Physics Informed Neural Networks based Model Predictive Control and dynamic obstacle avoidance

Pranav Gunreddy*
Dept of MEAM
University of Pennsylvania
grpranav@seas.upenn.edu

Vaidehi Som*
Dept of Robotics
University of Pennsylvania
somv@seas.upenn.edu

***Abstract*—Model Predictive Control (MPC) is a popular framework for autonomous systems. However, to achieve good control performance using MPC, an accurate dynamics model is quite necessary. Machine learning approaches, specifically neural networks, have been shown to model even complex dynamics but have not been quite effective. They are not suited for control tasks in their original form since they are not designed to handle variable control actions or variable initial values. Physics-informed neural networks (PINNs) impose known physical laws into the learning of deep neural networks, making sure they respect the physics of the process while decreasing the demand of labeled data. In this context, this work explores the idea of Physics-Informed Neural Nets-based Control (PINC), a unique control strategy where PINN-based architecture that is amenable to control problems is used along with MPC. The methodology enables the optimal control of dynamic systems, making feasible to integrate a priori knowledge from experts and data collected from plants in control applications. We showcase our method in the control of a self driving car with bicycle model dynamics. Our controller is then put to test against dynamic obstacles to see if the PINN modelled dynamics can accurately predict state vector in adverse situations. we use GRU as the trajectory predictors of the pedestrians, while making no assumptions on the underlying trajectory-generating distribution. To the best of our knowledge, these are the first results in such a setting.**

***Index Terms*—PINN, MPC, GRU, Lifelong learning A*, Dynamic obstacles, trajectory prediction, Motion planning**

I. Introduction

When we see the learning problem as a minimization of a cost function that can be augmented according to particular criteria, beneficial properties can be embedded into the deep network's learning. While regularization methods have proved to be very useful in machine learning, an alternative way to regularize supervised learning tasks consists of using the underlying process's known physical laws. As such, a deep neural network can learn the required behavior not only from data, but also from a priori knowledge built previously by experts or borrowed from the laws of nature [1]. This research area has been expanding quickly, mainly under the denomination of Physics-Informed Neural Networks (PINN). Control system dynamics are an obvious setting where PINN could prove to be extremely useful. Earlier work on neural networks based control are trained exclusively on data collected from the process and are thus not sample efficient as PINNs, as the latter can benefit from prior knowledge of the physics laws involved in the processes. Model Predictive Control (MPC) is one of the most popular frameworks thanks to its ability to simultaneously address actuation constraints and performance objectives through optimization. Due to its predictive nature, the performance of MPC hinges on the availability of an accurate dynamics model of the underlying system. This requirement is exacerbated by strict real-time constraints, effectively limiting the choice of dynamics models to simple first principle models. Combining MPC with a more versatile and efficient dynamics model would allow for an improvement in performance, safety and operation closer to the robot's physical limits

II. Literature Review

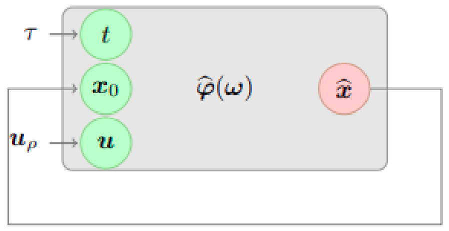
We primarily followed ideas presented by Antonelo [1] on how to replace the nonlinear dynamics with a PINN approximation in the context of nonlinear model predictive control (NMPC). PINNs have originally been talked about in [2] and [3]. [4] has explored PINN for control of multi link manipulators and successfully managed to model complex non linear dynamics of a robotic arm.

The use of ML techniques in MPC is not new. For instance, in [5] the controller itself is replaced by a neural network. [6] also talks about using deep learning models to learn complex dynamic systems. In the context of a-priori unknown tracking trajectories, such an approach is either not feasible or requires learning in a high-dimensional space. Hence we limit our work to learning only the dynamics of the control system. Lastly, Dynamic obstacle avoidance and the concept of safe planning ideas have been borrowed from [7] and human trajectory predictions using GRUs from [8]

III. Problem Formulation

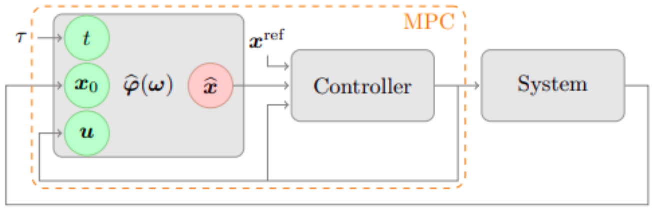
This work talks about PINC (Physics informed neural net based control) and attempts to solve the following problems -

1) To accurately model a PINN system with right i/o that can learn the dynamics of bicycle model
2) To Integrate the model with an MPC controller such that optimization in the prediction horizon and application of control action are performed in one pass
3) To test the efficiency of the PINC control system collision avoidance with dynamic obstacles



T
$\boldsymbol{u}_\rho$
t
$\boldsymbol{x}_0$
$\boldsymbol{u}$
$\widehat{\varphi}(\boldsymbol{\omega})$
$\widehat{\boldsymbol{x}}$

Fig. 1: PINN in self loop prediction



MPC
T
t
$\boldsymbol{x}_0$
$\boldsymbol{u}$
$\widehat{\varphi}(\boldsymbol{\omega})$
$\widehat{\boldsymbol{x}}$
$\boldsymbol{x}^{\text{ref}}$
Controller
System

Fig. 2: PINN based MPC in closed loop

IV. Methods

A. Physics Informed Neural Nets (PINNs)

PINNs are control amenable since control inputs are added to them. The initial state of the system, control input in addition to the continuous-time t make the inputs to the model. This augmentation is inspired by the multiple shooting and collocation methods, which are numerical methods for solving boundary value problems in ODEs, which split the time horizon over which a solution is sought into several smaller intervals (shooting intervals). Unlike neural networks that assume fixed inputs and conditions, the proposed PINC framework operates with variable initial conditions as well as control inputs that can change over the complete simulation, making it suitable for model predictive control tasks as illustrated in Fig. 2. The output of the network is given by:

$$y(t) = f_w(t, y(0), u)$$

where f_w represents the mapping given by a deep network parametrized by weights w

Since t is an input to the network, the state at t = T can be directly inferred by a single forward network propagation:

$$y[k] = f_w(T, y[k-1], u[k])(7)$$

where the initial state is set to the last state of the previous step, i.e., $y[k-1]$ We call f_w the control interface for the PINC framework. Thus, $\partial f_w / \partial u$ can be computed for providing the Jacobian matrix to solvers used in MPC, possibly by means of automatic differentiation. We then created a dataset mapping input vector x, u and t to predicted state space vector x' which are all derived by solving a regular MPC problem in different possible conditions. The dataset has been diversified to make it robust against new situations.

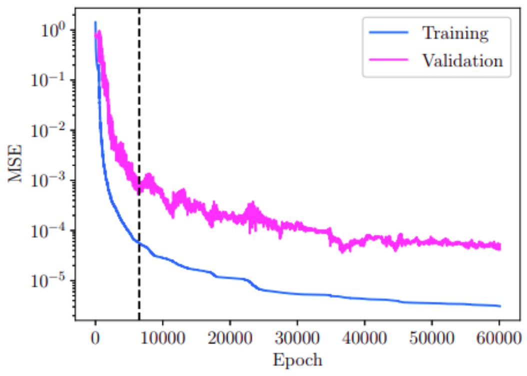


Fig. 3: MSE evolution during training of the final PINN

B. MPC with PINN (PINC)

Modes of operation of the PINC network as shown in Fig. 1 and Fig. 2 are

1) PINC operating in self-loop mode, using its own output prediction as next initial state, after T seconds. This operation mode is used within one iteration of MPC, for trajectory generation until the prediction horizon of MPC completes (predicted output from "Fig. 1")
2) Block diagram for PINC connected to the plant. One pass through the diagram arrows corresponds to one MPC iteration applying a control input u for Ts timesteps for both plant and PINC network. Note that the initial state of the PINC net is set to the real output of the plant. In practice, in MPC, these two operation modes are executed in an alternated way

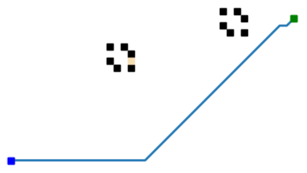
The state space vector thus, given by the PINN is used by the MPC controller to deduce the control scheme and apply it. Since we are solving the obstacle avoidance problem, we have carefully modelled the reference trajectory to account for sudden pedestrian movement. Lifelong planning A* has been chosen for this purpose.

LPA* is an incremental version of A*, which can adapt to changes in the graph without recalculating the entire graph, by updating the g-values (distance from start) from the previous search during the current search to correct them when necessary. Like A*, LPA* uses a heuristic, which is a lower boundary for the cost of the path from a given node to the goal. A heuristic is admissible if it is guaranteed to be non-negative (zero being admissible) and never greater than the cost of the cheapest path to the goal.

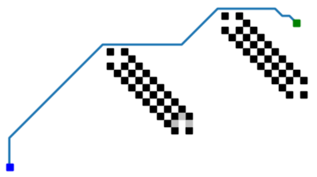
Since with a self driving car, we know the starting and end positions and we try to avoid obstacles, LPA* works perfectly as it gives the updated trajectories at each time step adding cost in case the PINN predicted vectors run into obstacles/ go out of path as shown in Fig. 4.

C. Trajectory prediction

We predict the trajectory of the agents using Gated Recurrent Units (GRU). Temporal data of pedestrians are fed to the model, which becomes the observed trajectory of that



(a) After predicted data of agent is used



(b) After predicted data of agent is used

Fig. 4: Change in path after predicting pedestrian's trajectory

agent. Using this data, the GRU predicts the next steps of that agent. We train our model using the ETH opensource dataset from [9]. Preprocessing has been performed on the data. Since the dataset is highly stochastic in terms of the number of pedestrians in each frame and trajectory length, preprocessing eliminated pedestrians based on

$$L_{pedes} < L_{total}$$

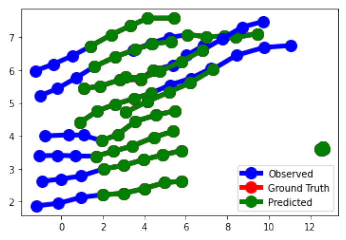
where, $L_{total} = L_{observed} + L_{predicted}$

Trajectories with length greater than L_{total} were truncated. The following three comparison metrics were used for comparing the prediction accuracy between different machine models and datasets:

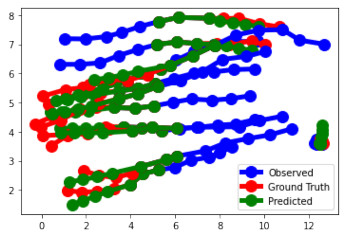
1) Mean Square Error (MSE)- We calculated the second norm of the difference between two vectors to evaluate the prediction of each pedestrian
2) Average Displacement Error - Euclidean distance between each step of the predicted trajectory and ground truth, further dividing it by the trajectory length
3) Final Displacement Error - Euclidean distance between the final predicted position and ground truth position of each pedestrian's trajectory

We have used this as the input for our reference trajectory in Fig. 2

GRU: 8 obs frames, 8 pred. frames			
Learning rate	0.0005	0.0007	0.0017
Avg train loss	0.1639	0.1491	0.1759
Avg test loss	1.7303	2.1536	2.8133
Avg train disp err	0.5853	0.5529	0.5307
Final train disp err	0.9363	0.8921	0.8583
Avg test disp err	1.2648	1.3957	1.5666
Final test disp err	2.1357	2.3529	2.5491



(a) 4 steps



(b) 6 steps

Fig. 5: Pedestrian trajectory predicted for different steps

V. Results

We tested our model in a dynamic setting with 2 pedestrians currently at $p_0 = [1.66, -2.67]$ and $q_0 = [1.84, 0.48]$ having travelled for the last $10s$. The trajectories of these pedestrians for the next $10s$ are predicted and motion is planned for the robot using the model in context. So here our observed length is $10s$ and predicted length is $10s$.

The Robot is initially at $r_0 = [0.5, -2.5]$ and aims to reach the point $r_f = [2.5, -7.5]$ in the next $10s$.

The circles around the pedestrian are the model(GRU) uncertainty that have been added around the pedestrian's predicted position as a safety check. As observed in Fig. 6 and Fig. 7, we can see that the model senses the dynamic obstacles correctly and predicts accurate control scheme to avoid them and reaches its goal.

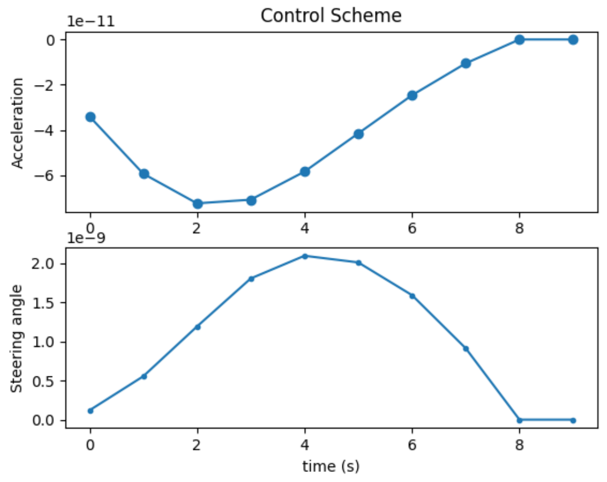
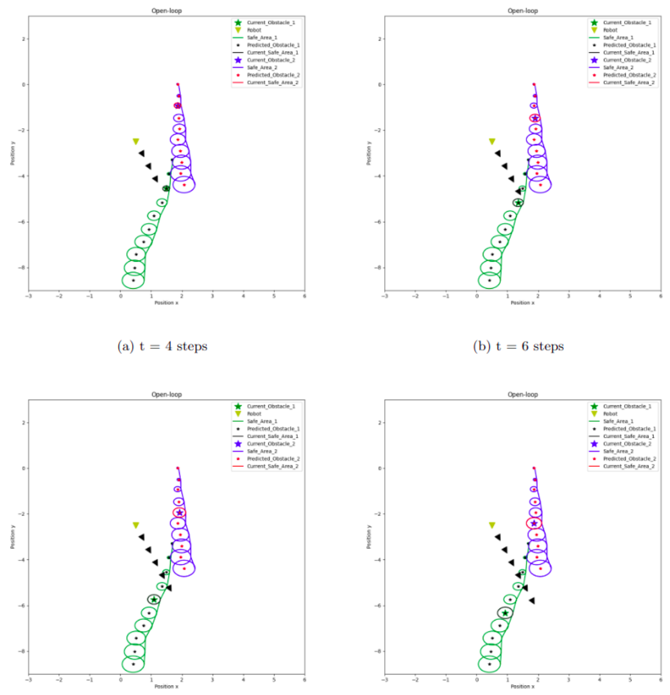


Fig. 6: Control scheme we got for the below shown scenario



(a) t = 4 steps

(b) t = 6 steps

(c) t = 8 steps

(d) t = 10 steps

Fig. 7: Final output path of ergo vehicle taking in account pedestrian's trajectory

VI. Conclusion & Future scope

We demonstrated our method by taking the case of a self driving car with bicycle model dynamics. We trained a PINN model with IPOPT solver in CASADI framework by generating reference motion vectors, to learn the dynamics. In cohesion with MPC controller, we designed a Physics informed neural net contoller.

1) Lifelong A* planning algorithm gave us the desired reference trajectory when we consider the temporal data of the pedestrians. The method also proved to be effective in generating paths that downright avoided trajectories. Since it is updated at every time step, even if our control actions predicted by PINN MPC do not consider obstacles, the reference trajectory they take make sure they do not collide by adding cost.
2) Coming to trajectory predictions, GRU models work well with very low error to predict 6 steps in future. Being computationally effective it can be run in real time for trajectory prediction.

In future, The primary problem to encounter would be to make the PINN model robust by training it on relevant data. Statistical tools need to be used to quantify uncertainty

1) PINNs are capable of predicting trajectories by considering dynamic obstacles too. Hence LSTMs/GRUs could be integrated so that our robot traverses safely and use of LPA* could be eliminated.
2) More pedestrians and other types of dynamic agents (cars, etc) could be introduced into the system so that the model learns how to efficiently predict trajectories in a crowded place without blowing computationally
3) Design PINN for closed loop MPC and explicitly compare time of computation between methods to prove PINN's utility
4) Use Transformers to predict pedestrian's trajectory. (Social) Transformers can be used to predict more steps into the future with more certainty while considering pedestrian's surroundings into account

References

[1] Antonelo, Eric & Camponogara, Eduardo & Seman, Laio & Souza, Eduardo & Jordanou, Jean & Hübner, Jomi. (2021). Physics-Informed Neural Nets-based Control.

[2] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (Part I): Data-driven solutions of nonlinear partial differential equations

[3] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[4] Jonas Nicodemus, Jonas Kneifl, Jörg Fehr, Benjamin Unger: Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators. arXiv:2109.10793

[5] Akesson, B.M. and Toivonen, H, T. (2006). A neural net- work model predictive controller. Journal of Process Control, 16(9), 937–946. doi:10.1016/j.jprocont.2006.06.001.

[6] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza and Markus Ryll: Real-time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms. arXiv:2203.07747v2

[7] Lars Lindemann, Matthew Cleaveland, Gihyun Shim, and George J. Pappas: Safe Planning in Dynamic Environments using Conformal Prediction. arXiv:2210.10254v1

[8] Luca Rossi, Marina Paolanti, Roberto Pierdicca, Emanuele Frontoni, Human trajectory prediction and generation using LSTM models and GANs, Pattern Recognition, Volume 120, 2021, 108136, ISSN 0031-3203

[9] A. Ess, B. Leibe and L. Van Gool, Depth and Appearance for Mobile Scene Analysis, 2007 IEEE 11th International Conference on Computer Vision, 2007, pp. 1-8