Project 1: To Brake or Not to Brake?

Project *Comp. Methods Physics* ASU PHY432 (2025)*

February 13, 2025 – Feb 28, 2025

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, Feb 23, 2025, 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**. If you get help from a fellow student in the *PHY432 Discord* you *may acknowledge them with their screen @name*—you do *not* have to ask them for their real name. Direct help from outside the class

*Current version of this document: February 13, 2025. See Appendix C for a list of changes since v1.

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) to their own **GitHub repository**. You may combine report and code in a **Jupyter notebook** as long as the notebook can be read like a report (i.e., not just bullet points or short comments) *or* you may submit the report in **PDF** format and the code as Python files.
- 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**.
- With your code, 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 (and you should state this in the `README.txt`).
- The *Late Policy* for Midterms from the Syllabus applies.

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

The detailed **grading rubric** is available on Canvas as part of the assignment **Project 1** and you are encouraged to look at it.

1. Submission instructions

Submission is to a **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 *py4phy/project-1-2025*-YourGitHubUsername and will only be visible to you and the instructor/TA. Follow the instructions below to submit this project.

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

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

   ```
   repo="project-1-YEAR-YourGitHubUsername.git"
   git clone https://github.com/py4phy/${repo}
   ```

   or, if you already have done so, `git pull` from within your Project 1 assignment directory.

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

3. **Create your solution in `Submission`.** Use git to `git add` files and `git commit` changes.

   You can create a **PDF file** or **Jupyter notebook** inside the `Submission` directory as well as Python code (if required). **Name your files `project_1.pdf` or `project_1.ipynb`**, depending on how you format your work. Files with code must be included if they are required for running your solution.

4. **Make sure that your submitted solution runs** and **describe in `README.txt` *how* to run your code.**

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[1] 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 (except as stated in the *Late Policy*).

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 $a = -3\,\mathrm{m/s^2}$) or
- to continue driving with constant v_0.[2]

Any reaction is delayed by the driver's reaction time $\delta = 0.8\,\mathrm{s}$. The yellow interval (time between green and red) is $\tau = 3\,\mathrm{s}$ and the width of the intersection is $W = 30\,\mathrm{m}$. 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

[1] `https://github.com/py4phy/project-1-2025-`*`YourGitHubUsername`*

[2] In this problem the driver does not have the (legal) option to accelerate because she is already driving at the speed limit.

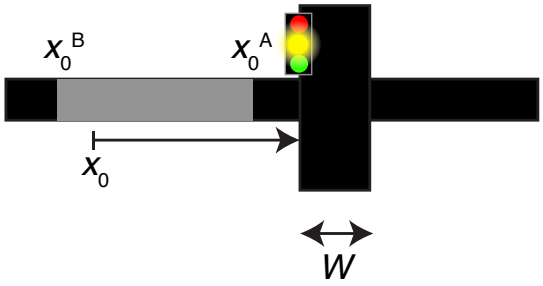

Figure 1. Geometry of the traffic situation. x_0 is the position of the car at time $t = 0$ when the traffic light at $x = 0$ switches to yellow. The width of the intersection is W. If $x_0 \geqslant x_0^A$ then the car will be able to cross the intersection before the light switches to red; if $x_0 \leqslant x_0^B$ then the car will be able to brake before the traffic light. The gray zone is the "*dilemma zone*".

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

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

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

3. Report

Write a report in which you address the tasks in Section 5 below.

3.1. Report structure

The report is a scholarly document and should read as a coherent whole, similar to a scientific publication. It should have the following structure:

- The report should contain all results (figures, tables, equations).
- It must contain a **title** and the **author's name**.
- It must contain the following **sections**

  Background motivation, problem description, definitions, equations to be solved, your approach to solving the problem

  Results and Discussion description and interpretation of results; you may use subheadings in this section for different results

  Summary short summary of the main results

  References Include a *References* section where you *cite the sources* for any code, material, or background-reading that you used; you should have at least one entry in the References to demonstrate that you can cite.

- If you had any form of outside help you must describe it in an *Acknowledgments* 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; axes must be labeled.

3.2. Submitting a written report (PDF)

If you write your report in a Word processor (Word, Google Docs, ...) or typesetting system (such as LaTeX) and produce a **PDF** then format it in the following way: Typeset 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. However, the page numbers are more like guidelines than actual rules and ultimately you will be graded on your report's content.

3.3. Submitting a Jupyter notebook

If you combine text and code in a **Jupyter notebook** then use standard Markdown formatting for text and headings/subheadings together with LaTeX for equations. You do not need to set specific font sizes or margins. Figures do not need to be numbered but should nevertheless contain text below that can be read as captions. The overall length of text and figures should be equivalent to a typed report but because of the inclusion of code no specific page number can be given. Just make sure to address all requirements of the problem in a clear and succinct manner.

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 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:

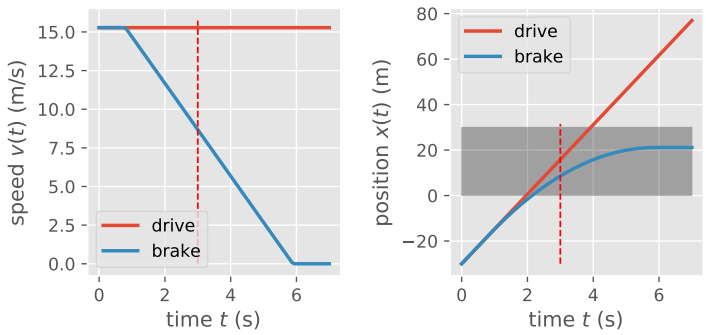

Figure 2. Example plot for time series $x(t)$ and $v(t)$ for $x_0 = -30\,\mathrm{m}$. The red dashed line indicates time τ and the shaded gray area indicates the extent of the crossing W.

(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.[3]

(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 $0 \leqslant t \leqslant 7\,\mathrm{s}$ in time steps of $\Delta t = 0.1\,\mathrm{s}$ for $x_0 = -30\,\mathrm{m}$ (which should look similar to Figure 2) as well as $x_0 = -70\,\mathrm{m}$ and $x_0 = -0.5\,\mathrm{m}$. 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.[4] Use your figures to *explain in words* for each situation whether a safe or a dangerous situation is occuring (see also (c)).

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

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

(c) *Computationally* classify outcomes as either *safe* or *dangerous*, depending on x_0, by considering the following steps:

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 series $v_{\text{break}}(t)$ and $x_{\text{break}}(t)$ for the decision to brake. Use a range of time points $0 \leqslant t_i \leqslant t_{\max}$ 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.

Use your computational approach to determine if the situation is safe for $x_0 = -70\,\text{m}$, $-30\,\text{m}$, and $-0.5\,\text{m}$.

Discuss if your algorithmic decision making (your situation classification algorithm) agrees with your analysis of your plots in (b) (the "human expert" analysis).

(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 $0.1\,\text{m}$ intervals and for time intervals with $\Delta t = 1 \times 10^{-3}\,\text{s}$. Determine for each x_0 if it allows for a safe decision or not.

(i) Plot the classification (e.g., `True` or 1 for *safe*, `False` or 0 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 to at least two decimals precision.[5] It is not sufficient to eyeball your plot to estimate the values although you can use the plot to check that you get the right values.

(e) Solve the whole problem analytically (i.e., find equations that directly provide an answer when you plug in x_0 and any other problem parameters):

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

Note: You should *not* use your analytical solutions in the solution of the preceeding problems (b)–(d); instead *compare* your numerical/computational solution to the analytical solution and discuss in your report how much they agree; discuss reasons for differences.

(f) BONUS: [6] Determine the size of the dilemma zone as a function of the initial speed, $s(v_0)$, and plot it for $20\,\mathrm{km/h} \leqslant v_0 \leqslant 100\,\mathrm{km/h}$. 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.

[5] These values depend on the interval at which you sample x_0 *and* the Δt interval at which you evaluate your trajectory. The smaller the two intervals are, the more accurate your numerical dilemma zone calculation should become.

[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 extra points although the overall total cannot exceed 100 points.

A. Heaviside step function

The Heaviside step function

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

can be computed with the Python function `numpy.heaviside(x, 0.5)`. 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:

```
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.

2025-02-13 initial version