

**UNIVERSITY OF VIRGINIA**  
**CS 6501 LEARNING IN ROBOTICS**  
**SPRING 2024**  
**HOMEWORK 0 - NOT GRADED**  
**DUE: 01/25 THU 11.59 PM**

---

---

**Instructions**

Read the following instructions carefully before beginning to work on the homework.

- You will submit solutions typeset in  $\text{\LaTeX}$  on Gradescope (strongly encouraged). You can use `hw_template.tex` on Collab in the Resources » Homeworks folder to do so. If your handwriting is *unambiguously legible*, you can submit PDF scans/tablet-created PDFs.
- Please start a new problem on a fresh page and mark all the pages corresponding to each problem. Failure to do so may result in your work not graded completely.
- Clearly indicate the name and UVA email ID of all your collaborators on your submitted solutions.
- For each problem in the homework, you should mention the total amount of time you spent on it.
- You can be informal while typesetting the solutions, e.g., if you want to draw a picture feel free to draw it on paper clearly, click a picture and include it in your solution. Do not spend undue time on typesetting solutions.
- You will see an entry of the form “HW 0 PDF” where you will upload the PDF of your solutions. You will also see entries like “HW 0 Problem 1 Code” where you will upload your solution for the respective problems. **For each programming problem, you should create a fresh Python file.** This file should contain all the code to reproduce the results of the problem and you will upload the `.py` file to Gradescope. If we have installed Autograder for a particular problem, you will use the Autograder. Name your file to be the same filename as stated in the respective problem statement.
- **You should include all the relevant plots in the PDF, without doing so you will not get full credit.** You can, for instance, export your Jupyter

notebook as a PDF (you can also use text cells to write your solutions) and export the same notebook as a Python file to upload your code.

- **Your PDF solutions should be completely self-contained. We will run the Python file to check if your solution reproduces the results in the PDF.**

**Credit** The points for the problems add up to 120. You only need to solve for 100 points to get full credit, i.e., your final score will be  $\min(\text{your total points}, 100)$ .

---

1 **Problem 1 (15 points).** Suppose  $X \sim N(\mu_1, \sigma_1^2 I)$  and  $Y \sim N(\mu_2, \sigma_2^2 I)$  are  
 2 two independent Gaussian random variables;  $\mu_1, \mu_2 \in \mathbb{R}^n$  are the means and  
 3  $\sigma_1 I, \sigma_2 I \in \mathbb{R}^{n \times n}$  are diagonal covariance matrices. Compute the distribution of  
 4  $X + Y$ .

5 **Problem 2 (10 points).** Let us imagine a robot that would like to go into a room.  
 6 The door to the room has two possible states: **open** and **closed**. We will represent  
 7 these states using a discrete-valued random variable  $X$

$$X = \begin{cases} 0 & \text{if door is open} \\ 1 & \text{if door is closed.} \end{cases}$$

8 We will assume that initially we have no knowledge of the door state, that is,  
 9  $P(X = 1) = P(X = 0) = 0.5$ . The robot has a sensor to detect the state of the  
 10 door which we will model using another discrete-valued random variable  $Y$ ; the  
 11 reading  $Y = 1$  indicates that the door is closed and vice-versa. We can think of the  
 12 value of  $Y$  as an observation for the state of the door, i.e., the value of  $X$ . However,  
 13 sensors are often erroneous and this observation is not always correct. We have

$$P(Y = 1 | X = 1) = 0.8$$

$$P(Y = 1 | X = 0) = 0.2.$$

14 Use the Bayes rule to compute the probability that is the door is open when the  
 15 sensor detects that the door is open. How does your answer change if you take  
 16 multiple measurements?

17 **Problem 3 (15 points).** This problem is an exercise in linear dynamical systems.  
 18 Given a state  $x(t) \in \mathbb{R}^n$  and a control input  $u(t) \in \mathbb{R}^p$  a linear dynamical systems  
 19 evolves using the equation

$$x(t + 1) = Ax(t) + Bu(t) + \xi(t)$$

20 where  $x(t + 1)$  is the state of the system at the next time-step, the matrix  $A \in \mathbb{R}^{n \times n}$   
 21 is the state-evolution matrix and the matrix  $B \in \mathbb{R}^{n \times p}$  is the control matrix. The  
 22 variable  $\mathbb{R}^n \ni \xi(t) \sim N(0, \Sigma)$  is the unmodeled part of the dynamics which we  
 23 can think of as zero-mean Gaussian noise with a symmetric covariance  $\Sigma \in \mathbb{R}^{n \times n}$ .  
 24 Suppose  $x(0) \sim N(0, I)$  and we pick a certain control input  $u(0)$  argue why the  
 25 probability distribution of  $x(1)$  is also a Gaussian. Compute the mean and variance  
 26 of  $x(1)$ . It is known that if the controller  $u(t)$  stabilizes the system (what does this  
 27 mean?) and all eigenvalues of  $A$  are smaller than 1 in magnitude, the variance of  
 28  $x(t)$  reaches a non-degenerate steady-state as  $t \rightarrow \infty$ , compute this variance. Can  
 29 you argue as to why the variance of  $x(1)$  or  $x(t)$  does not seem to depend on  $u(t)$ ?

30 **Problem 4 (30 points).** This problem will take you through the basics of using a  
 31 Python package manager named Miniconda, Google Colab and installing and using  
 32 a deep learning library named PyTorch. There are a number of ways to install these  
 33 on your own system and you are free to use whatever setup you prefer. However,  
 34 we recommend, installation using the conda package manager.

35 1. If you don't have a working Python programming setup on your laptop yet,  
36 follow the instructions at <https://docs.conda.io/en/latest/miniconda.html> to  
37 install and run Miniconda. Familiarize yourself with Jupyter (or IPython)  
38 using <https://realpython.com/jupyter-notebook-introduction>. Run `conda`  
39 `install numpy pandas matplotlib pytorch jupyter spyder`.  
40 Now open a Python terminal (`python3`) and verify that

```
41 import torch  
42 import numpy as np  
43 import matplotlib.pyplot as plt  
44
```

46 executes without issue. The spyder IDE will also now be installed if you  
47 prefer as well as jupyter. We recommend spyder.

48 2. The next bit of infrastructure we would like to introduce is Google Colab  
49 (<https://colab.research.google.com>). Google Colab is a free to use tool  
50 which gives access to two CPU cores, about 12 GB RAM and one (very  
51 good) GPU for 12 contiguous hours, through a Jupyter notebook. You  
52 should be able to complete most of the homeworks on your laptop with  
53 Anaconda above. You can certainly use Colab if you wish.

54 3. Use the Github repository at <https://github.com/jakevdp/PythonDataScienceHandbook>  
55 to brush up on Numpy (02.02), Pandas (03.00) and plotting using Matplotlib  
56 (04.00).

57 4. PyTorch is already installed on Google Colab, and it should have been  
58 installed through conda in step (1). PyTorch is very similar to Numpy in its  
59 functionality except that it is tailored to deep neural networks. You can fol-  
60 low the tutorial at [https://pytorch.org/tutorials/beginner/deep\\_learning\\_60min\\_blitz.html](https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html)  
61 to learn more. We will provide more material on how to use PyTorch when  
62 we get to the Reinforcement Learning part of this course.

63 **Problem 5 (20 points).** Jarvis likes to bet on coin tosses; he bets a dollar each  
64 time that the coin will come up heads. Jarvis begins with  $m$  dollars and quits if he  
65 either loses all the money or ends up with  $n$  dollars. The coin comes up heads with  
66 probability  $p < 1/2$ . Let  $q = 1 - p$  and  $B_k$  be the event that Jarvis is betting on the  
67  $k^{\text{th}}$  toss, and let  $X_k$  be the money left after  $k^{\text{th}}$  coin toss. If  $X_0 = m$ ,

- 68 (i) what is the probability the Jarvis loses all the money?  
69 (ii) what is the expected number of bets?

70 **Problem 6 (30 points).** This problem will teach you how to use numpy/pytorch/  
71 matplotlib efficiently. This problem can be done on your laptop.

72 1. **Programming Problem 1:** A key motivation for libraries like numpy and  
73 pytorch is that they have highly optimized functions for vectorization. Be-  
74 cause python is an interpreted language, `for` loops are extremely slow. If  
75 a `for` loop is independent—that is, the iterations can be executed in any  
76 order—then often the loop can be vectorized to run all iterations simultana-  
77 neously.

78 Consider an affine system  $x(t + 1) = Ax(t) + b$ , where  $A \in \mathbb{R}^{n \times n}$ ,  $b \in$   
79  $\mathbb{R}^n$ , and  $x \in \mathbb{R}^n$ . It is often desirable to simulate a system for many different  
80 initial conditions, for instance when training RL models or in a particle  
81 filter, as we will discuss later. Complete the function `sim_systems` in  
82 `hw0_solution.py` to find  $x(t + 1)$  for an arbitrary number of initial  
83 conditions. Pay attention to the structure and dimensionality of inputs and  
84 outputs.

85 **Hint:** You may find a discussion of *broadcasting* in python to be helpful.

86 2. **Programming Problem 2:** Complete the function `compute_derivative`  
87 in `hw0_solution.py` to take the partial derivatives of N-dimensional  
88 functions along each dimension. Recall that for  $f(x_1, x_2, \dots)$ , the partials  
89 are approximated for the discrete case by

$$\left. \frac{\partial f}{\partial x_i} \right|_{(c_1, c_2, \dots)} \approx \frac{f(c_1, c_2, \dots, c_i + \epsilon, \dots) - f(c_1, c_2, \dots, c_i, \dots)}{\epsilon}$$

90 Plot the derivative of a 1D function, and the  $x$  and  $y$  partials for a 2D  
91 function (use `plt.imshow`) of your choice. Pick functions that demon-  
92 strate the functionality of your implementation. Include these figures with  
93 your written answers.

94 3. Submit `hw0_solution.py` to the autograder. Upload `hw0_solution.py`  
95 and all supporting files to gradescope under **Homework 0 - Code**. The  
96 autograder may take several minutes, after which you will receive your  
97 score. You may submit multiple times, but we **strongly** encourage you to  
98 come up with your own local test cases for debugging.

99 **Some notes about the autograder:** The autograder will run your code  
100 through a number of test cases. There are 2 types of test cases: binary  
101 ones, and those providing partial credit. For binary cases, you will either  
102 receive all or none of the points for that case. For partial credit, there are 2  
103 thresholds at 100% and 60%. The autograder will tell you these thresholds  
104 and your score. If you score worse than the 60% threshold, you will receive  
105 no credit for that test case. If you score better than the 100% threshold, you  
106 will receive full credit. If you score in between, your credit will be scaled  
107 linearly between 60% and 100%.

108 For instance autograder feedback could look like the following:

```
109 Problem 1: Execution Time (sec) (4.33/5)
110 score: 2
111 100%: 0.5, 60%: 5.
```

112 Here, the student's test took 2 sec. This is 67% of the way to the 100%  
113 threshold, so the final score is 4.33/5. This system is intended to reward  
114 code which works reasonably well, but also reward and encourage you to

115 get things to work as well as they can (we promise, it's possible to get 100%  
116 on all test cases).