Changes after the deadline will not be taken into account for grading.

Work must be legible and intelligible or may otherwise be returned ungraded with 0 points.

2. Problem description

A driver in a car approaches an intersection with constant speed $v_0 = 55$km/h. She sees the traffic light switch from green to yellow when she is at a distance x_0 from the intersection. She has to decide

- to brake (with deceleration $a = -3\,\text{m/s}^2$) or

- to continue driving with constant v_0.[3]

Any reaction is delayed by the driver's reaction time $\delta = 0.8\,\text{s}$. The yellow interval (time between green and red) is $\tau = 3\,\text{s}$ and the width of the intersection is $W = 45\,\text{m}$. You can ignore the length of the car. Take the entrance of the intersection and the position of the traffic light to be at $x = 0$ (see Figure 1).

Your task will be to determine if there are situations in which the driver will not be able to adhere to the traffic code, i.e., if she will either run a red light or not clear the intersection before the traffic light has switched to red. In particular, you should find out if there is a starting position (x_0^B in Figure 1) before which braking is safe (i.e., the car will stop before the traffic light) and if there is a starting position (x_0^A) beyond which crossing the intersection is safe (i.e., the car will cross the intersection before the light has switched from yellow to red).

Under certain conditions there can exist a range of values $x_0^B < x_0 < x_0^A$ where neither the decision to drive across the intersection nor to brake will be successful and therefore lead to a dangerous traffic situation. This range of values is called the *dilemma zone* (see the gray area in Figure 1). The size of the dilemma zone is

$$s := x_0^A - x_0^B \tag{1}$$

---

[2]`https://github.com/ASU-CompMethodsPhysics-PHY494/assignments-2018-`*`YourGitHubUsername`*
[3]In this problem the driver does not have the (legal) option to accelerate because she is already driving at the speed limit.
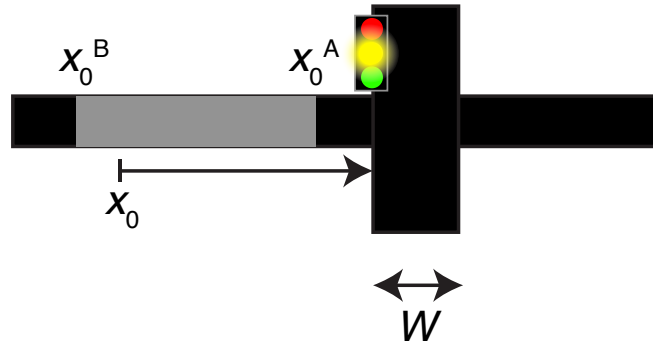
Figure 1. Geometry of the traffic situation. x_0 is the position of the car at time $t = 0$ when the traffic light at $x = 0$ switches to yellow. The width of the intersection is W. If $x_0 \geqslant x_0^A$ then the car will be able to cross the intersection before the light switches to red; if $x_0 \leqslant x_0^B$ then the car will be able to brake before the traffic light. The gray zone is the "*dilemma zone*".

and the dilemma zone exists when $s > 0$.

3. Report

Write a report in which you address the tasks in Section 5 below. The report should contain all results (figures, tables, equations). It must contain a title, author's name, sections *Background* (problem description, definitions), *Results and Discussion* (description and interpretation of results), and *Summary* (short summary of the main results). If you had any form of outside help you must describe it in an *Acknowledgments* section. If you use code or material from elsewhere you *must cite the source* (add a *References* section). Any bonus material can be shown in an optional *Appendix*.

The report must be written in full sentences and read as a coherent piece of work. Figures must have legends, labels, and captions. Type set in an 11pt font with single line spacing (captions, labels, legend may have smaller font sizes but must still be legible) and leave at least 1 in margins.

Overall, a length of about four pages is expected; the report should not be less than three or more than six pages long.

4. Code

For all numerical calculations use Python 3.x. You may use any of the Python packages that are part of the Anaconda 3 distribution such as `numpy` and `matplotlib`.

Include all the code that is needed to generate the results shown in your report. This can consist of Python programs, modules, a Jupyter notebook, or a mixture thereof. Include
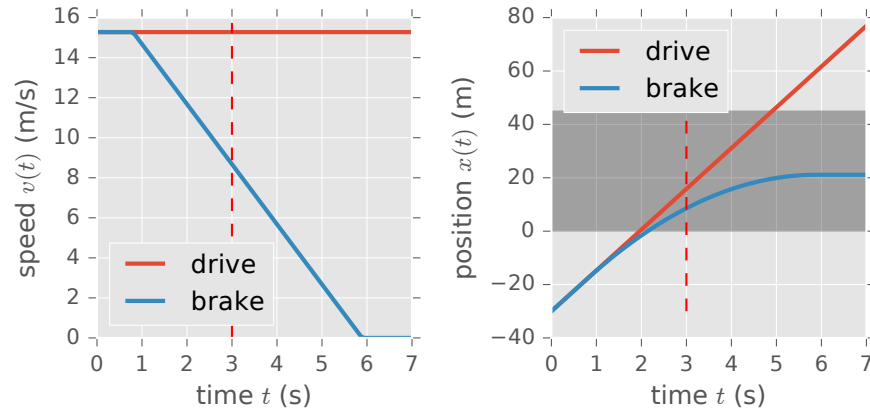
Figure 2. Example plot for time series $x(t)$ and $v(t)$ for $x_0 = -30$ m. The red dashed line indicates time τ and the shaded gray area indicates the extent of the crossing W.

a separate file `README.txt` that explains how to run your code in order to generate the results in your report. Your code must run in order for you to be awarded full marks.

5. Tasks

Address the following tasks in your report:

(a) Write down the equations of motions of the car, namely $v(t)$ and $x(t)$, depending on x_0 for both the decision to brake or to continue driving.[4]

(b) Use your equations of motions from (a) to define Python functions to compute the velocity $v(t)$ and position $x(t)$ for any t, depending on x_0 and for both the decision to brake or to continue driving.

Plot $v(t)$ and $x(t)$ for $0 \leqslant t \leqslant 7$ s in time steps of $\Delta t = 0.1$ s for $x_0 = -30$ m (which should look similar to Figure 2) as well as $x_0 = -70$ m and $x_0 = -0.5$ m. Plot graphs for *driving* and *braking* in the same figure. Indicate the time τ at which the traffic light switches to red and the extent of the intersection W in your figures.[5] For each figure explain whether a safe or a dangerous situation is occuring (see also (c)).

(c) Classify outcomes as either *safe* or *dangerous*, depending on x_0: For a given x_0, calculate the time series of the velocities $v_{\text{drive}}(t)$ and positions $x_{\text{drive}}(t)$ when the decision is made to attempt to *drive* across the intersection. Also calculate the time

---

[4]The Heaviside step function Eq. 2 and its Python implementation in Appendix A might be useful.

[5]See Appendix B for Python code to plot a dashed vertical line and a gray rectangle, similar to those shown in Figure 2.

5

series $v_{\text{break}}(t)$ and $x_{\text{break}}(t)$ for the decision to brake. Use a range of time points $0 \leqslant t_i \leqslant t_{\text{max}}$ and $\Delta t = 0.1\,\text{s}$. You need choose t_{max} sufficiently large to see the car to come to a stop or to cross the intersection.

Analyze the position trajectories $x(t)$ for driving/braking: The outcome is safe if

driving $x > W$ for $t > \tau$

braking $x < 0$ for all t

Otherwise it is dangerous. For a given x_0, the outcome is safe if there is at least one safe course of action.

Determine if the situation is safe for $x_0 = -70\,\text{m}$, $-30\,\text{m}$, and $-0.5\,\text{m}$.

(d) Map out the zone of values x_0 that lead to dangerous situations, i.e., that do not allow the driver to either brake or complete the crossing of the intersection in time.

Run the calculations for x_0 ranging from $-100\,\text{m}$ to the entrance of the intersection $x_0 = 0$ in $0.5\,\text{m}$ intervals and determine for each x_0 if allows for a safe decision or not.

(i) Plot the classification (e.g., `True` for *safe*, `False` for *dangerous*) against x_0. Show and discuss your plot.

(ii) Determine the numerical values of the dilemma zone x_0^A, x_0^B, and s (Eq. 1) from your data.

(e) Solve the whole problem analytically:

(i) Find general expressions for x_0^A, x_0^B, and s, which should only depend on x_0, v_0, δ, τ, and W.

(ii) Compute x_0^A, x_0^B, and s for the given parameters of the problem and compare to your simulation results.

(f) BONUS: [6] Determine the size of the dilemma zone as a function of the initial speed, $s(v_0)$, and plot it for $20\,\text{km/h} \leqslant v_0 \leqslant 100\,\text{km/h}$. In a second plot show $x_0^A(v_0)$ and $x_0^B(v_0)$ for the same range of v_0. You may solve this problem either numerically or analytically.

A. Heaviside step function

The Heaviside step function

$$\Theta(x) = \begin{cases} 1, & x > 0 \\ \frac{1}{2}, & x = 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

---

[6] You do not have to work on this problem and you can still get 100% of the points on the whole project but if you include results for the BONUS: problem then you can get additional bonus points.

can be computed with the Python function

```python
import numpy as np

def heaviside(x):
    """Heaviside step function.

    Arguments
    ---------
    x : scalar or array

    Returns
    -------
    float64 scalar or array

    From http://stackoverflow.com/a/15122658/334357
    """
    return 0.5 * (np.sign(x) + 1)
```

It has the advantage that it functions as a NumPy ufunc, i.e., it works equally well with scalar and array input.

B. Plotting

The following code can be used (perhaps with modifications) to add a vertical dashed line to a plot and to plot a gray filled area:

```python
import matplotlib.pyplot as plt

# plot dashed red line at tau
plt.plot([tau, tau], [x0, 1.05*W], "--", color="red", lw=1)

# plot gray area to indicate intersection
plt.fill_between([0, 7], [W, W], color="black", alpha=0.3)
```

C. History

Changes and updates to this document.

2018-02-06 initial version

2018-02-09 fixed unit of acceleration