SC42120 - Adaptive Control (2018/19 Q3)

Final Assignment

February 14, 2019

This project is a compulsory part of the course Adaptive and Predictive Control. It will be graded, and will count for 50% of the final grade of the course.

This project has to be carried out in groups of 3 people, arranged by the teaching staff. It must be worked out in the form of a report in English (in a single PDF file), to be delivered along with the corresponding MATLAB/Simulink software (in a single ZIP file). The report and the software must be sent via e-mail to Professor Simone Baldi (**s.baldi@tudelft.nl**) and in carbon copy (cc) to the TAs Arun (**A.G.Thomas@student.tudelft.nl**) and Vittorio (**V.Giammarino@student.tudelft.nl**).

The deadline for submission (via e-mail) is **April 9, 2019 at 23:59**. It is enough that one member of the group sends the e-mail, with the other members in cc. Please mention your student numbers in your e-mail.

Two lectures will be given in order to explain the basic concepts needed to solve this assignment:

- 11th of March - 45 minutes lecture on the first section of this assignment;
- 14th of March - 45 minutes lecture on the second section of this assignment.

MATLAB/Simulink: the students will have to use MATLAB and Simulink for the final assignment. The solutions of these exercises were tested using **ode23tb** as solver, setting the **max step size set as auto** , with **adaptive zero-crossing algorithm**. Check your Simulink *Model Configuration Parameters* and select the same options.

Theoretical explanations and rigorous mathematical arguments are highly appreciated. Moreover, always explain your design choices (e.g. when you tune a design parameter).

1 Formation of UAVs

The assignment is based on the paper "Adaptive hierarchical formation control for uncertain Euler–Lagrange systems using distributed inverse dynamics"[1]. Thus reading the paper to solve the assignment is highly recommended. The expectation of the assignment is that all the (uncertain) UAV agents should follow the given reference trajectory using the adaptive laws. The reference trajectory is a Circular Orbit centered at (x=250,y=500) in inertial plane with a radius of 50units.

1.1 Background Information

1.1.1 Frame of References

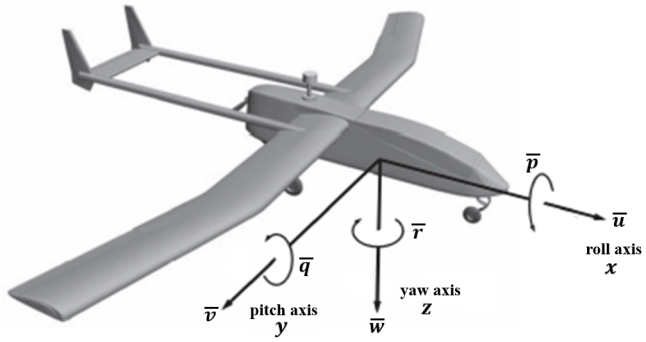

Figure 1: Fixed-Wing UAV body Frame.

Let us start by defining the frame of references. For the purpose of solving the assignment, it is important to understand three frames of references. They are:

1. *The inertial frame* $\mathscr{F}^i$: is an Earth-fixed frame whose unit vectors are $\mathbf{i}^i$, $\mathbf{j}^i$, and $\mathbf{k}^i$ directed towards north, east, and down respectively (NED convention).

2. *The vehicle frame* $\mathscr{F}^v$: represents the inertial frame translated onto the center of mass of the vehicle.

3. *The body frame* $\mathscr{F}^b$: is defined as shown in the Figure 1. The Euler angles ψ, θ, and ϕ (yaw, pitch and roll) about the axes $\mathbf{k}^i$,$\mathbf{j}^{v1}$, and $\mathbf{i}^{v2}$ respectively($\mathscr{F}^{v1}$ and $\mathscr{F}^{v2}$ are the intermediate vehicle frames when Euler rotations occurs from inertial to body frame). The rotation matrix from the vehicle frame (also inertial frame) to the body frame can be written as:

$$\mathcal{R}_v^b(\phi,\theta,\psi) = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \tag{1}$$

1.1.2 Ground Speed $\mathbf{V}_g$

The ground speed $\mathbf{V}_g$ relative to $\mathscr{F}^i$ is different from the Airspeed $\mathbf{V}_a$ in presence of wind (with wind velocity $\mathbf{V}_w$). They are related by the so-called wind triangle. The relation in vector sense is:

$$\mathbf{V}_g = \mathbf{V}_a + \mathbf{V}_w \tag{2}$$

In windy conditions, since wind speed is unknown, tracing a given trajectory is difficult by ordinary control. Thus, an estimation of $\mathbf{V}_g$, i.e. $\hat{V}_g$ is used to compute a reference trajectory. Students are not expected to implement this, and the required SIMULINK implementation is made available with the assignment.

1.1.3 Euler-Lagrange Systems

For the assignment, we are considering the UAVs to be modelled as Euler-Lagrange(EL) systems. The dynamics of a network of Euler-Lagrange agents can be described by

$$D_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i, \quad i = \{1, ..., N\} \tag{3}$$

where the term $D_i(q_i)\ddot{q}_i$ is proportional to the second derivatives of the generalized coordinates, the term $C_i(q_i, \dot{q}_i)\dot{q}_i$ is the vector of centrifugal/Coriolis forces, proportional to the first derivatives of the generalized coordinates, and the term $g_i(q_i)$ is the vector of potential forces. Finally, the term τ_i represents the external force applied to the system.

1.1.4 Inverse Dynamic Based Control

Inverse dynamic based control is a typical control method for EL systems, which is here recalled for completeness. Given the dynamics (3), the objective of inverse dynamic based control is to cancel all the non-linearities in the system and introduce simple PD control so that the closed-loop system is linear. In the known parameter case, the cancellation of non-linearities can be achieved via the controller

$$\tau_i = D_i(q_i)a_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) \tag{4}$$

where the term a_i is defined as

$$a_i = \ddot{q}^d - K_v\dot{e}_i - K_p e_i \tag{5}$$

with $e_i = q_i - q^d$ and K_p, K_v being the proportional and derivative gains of the (multivariable) PD controller. Note that q^d, $\dot{q}^d$,and $\ddot{q}^d$ are desired trajectories, velocities, and accelerations to be specified by the user. Substituting (4) into (3) gives us

$$\begin{aligned} D_i(q_i)(\ddot{q}_i - \ddot{q}^d + K_v\dot{e}_i + K_p e_i) &= 0 \\ \ddot{e}_i + K_v\dot{e}_i + K_p e_i &= 0. \end{aligned} \tag{6}$$

The resulting second-order error equation can be written as

$$\begin{bmatrix} \dot{e}_i \\ \ddot{e}_i \end{bmatrix} = \begin{bmatrix} 0 & \mathbb{1} \\ -K_p & -K_v \end{bmatrix} \begin{bmatrix} e_i \\ \dot{e}_i \end{bmatrix} \tag{7}$$

or equivalently,

$$\begin{bmatrix} \dot{q}_i \\ \ddot{q}_i \end{bmatrix} = \begin{bmatrix} 0 & \mathbb{1} \\ -K_p & -K_v \end{bmatrix} \begin{bmatrix} q_i \\ \dot{q}_i \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbb{1} \end{bmatrix} (\ddot{q}^d + K_v\dot{q}^d + K_p q^d). \tag{8}$$

1.2 EL Agents & Communication Graph

For the assignment, we consider a system of four EL agents following a reference trajectory in a particular formation. The graph depicts the allowed information flow between the agents (nodes). Each node of the graph is an agent.

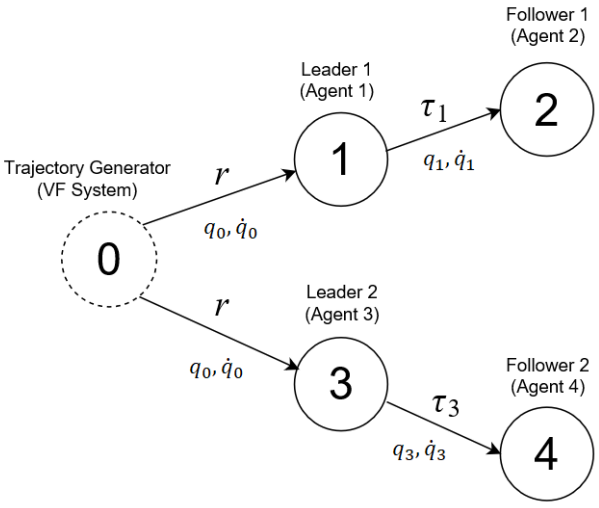

Figure 2: Communication graph $\mathscr{G}$.

In the given graph, the Node 0 (called pinner) generates a trajectory (The SIMULINK implementation for Node 0 is made available with the assignment and students are not expected to implement it). Node 0 doesn't receive any information from any other agents in the network. Other agents needs to follow this pinner in the formation, respecting the allowed information flows.

Leader 1 and Leader 2 (Node 1 & Node 3 respectively) can only receive from Node 0. Follower 1(Node 2) can only receive information from Leader 1. Similarly, Follower 2(Node 4) can only receive information from Leader 2.

This graph $\mathscr{G}$ (pinner excluded) is described by the following adjacency matrix:

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

where each element a_{ij} of the matrix $\mathcal{A}$ is obtained as explained during the lecture. The node zero (pinner) has linearised reference dynamics as defined in

$$\begin{bmatrix} \dot{q}_0 \\ \ddot{q}_0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \mathbb{1} \\ -K_p & -K_v \end{bmatrix}}_{A_m} \underbrace{\begin{bmatrix} q_0 \\ \dot{q}_0 \end{bmatrix}}_{x_m} + \underbrace{\begin{bmatrix} 0 \\ \mathbb{1} \end{bmatrix}}_{B_m} r \tag{9}$$

where q_0, $\dot{q}_0 \in R^n$ is the state of the reference model and $r = \ddot{q}^d + K_v\dot{q}^d + K_p q^d$ is a user-specified reference input. The reference dynamics (9) basically represent some homogeneous dynamics all agents should synchronize to. The other agents are described by the EL dynamics given in equation (3) .i.e.,

$$D_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i, \quad i = \{1, 2, 3, 4\} \tag{10}$$

The parameters for each agent is as given in Table 1.

Table 1: Fixed-wing UAVs parameters and initial conditions

	Mass (kg)	Initial cond. $[x, y, z, \phi, \theta, \psi]'(0)$	Initial cond. $[\bar{u}, \bar{v}, \bar{w}, \bar{p}, \bar{q}, \bar{r}]'(0)$	Moment of Inertia (kg m^2)
Agent 1 (Leader 1)	20	[-50,750,-75,0.5,0.05,0.5]'	[25,0,0,0,0,0]'	$\begin{bmatrix} 0.1 & 0 & -0.01 \\ 0 & 0.05 & 0 \\ -0.01 & 0 & 0.1 \end{bmatrix}$
Agent 2 (Follower 1)	30	[-100,-1000,-100,1,0.1,1]'	[5,0,0,0,0,0]'	$\begin{bmatrix} 0.2 & 0 & -0.02 \\ 0 & 0.1 & 0 \\ -0.02 & 0 & 0.2 \end{bmatrix}$
Agent 3 (Leader 2)	40	[-50,250,-150,-0.5,-0.05,-0.5]'	[20,0,0,0,0,0]'	$\begin{bmatrix} 0.4 & 0 & -0.04 \\ 0 & 0.2 & 0 \\ -0.04 & 0 & 0.4 \end{bmatrix}$
Agent 4 (Follower 2)	50	[-100,0,-200,-1,-0.1,-1]'	[10,0,0,0,0,0]'	$\begin{bmatrix} 0.8 & 0 & -0.08 \\ 0 & 0.4 & 0 \\ -0.08 & 0 & 0.8 \end{bmatrix}$

1.2.1 Task 1 - Synchronisation in Known Parameters Case

For this task, assume that all the parameters are known for generating the control actions needed to synchronise the agent with the reference.

Leader 1(Node 1) Synchronisation:

- Configure SIMULINK 6DOF Block with the appropriate parameter values.
- Construct D_1, C_1 and g_1 matrices combining the output signals and parameter values.
- Comment on the ideal control law for leader.
- Implement the ideal control law.

Follower 1(Node 2) Synchronisation:

- Configure SIMULINK 6DOF Block with the appropriate parameter values.
- Construct D_2, C_2 and g_2 matrices combining the output signals and parameter values.
- Comment on the ideal control law for follower.
- Implement the ideal control law.

Repeat the same exercises for **Leader 2(Node 3) and Follower 2 (Node 4) Synchronisation** with appropriate parameter values.

- Show the plots and write results.
- Comment on the advantages and disadvantages of the above approach.

1.2.2 Task 2 - Synchronisation in Unknown Parameters Case

For this task, none of the parameters given are known for generating the control actions.

- Implement the synchronization with adaptive laws for all the agents.
- Show the plots and write results.
- Consider the UAVs have a V formation in the inertial plane, as shown in Figure 3. Implement the formation.

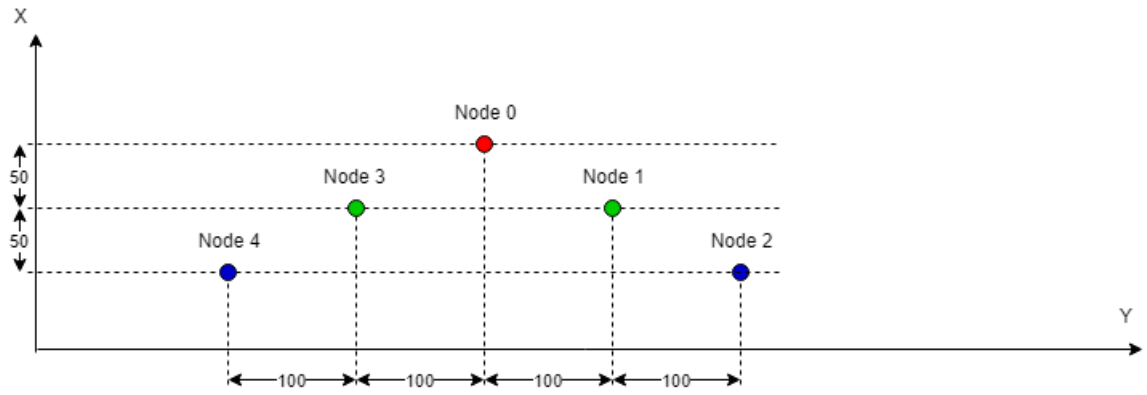

Figure 3: UAVs in V formation

- Show the plots and write results.

1.3 Notes

- Make sure that you have Aerospace Blockset in your MATLAB.
- Always be sure that you run "SC42120_UAV_Preconditions.m" before running the SIMULINK Block with Node 0. Please be careful to not tamper with any of the workspace variables defined by the implementation. The SIMULINK model is provided in "SC42120_UAV_SIMULINK.slx".
- The supplied files runs best in MATLAB 2017b on Windows.

- For solving the assignment, you can assume that the control actions (Forces and Moments) generated can be directly applied to the UAVs i.e. there is no Control Allocation.
- You need to be careful with the frame of references.
- The simulation for full adaptive synchronisation takes 30-45 minutes to execute. Depending on implementation and PC hardware it may vary.
- SIMULINK 6DOF block doesn't consider the action of gravitational force. So you need to add action of gravity in addition to your control actions. You can see the implementation inside 6DOF Block by looking under the mask of the same.

1.4 Submission

- Write-up and plots in report.
- MATLAB AND SIMULINK files expected along with supplied code:

Task 1	'UAV_sync_known_sim.slx' and 'UAV_sync_known.m'
Task 2 (without gap)	'UAV_sync_unknown_sim.slx' and 'UAV_sync_unknown.m'
Task 2 (with gap)	'UAV_sync_unknown_gap_sim.slx' and 'UAV_sync_unknown_gap.m'

2 Optimal Switching and Control of Nonlinear Switching Systems Using Approximate Dynamic Programming

In this second part of the assignment, you will gradually learn how to design an Approximate Dynamic Programming algorithm (ADP). First, you will develop an ADP algorithm in order to find the global optimum of a non-convex function [2]; this part of the assignment will be useful to get confident with some of the concepts exploited in ADP, such as Dynamic Programming, value and policy iteration and Actor-Critic structure. The interested reader could find all these notions extensively explained in [3]. Finally, you will consider a nonlinear switching system. The ADP algorithm in [4] will be used in order to find optimal control and optimal switching time.

2.1 Background

The Answers to the following questions can be easily found in the provided references.

- Give a definition of Dynamic Programming. In which sense does it constitute a *backwards-in-time* procedure? What is the principle it is based on?
- Give a brief definition of Actor-Critic structure.
- What does a basis function of a Neural Network represent? Why does it assume great importance in the Actor-Critic architecture?
- The Neural Networks update their weights by means of the least square method. Describe the steps that lead to the solution of a linear least square problem.

2.2 Non-convex function minimization using ADP

The ADP algorithm can be used for instance to learn the global minimum of non-convex functions. Given the function

$$\psi(x) = 2.1333x^4 + 0.9333x^3 - 2.1333x^2 - 0.9333x + 1 \tag{11}$$

The code "SC42120_non_convex_function_minimization.m" will lead you through the main steps necessary for the ADP design. The reference for this section is [2] and in particular, the **Algorithm 1**. Regardless the completion of the tasks, it is important trying to understand what the code is doing. Finally, the only parts of the code that should be modified are the %TODO lines.

2.2.1 Tasks

- In Step 2, the Critic weights are initialized for k=N. Determine "FinalW(:,k)".
- During step 6, the algorithm iterates the control Action $u_k = -\frac{1}{2}\bar{R}^{-1}\nabla\Phi(x_{k+1})^T W_{k+1}$. Determine $u_k^{i+1,[j]}$ and x_{k+1}.
- In step 9, $W_k^T\Phi(x_k^{[j]}) \simeq J_k(x_k)$. Find W_k using the least square method.
- Plot the results for $\psi(x)$ and $J_0 = W_0^T\Phi(x)$. Compare $x_{min} = argmin(\psi(x))$ with $x_{learnt} = argmin(J_0)$. Which is the percentage error?
- Re-run the code, does the percentage error change? Why?

For the sake of checking whether the code has been implemented correctly, Figures 4 and 5 show respectively the expected results for $\psi(x)$ and J_0.

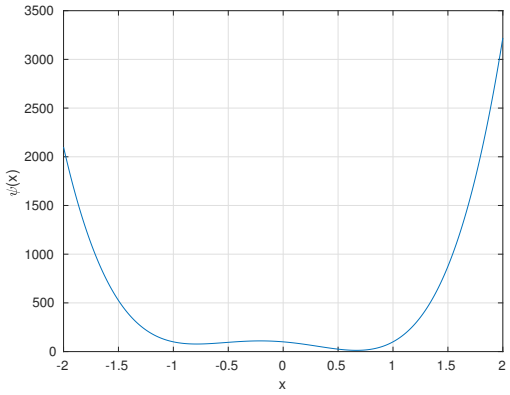

Figure 4: $\psi(x)$

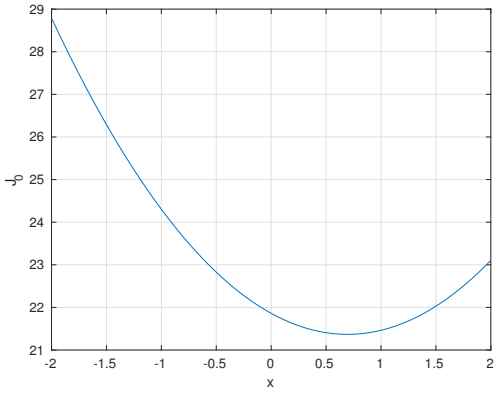

Figure 5: J_0

2.3 ADP for optimal switching and control of a non-linear switching system

In this last section, the steps for the ADP developed in [4] and in particular in the **Algorithm 2** are illustrated. The considered dynamics is a switching nonlinear system with two subsystems and one switch. The subsystem dynamics are

$$\begin{aligned} \text{Subsystem 1}: \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \begin{bmatrix} x_2 \\ x_1^2 - x_2^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ \text{Subsystem 2}: \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \begin{bmatrix} x_2 \\ -x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} u \end{aligned} \tag{12}$$

Where the elements of the state vector x are denoted by x_1 and x_2. A quadratic cost function in the following form is selected:

$$J = x(t_f)^T S x(t_f) + \frac{1}{2}\int_0^{t_f} \Big(x(t)^T Q x(t) + u(t)^T R u(t) \Big) dt \tag{13}$$

The code to complete is in "SC42120_Optimal_switching_and_control" and as previously, the only parts that should be modified are the %TODO lines.

2.3.1 Tasks

- Give a definition of $\psi(x)$ and determine it in function of S. Use Function Handle as done in "SC42120_non_convex_function_minimization.m". For further information: type "doc function handle" on the Matlab terminal.

- Complete with the systems dynamics in continuous time. (Lines: [29-46])
- In [4] a time transformation is defined. Why do we need it? What does $\hat{t}$ represent?
- Define the discretized versions $\bar{Q}_1$, $\bar{Q}_2$, $\bar{R}_1$, $\bar{R}_2$, $\bar{f}_1$, $\bar{f}_2$, $\bar{g}_1$ and $\bar{g}_2$ of Q, R, f_1, f_2, g_1 and g_2. Consider carefully how the time transformation affects the discretization.
- Step 1 of Algorithm 2, define W_N.
- Step 6, determine x_{k+1} and u_k.
- Step 7, determine V_k^{i+1}.
- Step 10, determine W_k^T.
- Once the learning is concluded, show the plots obtained for $x(t)$ and $J_0(t_{switch})$ for $x_0 = [-1; 0.5]$. Give a brief comment.
- Change $x_0 = [0; -1]$, show the plots and comment the obtained results. Why, in your opinion, do different initial conditions lead to different results?

The Figures below show the expected results for $x_0 = [-1; 0.5]$.

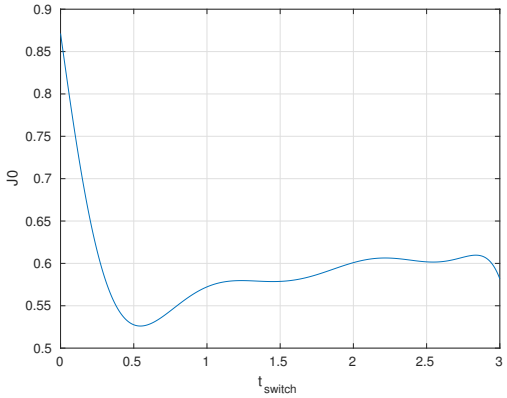

Figure 6: J_0

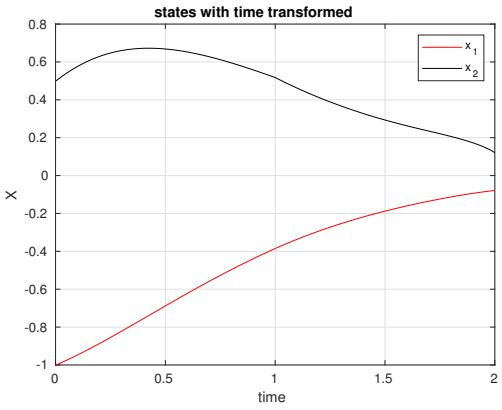

Figure 7: X with time transformation

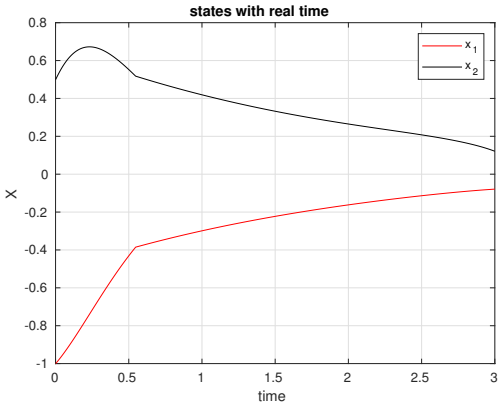

Figure 8: X real time

2.4 Submission

For the submission of this second part of the assignment is expected:

- A report where the different tasks are discussed.
- The following completed MATLAB files:

Task 1	"SC42120_non_convex_function_minimization.m"
Task 2	"SC42120_Optimal_switching_and_control.m"

References

[1] M. R. Rosa, S. Baldi, X. Wang, M. Lv, and W. Yu, "Adaptive hierarchical formation control for uncertain euler–lagrange systems using distributed inverse dynamics," *European Journal of Control*, 2018.

[2] A. Heydari and S. N. Balakrishnan, "Global optimality of approximate dynamic programming and its use in non-convex function minimization," *Applied Soft Computing*, vol. 24, pp. 291–303, 2014.

[3] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, 2009.

[4] A. Heydari and S. N. Balakrishnan, "Optimal switching and control of nonlinear switching systems using approximate dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1106–1117, 2014.