Design Lab - Report

Raj Kishore*
Indian Institute of Technology Kharagpur
West Bengal, India
rajkishore@kgpian.iitkgp.ac.in

Abstract

This document consists of my notes and progress about the project for the Design Lab. The project topic is titled as "Data-driven Koopman model predictive control". This piece of work aims to reproduce the results for reference signal tracking problem-1 in the paper referenced below. In problem-1, dynamics is being governed by Burgers' Partial differential equation which is a non-linear PDE and that in-turn makes the dynamics non-linear. Koopman operator theory is used to convert non-linear dynamics to a linear one.

1 Introduction

There could be many ways of solving a problem which in our case is reference signal tracking problem in case of non-linear dynamical systems. In our case, we adopt a data-driven approach to convert a non-linear dynamics to a linear dynamics. Further, we build model predictive control(MPC) in that uplifted space(where dynamics became linear) and use the very first control input from the sequence of control inputs obtained over a finite time horizon for manipulating the actual system dynamics. We follow data-driven approach with MPC to converge to a changing reference signal.

2 Theory

Flow of theory is in accordance to the code written. We will elaborate it for the example-1 where dynamics is given by Burgers' non-linear partial differential equation.

we consider the Burgers' equation with periodic boundary condition,

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial z} = \nu\frac{\partial^2 v}{\partial z^2} + f(z,t) \tag{1}$$

$$z \in [0,1]$$
$$t \in [0,\infty]$$
$$v(0,t) = v(1,t)$$

where ν is the kinematic viscosity. Note that we have used z to denote the spatial coordinates in the flow examples.

*https://in.linkedin.com/in/raj-kishore-55403b25a

We assume the forcing $f(z,t)$ is given by $f(z,t) = u1(t)f1(z) + u2(t)f2(z)$

where,

$$f1(z) = e^{-(15(z-0.25))^2}$$
$$f2(z) = e^{-(15(z-0.75))^2}$$

with the control input $u = (u1, u2) \in R^2$

The control objective is to follow the reference state, starting from the initial condition,

$$v(z,0) = ae^{-(5(z-0.5))^2} + (1-a)sin(4\pi z)$$

with $a \in [0,1]$ chosen randomly, and with the input signals constrained as $|u1|, |u2| < 0.1$

We have first used the reference signal given in the paper and then used self-made reference signal for simulation and one could try playing with the reference signal.

2.1 Data Collection

We first collected data for system dynamics and instead of generating it by our own we used the data generated by the original code itself. For numerically solving the partial differential equation, the method of finite differences is used. Three data matrices have been obtained X, Y and U. Data is collected for 100 trajectories each with simulation length of 200
where,
X is data matrix,
Y is data matrix shifted to 1 time unit in the future and
U is the control input matrix.

Now, creating uplifted matrices of X, Y and U as X_{lift} and Y_{lift}.

2.2 Koopmanism and EDMD-Algorithm

According to Koopman operator theory, System dynamics converted from non-linear to linear in case of control inputs could be written as:

$$Y_{lift} = AX_{lift} + BU$$
$$X = CX_{lift}$$

After creating such matrices we need to find approximated Koopman matrix on solving the following least squares problems -

$$\min_{A,B} |Y_{lift} - AX_{lift} - BU|$$
$$\min_{C} |X - CX_{lift}|$$

C could also be approximated as done in the code.

This Algorithm is termed as Extended Dynamic Mode Decomposition(EDMD).

Using EDMD Koopman matrix have been found which is of the form,

$K = \begin{pmatrix} A & B \\ C & 0 \end{pmatrix}$
where K is the Koopman Matrix.

2.3 Simulation loop and MPC

Now as we have converted our dynamics to a linear one, we continue to build a controller in the lifted space using model predictive control(MPC) and pick the very first candidate from a sequence of control inputs, calculated over a chosen finite time horizon, via MPC on minimizing a certain objective function.

Objective Function,

$$\mathrm{J} = (X.T) * Q_N * (X) + \sum_{i=1}^{N-1} x_i.T * Q * x_i + u_i.T * R * u_i - 2 * x_{ref}.T * Q * x_i - 2 * u_{ref}.T * R * u_i$$

where, N is the size of finite time horizon

This step of MPC is repeated under a simulation loop for a specific number of times.

3 System setup

• We coded the entire problem and solution steps in Python.

• We first installed **VScode**-software and **python(version 3.11.8)**, Integrated the two in order to write code in python.

• Created virtual environment and installed all the necessary packages before continuing further, also **requirements.txt** file is already provided in the github repository.

• Unlike the original paper provided in the reference, we have not hard coded everything using basic code snippets, instead we made use of **casadi** as a facilitator in-order to formulate the **model predictive control(MPC)** problem as an optimization problem using **multiple shooting method**.

• It has already been indicated in the github repository itself about which files needed to be kept where in-order to run the code.

4 Conclusion

For this project, results have obtained and tested for two different square wave and it works absolutely fine. The results obtained from the code are for a specific reference signal and in-order to converge to any other square wave reference signal one must change max and min value of the control input accordingly.

Appendix

A How to Run Matlab Code

As of now in order to run code for Burger's Example in matlab :

• I had to install minGW using matlab itself.

• Keep all the related files under "C: - Users - premsha73866453 - Documents - MATLAB - thehood - qpOASES-3.1.0 - interfaces - matlab"

• Open matlab and type -
make [enter]
help qpOASES [enter]
BurgersExample [enter]

B Key Learning

• Providing a Numerical solution to a Partial Differential Equation(PDE) especially using the Finite Differences Method(FDM). Implemented as a code for solving Burgers' Equation.

• Build a good understanding on how a non-linear dynamics could be converted or re-framed to depict a linear dynamics over some other vector space using Koopman operator theory. Also, got hands on experience for implementing EDMD algorithm as python code for our reference signal tracking problem.

• Learnt about Model Predictive Control(MPC) and how it could be used to design a controller for dynamical systems. It also includes understanding of casadi framework, formulation of MPC problem into an optimization problem(multiple shooting method) and other related technicalities. Implemented as a code for building a controller for our case.

C Plan

Read the paper and start playing with the MATLAB code, which is available at https://github.com/arbabiha/KoopmanMPC_for_flowcontrol

I plan the project as follows:

C.1 Week 1

• Learn Basics of PDE(s), Dynamical systems, Koopman Operator Theory.

C.2 Week 2

• Learn the construction of Non-linear dynamics with finite dimension to Linear dynamics with infinite dimension, using Koopman operator theory.
• Learn about Model Predictive Control and how to optimize our objective function using it.

C.3 Week 3

• Download, Install Matlab and other necessary tools for further progression of the project.
• Try to understand the stuff of week-2 completely.
• Try understanding the Matlab code. (Githhub link given in the paper itself)

C.4 Week 4

• Try understanding week-3 stuff completely.
• Try Playing with the original code by changing some parameters and getting a firm idea how code actually works.

C.5 Week 5

• Try to wrap-up things covered up-until week-4, before moving onto Python Coding Part.

C.6 Week 6

• I would be solving a problem where system dynamics are goverened by Burgers Equation.
• Understand such a system and try to play with the matlab code in order to make things clear about the qpOASES, quadprog, how to create mpc, how to use EDMD in coding and other relevant stuff needed for coding.
• If done with the above part, start coding the problem for Python Programming Language.

C.7 Week 7

• Continue the coding part to solve the Burgers Equation related system dynamics.

C.8 Week 8

• Continue the coding part to solve the Burgers Equation related system dynamics.
• Make the python repository public on github.

D Progress

Here I document my progress.

D.1 Week 1

• Overview of PDE (in brief),
• Learnt about the basics of dynamical systems,
• Read about Koopman operator theory.

D.2 Week 2

• Learnt about construction of koopman linear systems for non-linear dynamical system.
• Learnt the Difference between DMD(Dynamic Mode Decomposition) and Koopman operator for converting non-linear system to linear system.
(DMD is an approximation of the original non-linear system into a linear system while on performing the co-ordinate change same non-linear system could be converted to a linear system but it is infinite dimensional which is Koopman theory.)
• References - prof. Nathan Kutz (yt-videos) and the above mentioned paper.

D.3 Week 3

• Downloaded and installed Matlab and python on my PC.
• Tried Learning about MPC and how optimization algorithm works.
(Have some doubts)
• Also tried going through matlab code present on Hassan Arbabi's Github.

D.4 Week 4

• Learnt DMD algorithm also EDMD algorithm is similar to DMD but it is applied with koopman linear systems.

• also went again with week-3 stuff and also read the paper.
(Have some doubts)

D.5 Week 5

• Completed things which one should know before moving to the coding part.
• Some of the important stuff is - how does MPC work, how to use EDMD-Algorithm, how to formulate problem into something which fits good with Koopman Operator Theory i.e. to convert non-linear dynamics into a linear one.
• Also ran Matlab code for Burgers Equation from the github repo as provided by Hassan Arbabi in his paper.

D.6 Week 6

• Wrote in PYTHON code for generation of data for Burger's Equation.
• Stuck on installing qpOASES for windows-64bit for Python.

D.7 Week 7

• Installed Casadi for solving the optimal control problem.
• Wrote code for EDMD-Algorithm to find Koopman matrix in order to approximate the original non-linear dynamics to a linear one.
• Implemented MPC using Casadi-facilitator via Multiple shooting method.
• For this time, solver used is ipopt.
• graph plotted for refrence signal against the system dynamics for each iteration resulting into an animation.
(Got some doubts)

D.8 Week 8

• Added reference control inputs.
• Made a few changes to objective function in order to account for the reference control inputs.
• Graph added for control inputs with respect to time or iterations.
• **Code finally works as desired.**

References

[1] H. Arbabi, and M. Korda, and I. Mezic. A data-driven koopman model predictive control framework for nonlinear partial differential equations. *C*onf. on Decision and Control, 6409–6414, 2018.

[2] MPC coding using casadi youtube tutorial by Mohamed W. Mehrez. (Post-Doctral at university of Waterloo) *h*ttps://www.youtube.com/watch?v=RrnkPrcpyEA

[3] Lectures on Koopman theory, Dynamic Mode Decomposition Algo., Extended Dynamic Mode Decomposition Algo. and other relevant concepts by *P*rof. Nathan Kutz (Post-Doctral at university of Washington)