Learning for Adaptive and Reactive Robot Control
Instructions for exercises of lecture 7

Professor: Aude Billard
Assistants: Harshit Khurana,
Lukas Huber and Yang Liu
Contacts:
aude.billard@epfl.ch, harshit.khurana@epfl.ch,
lukas.huber@epfl.ch, yang.liuu@epfl.ch

Spring Semester 2022

Introduction

INTRO

This part of the course follows *exercises* 9.1 *to* 9.3 and *programming exercise* 9.1 of the book "*Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach. MIT Press, 2022*".

1 Theoretical exercises [1h]

Consider the nominal DS $f(x) = Ax + b$ with $A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$ and $b = [-1, 1]^T$. We then construct a modulation to add a circular obstacle as follows:

$$\dot{x} = \mathbf{M}(x) f(x) \quad \text{with} \quad \mathbf{M}(x) = \mathbf{E}(x)\mathbf{D}(x)\mathbf{E}(x)^{-1}, \tag{1}$$

where $M(x)$ is build through eigenvalue decomposition, using as the basis of the eigenvectors the normal $n(x)$ and tangents to the obstacle :

$$\mathbf{E}(x) = [\mathbf{n}(x)\ \mathbf{e}_1(x)]\,,$$

$$\mathbf{D}(x) = \mathbf{diag}\left(\lambda_n(x), \lambda_e(x)\right),$$

where the tangent $\mathbf{e}(x)$ forms an orthonormal basis to the gradient of the distance function $d\Gamma(x)/dx$.

We then set the eigenvalues to cancel the flow in the normal direction and increase the flow in the tangent direction as it reaches the boundary of the obstacle:

$$\lambda_n(x) = 1 - \frac{1}{\Gamma(x)} \quad \text{and} \quad \lambda_e(x) = 1 + \frac{1}{\Gamma(x)}$$

With $\Gamma(x)$ our distance function constructed so that :

- $\Gamma(x) > 1$ outside the obstacle
- $\Gamma(x) = 1$ at the obstacle boundary

- $\Gamma(x) < 1$ inside the obstacle

Since we want to model a circular obstacle, we first recall the circle equation :

$$d(x, x_o)^2 = r^2,$$

where x_o is the center of the circle, $d(x, x_o)$ is the euclidean distance between x and x_o and r is the radius of the circle. We can implicitly embed the circle equation into $\Gamma(x)$ to create a circular obstacle, while still keeping the properties cited above by defining $\Gamma(x)$ as :

$$\Gamma(x) = d(x, x_o)^2 - r^2 + 1.$$

This definition ensures $\Gamma(x)$ follows the prerequisites to implement obstacle avoidance in our DS.

1.1

The modulation generated by the eigenvalues results in a change of magnitude along the various basis directions, as described earlier. The increase in velocity along the tangent direction is bounded.

- What is its upper bound?
- Where does it occur?

1.2

In certain scenarios, the conservation of the magnitude of the velocity is required.

1. How can you change the eigenvalues such that the modulated velocity is always slower than the initial velocity, i.e., $\|\dot{x}\| \leq \mathrm{f}(x)$?
2. How can you change the eigenvalues such that the modulated velocity is always faster than the initial velocity, i.e., $\|\dot{x}\| \geq \mathrm{f}(x)$?
3. Can you set the eigenvalues such that the norm of the velocity is preserved around the obstacle?

1.3

How do you need to modify the modulation in Equation (1), such that it is evaluated with respect to a linearly, moving obstacle? Assume that we know the obstacle's linear velocity v^o.

1.4

(Optional) Use the obstacle avoidance formulation to create a dynamical system which locally converges towards a local spurious attractor, while remaining stable at the origin? (See Figure 1)

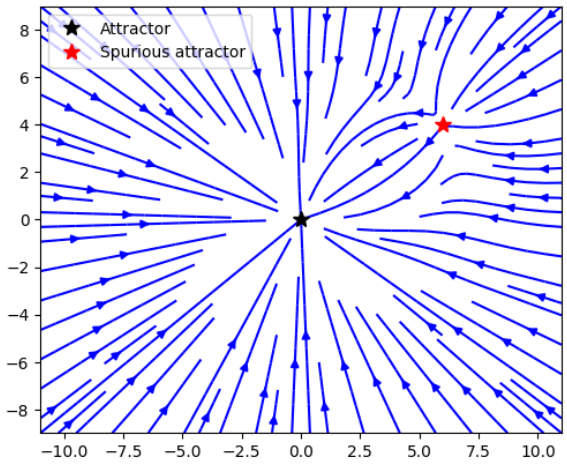

Figure 1: Dynamical system with spurious attractor

2 Programming exercises [1h]

Preliminaries: Instructions for proper installation of MATLAB libraries

Download the folder `lecture7-obstacle-avoidance` and place it next to the existing folders. Your folder `matlab_exercises` should now have the following structure:

- matlab_exercises
  - lecture1-introduction
    - exercises
    - solutions
  - lecture2-learning-from-demonstration
    - exercises
    - solutions
  - lecture3-intro-to-dynamical-systems
    - exercises
    - solutions
  - lecture4-learning-control-laws
    - exercises
    - setup_code_ch3.m
  - lecture6-modulating-ds
    - exercises
    - solutions
    - libraries
  - lecture7-obstacle-avoidance
    - exercises
  - libraries

2.1 Programming Exercise: Obstacle avoidance

Book correspondence: Programming Ex9.1, p.248

The aim of this exercise is to help readers to get better acquainted with the obstacle avoidance algorithm and the effect of the open parameters on obstacle modulation. Open MATLAB and

set the directory to the following folder:

```
lecture7-obstacle-avoidance/exercises
```

In this folder, you will find three files, one for each question. These files allow you to change the parameters of the modulation function and to generate a local modulation to create a simple obstacle.

1. Open file `lect7_ex1_1.m`
   (a) Change the modulation such that the norm of the velocity is preserved when going around the obstacle.
   (b) Set the eigenvalues to let the robot slow down or accelerate when moving around obstacles.
2. Open file `lect7_ex1_2.m`.
   (a) Change the shape of the obstacle to create an ellipse by modifying the variables `x_object` and `y_object`. Update the `gamma` function and its gradient `gradient_gamma` using the implicit equation of a non-tilted ellipse.
   Warning : Don't forget about the properties of the gamma function
   (b) What happens if you do not set a reference point inside the obstacle, nor use the reference direction in the modulation matrix ? Fix that issue.
3. Open file `lect7_ex1_3.m`.
   - Set a linear DS to make the obstacle move.
   - Change the code to generate the flow over time of the DS and observe the effect on the modulation over time.

   Hint : To visualize the obstacle moving, you need to loop over the implementation of the obstacle and the plot of the modulated DS and update the position of the obstacle at each iteration.

2.2 Programming Exercise: Spurious attractor (*Optional*)

Keep your directory in the following folder:

```
lecture7-obstacle-avoidance/exercises
```

Open `lect7_ex2.m`. This file generates a linear nominal DS, as proposed in last week's exercises. You will now implement a local spurious attractor, using the obstacle avoidance formulation.

IMPORTANT : Do not forget to uncomment the `plot_ds()` function call after the loop to plot your modulations.

References

[1] Aude Billard, Sina Mirrazavi, and Nadia Figueroa. *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. MIT press, 2022.