Louis Carton Winter Quarter

MAE 200 : Controls - Final Project

Study of a Dual Pendulum

1. Statement of the problem

For this project, we want to design a set of controllers and estimators in order to realize a swing up of a dual pendulum system. The dynamics of such system have been detailed in professor Bewley's book (Numerical Renaissance) on page 506. What is of interest for this project is the controls operated to those dynamics.

The problem will be separated in two distinct parts, Phase A and Phase B. The first one will deal with the "swing-up" of both pendulums along an optimized trajectory and therefore constitutes a linear-time-varying (LTV) problem. Phase B will focus on the stabilization of both pendulums at the vertical position, which is a linear-time-invariant (LTI) problematic.

This document is not an exhaustive description of how such problem would be solve, rather a guide of the Matlab code used to solve this problem. However, the important equations as well as the important results will be displayed and commented. For more comprehensive information and details, please refer to professor Bewley's book (Numerical Renaissance).

Note : In order to have a better understanding of all the functions and their link between them, you will find at the end of this document a scheme detailing their interactions.

2. Phase A

a. Step 1 : Optimization of the trajectory using Model Predictive Control

The objective of this first step is to do what we call a "worm start" by running the code "Example_22_2" on Matlab with the loop created in the first box of "Project_Dual_Pendulum".

The objective is to minimize the cost function :

$$J = \int_0^T x^T(t)Q(t)x(t) + u^T(t)Ru(t)\, dt + x^T(T)Q(T)x(T)$$

With T being the time maximum for the dual pendulum to "swing-up". For our problem, T=3sec.

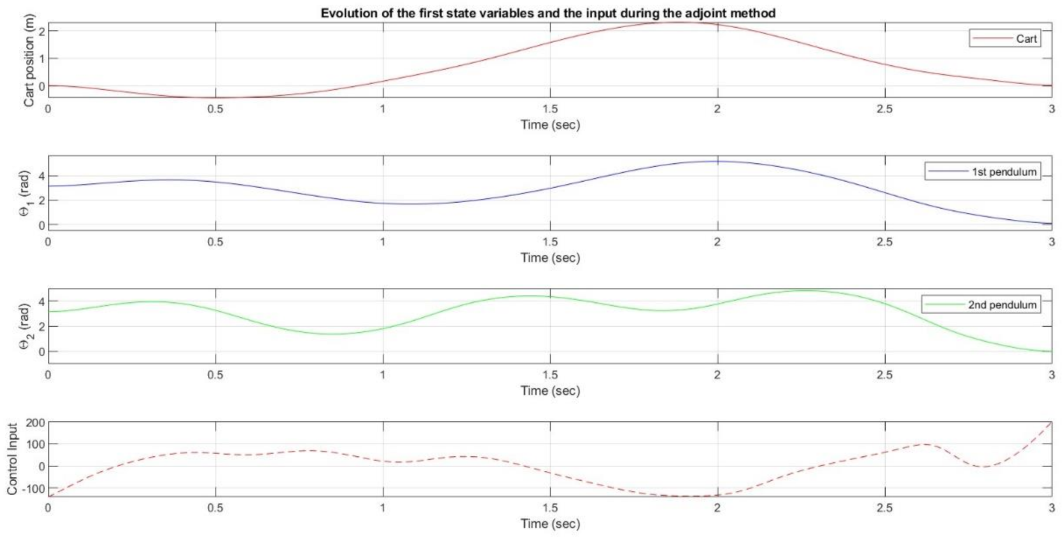
$Q(t), R \text{ and } Q(T)$ are what we call weight matrixes. They are used to penalize some state variable, at time $t \in [0, T[$ with Q(t) or directly at the final state when $t = T$ with Q(T).

We start with an initial input, here [0 0 0 0 0 0], then compute the associated x_k (=optimized trajectory) through the adjoint method, using the gradient of the cost function etc. For more detail, refer to chapter 22.1.3 in "NR".

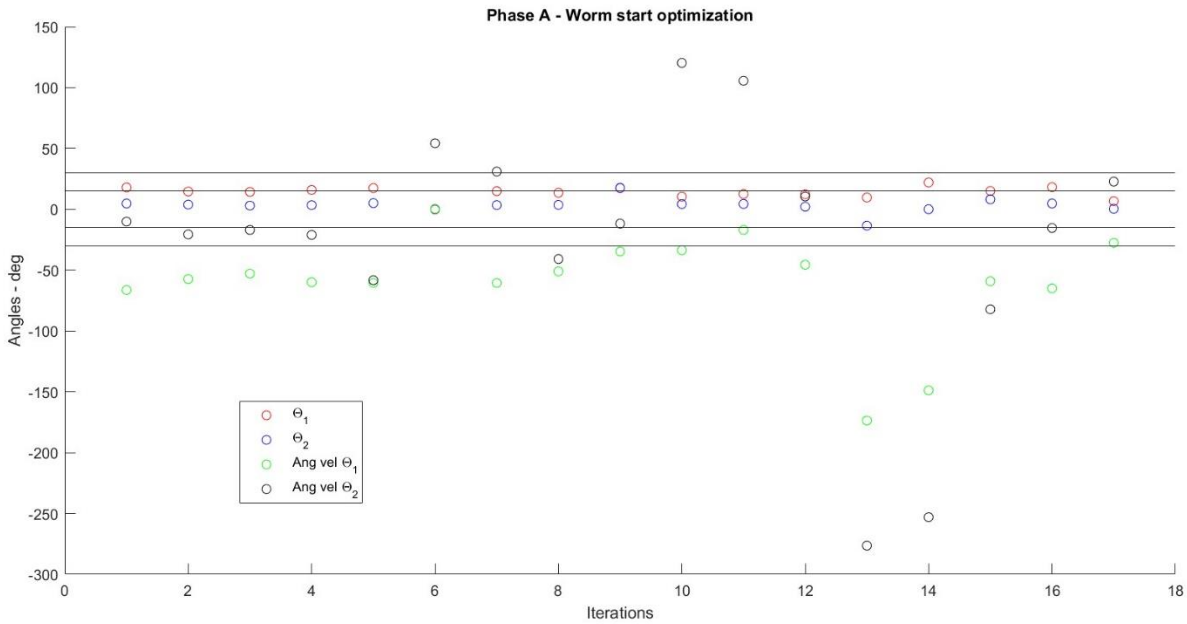
We then put some conditions on our final state x_k. Indeed, as we want to swing-up the pendulum, we need to have the angles of the pendulum close to zero at the end of time T. At the beginning I thought that only the angle position needed to be constraint, however when continuing the project, I remarked that I also needed to constrain the angular velocity of the pendulum. This is the case because in Phase B, we are linearizing the system around the constant position [0 0 0 0 0 0]. And this is only possible if the final state of Phase A that is the transition to Phase B is close enough to this state. When the loop is finished, we feed it with the last set of input in order to get closer and closer to the wanted result.

The whole code for this part is in "Phase A – Part 1 : Swing up" and is commented with the used constraints on both the positions and velocities.

On the graph below we can see the evolution of the state space variables as well as the control input from time t=0 to time t=T during the whole loop process.



On the second graph we can see the precise evolution of the variables of interest (angles and angular velocity) as the optimization is underway (# iterations). The black lines are the thresholds. The one for the angles is lower than for the angular velocities as they are more difficult to control.



At the moment when the targeted values for the angles and for angular velocities are reached, the optimization is over and we get as a result the optimized trajectory for the swing-up.

This is a crucial part because it will determine the rest of all the rest of the work, up until the very end (see 3.b).

b. Step 2 : Optimal Control Design – LTV

The objective of this step is to design the matrix K(t) which is the optimal control of the system.

Again, for more details in the step to follow as well as explanations in the methods, please refer to chapter 22.1.2 from “NR”.

The most important equation for this problem is the Differential Riccati Eqaution (DRE) that we will need to solve backward in time with an appropriate RK4 function. After manipulating the function, we have it under the following form that we then can implement in Matlab under the function “DRE”.

$$\frac{dX}{dt} = -E^{-T}A^{T}X - XAE^{-1} + BR^{-1}B^{T}X - E^{-T}QE^{-1}\ with\ X(T) = Q(T)$$

After solving for X we can compute the optimal controller K at each time step :

$$K = -R^{-1}B^{T}XE$$

Since we want to stabilize our system, we check the obtain results by computing the eigenvalues of “A+BK” with the Matlab function $eig(E^{-1}A + E^{-1}BK)$.

However I found out that I do not obtain at every time step, the real part of the eigenvalues of the system to be negative. It turns out that this method to ensure the stability of a system by checking the eigenvalues is not suitable for time varying problem. In fact, the eigenvalues of the close loop system and for each state variable is going to alternate between positive and negative. However we can still have a stable system because it just means that at the time t=t’ and during the time step s.h=0.001sec this state variable was not stable but will be in the time t”=t’+s.h which will make, in the big picture, the system stable.

To check that the system is really stabilized by the optimal control, we will compute the dynamics of x’, the “controlled” path and check if it follows the optimized path $\bar{x}$.

We are studying the following system :

$$E\frac{dx}{dt} = Ax + Bu$$

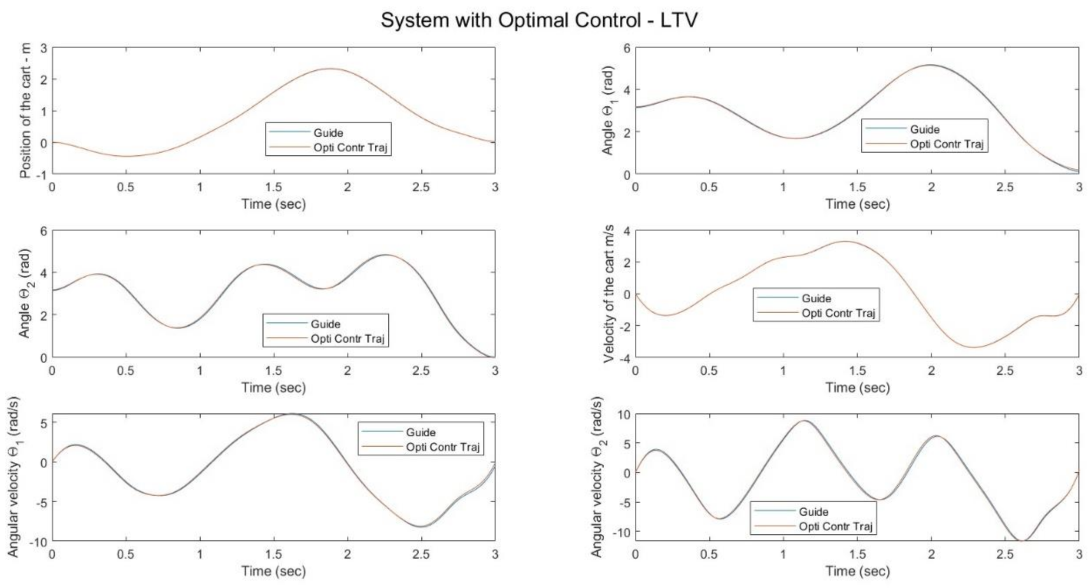
With $\begin{cases} x = \bar{x} + x'\ with\ \bar{x}\ the\ optimized\ trajectory\ and\ x'\ the \text{ 'controlled' trajectory} \\ u = \bar{u} + u' \end{cases}$

Replacing x and u in the first equation and using the optimal control relation (u=Kx) we eventually find the following equation for the dynamics of x’:

$$\frac{dx'}{dt} = -E^{-1}BK\bar{x} + E^{-1}B\bar{u} + E^{-1}(A + BK)x'$$

We are solving this differential equation with the RK4 method and implemented in “Phase A – Part 2 : Optimal Control LTV (K(t))”.

The result is displayed below and show a perfect control of the trajectory to stay as close as possible to the optimal trajectory. This is a proof that the optimal control stabilize our system.



c. Step 3 : Kalman Filter Design – LTV

The objective of this step is to design the matrix L(t) in order to realize a Kalman filter for the system.

Once again we will need to solve the DRE equation but with different matrix and coefficient than for the previous section. For more details, please refer to chapter 23.2.1 of "NR".

$$\frac{dP}{dt} = AP + PA^T - PC^T R^{-1} CP + Q \; with \; P(0) = P0$$

We will solve this equation forward in time with the RK4 method and using again the "DRE" function in Matlab.

Then we compute

$$L(t) = -PC^T R^{-1}$$

In order to check the stability of the closed loop system, we do the same process as the previous section. Computing the eigenvalues of A+LC but we have the same results. They are not all negative.

We then compute the dynamics of the estimated system $\hat{x}$ and check if it follows the optimized trajectory.

For the estimator we are studying the following system :

$$\frac{dx}{dt} = (A + LC)x + Bu - Ly$$

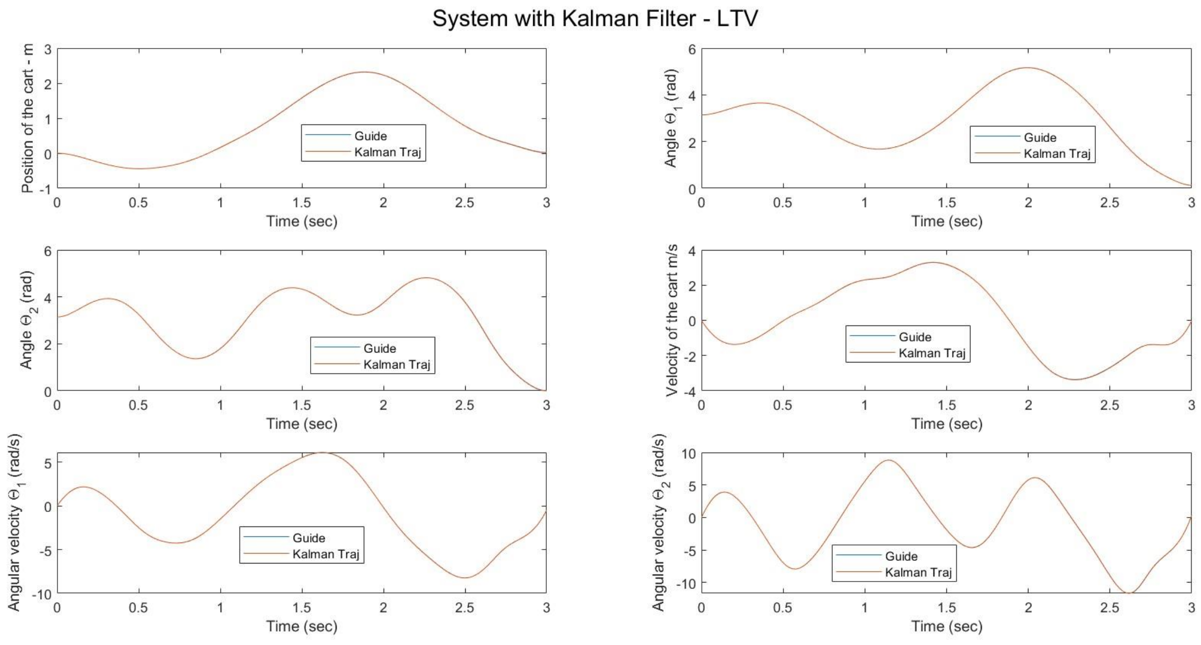
$$\text{With} \begin{cases} x = \bar{x} + \hat{x} \; with \; \bar{x} \; the \; optimized \; trajectory \; and \; \hat{x} \; the \; \text{'estimated' trajectory} \\ u = \bar{u} + u' \\ y = C(\bar{x} + \hat{x}) \\ x' and \; u' \; the \; controlled \; trajectory \; and \; input \end{cases}$$

After reorganizing, we obtain the following equation for the dynamics of the estimated system :

$$\frac{d\hat{x}}{dt} = (E^{-1}A + LC)\hat{x} + (E^{-1}BK - LC)x' + E^{-1}B\bar{u} - E^{-1}BK\bar{x}$$

We solve this equation with again the RK4 method, and using the function "RHS_Kalman" in Matlab.

We obtain the following result, that satisfies completely the design of L(t) since the two paths have the same trend.



3. Phase B

For this phase, the problem is completely different and we are only looking for the design of a controller K in order to stabilize the close loop as well as an estimator L. Those problems are easier to solve because we linearize around a position and we are working on the infinite horizon where $t \in]T, \infty[$ which has some particular implications.

a. Step 4 : Design of the controller – LTI

We have to solve the ICARE equation :

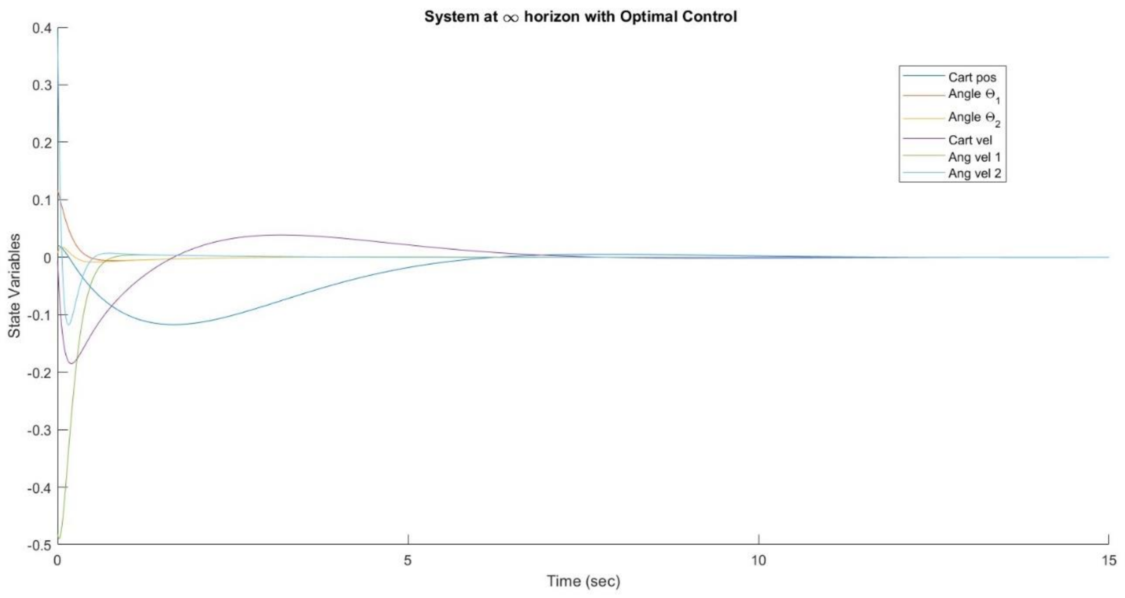
$$0 = A^T XE + E^T XA - E^T XBR^{-1}B^T XE + Q$$

There exist a function directly in matlab that solves it and give the gain function as well as the eigen values of the close loop system.

For this step 4, the gain function is the controller K and the eigenvalues are (A+BK), for which we have to have $Re\{\lambda(i)\} < 0$ in order to ensure stability.

This is the case, we can check them with the variable “EV_contr_b”.

We can see the evolution of the state variable, from the end of phase A, T=3sec. The time on this plot is equal to t=t+T. The system stabilize pretty quickly which is good and is another proof (graphical this time) that our controller is nicely designed.



b. Step 5 : Design of the estimator – LTI

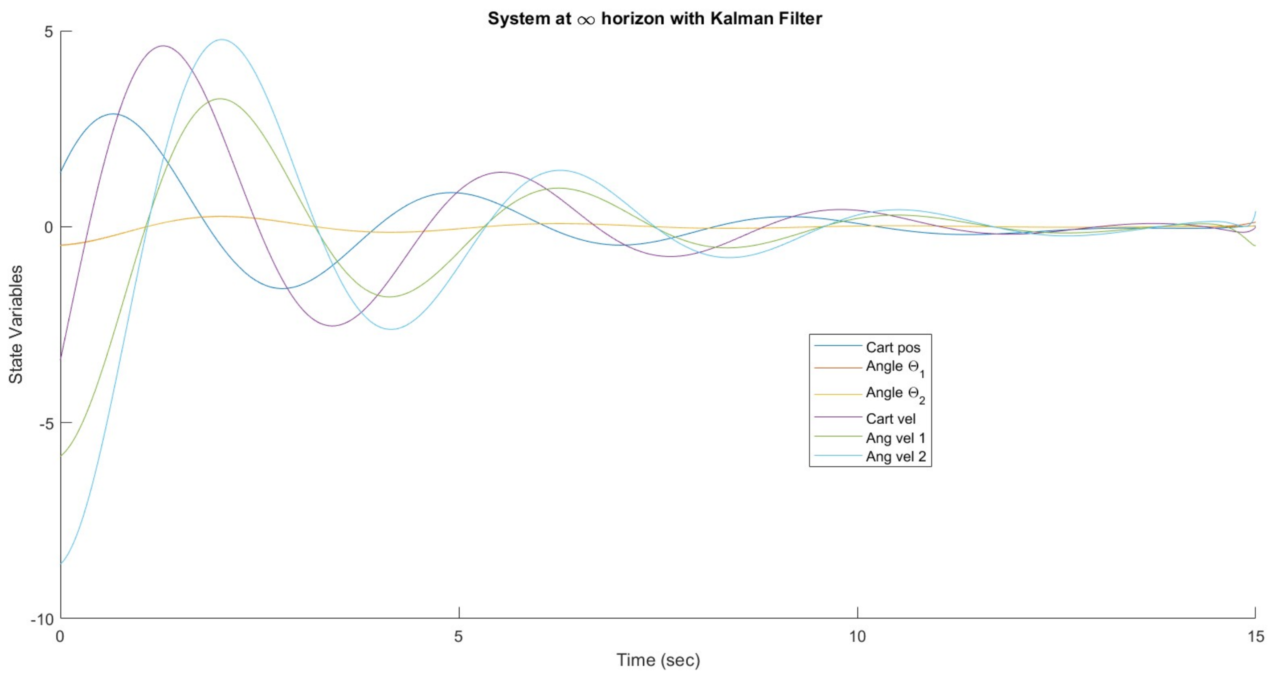
We have to solve the same ICARE equation but with different coefficients (same idea as for phase A).

$$0 = AP + PA^T - PC^T R^{-1} CP + Q$$

Again, the function icare in Matlab is solving the equation for us and is giving us the estimator L as well as the eigenvalues of the closed system.

In that case too, the $Re\{\lambda(i)\} < 0$ so we are guaranteed of the stability. This can be easily checked in the variable "EV_obs_b".

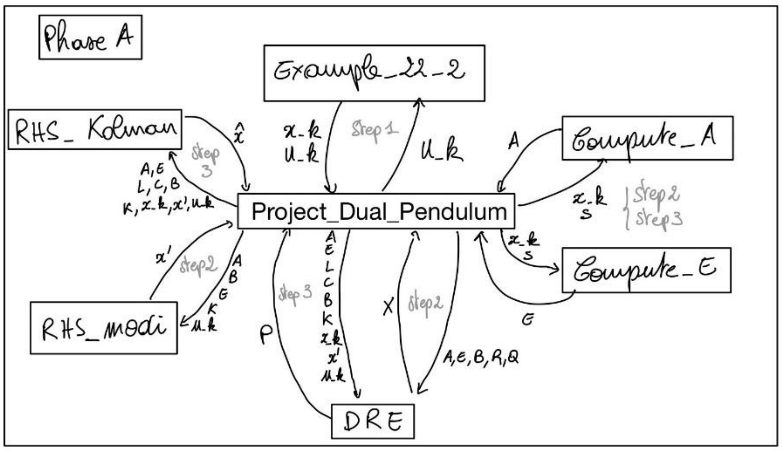
This last part is important because the plot of the closed loop system, including both the controller and the state estimator will tell if the optimized trajectory from Phase A is good enough for the Phase B to bring to the unstable equilibrium position in the upward position.



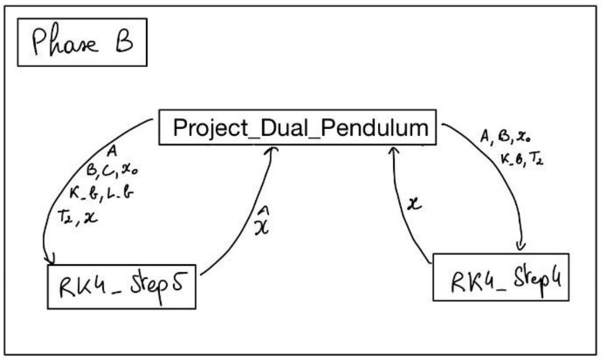
As we can see on the graph below, even if the state at the end of phase A is not perfectly equal to zero (not at all in fact), the system seems to still manage to bring it to zero after a time t=T2=15sec.

This means that the dual pendulum system needed t=T1+T2=3+15=18 sec to reach from the position downwards to swing up and stabilize upward.

- Functions diagram



In this diagram, only the functions that do not exist already in Matlab are shown.


Phase B
Project_Dual_Pendulum
A
B, C, x₀
K_B, L_B
T₂, x
x̂
x
A, B, x₀
K_B, T₂
RK4_Step5
RK4_Step4