Simulating the Effectiveness of Physics-informed Machine Learning in Nonlinear Model Predictive Control for Robotic Manipulators

Asa Paparo

Abstract

Existing methods of robotic motion planning rely on numerical optimization to generate control actions within constraints. One of the most successful methods of control has been Model Predictive Control (MPC), which continuously replans a trajectory by solving an online optimization problem. This means that for high dimension dynamical systems, the computational cost of MPC can rapidly become prohibitive. Physics Informed Neural Networks (PINNs) allow for the data driven creation and solving of high dimensional differential equations. Existing research has made progress in combining them, but lacks rigor and code optimization. In this paper, I develop an efficient NMPC implementation that relies on a PINN to plan movements. Then, I set up and integrate the controller with a state of the art simulation to demonstrate its effectiveness for a 7 DoF robotic arm. This development presents a significant advancement for fields that employ mobile robotics, such as manufacturing and search/rescue.

1 Introduction

Robotic manipulation is an area of high importance as global manufacturing increasingly relies on automation to accomplish tasks that would typically be done by humans. Advances in robotics and machine learning promise to be integral in the fourth industrial revolution, improving efficiency and worker safety [6]. Specifically, the advancement and effectiveness of industrial robotic systems is central to a developed nation's prosperity and economy. Due to high domestic costs and restrictions, offshore manufacturing is utilized by companies seeking to minimize cost of production. To improve economic conditions without worsening worker conditions, the United States must rely on robotic manufacturing [4]. Additionally, search and rescue presents a major opportunity for robotics to improve by making it safer and more efficient [1]. For medicine, surgical robots can accomplish tasks with more precision and safety than their human counterparts [**2**, **3**, **4**, **5**].

A primary goal of the field of control theory is the efficient and constrained control of complex dynamical systems. A highly popular approach for this is Model Predictive Control, which relies on continuously replanning a trajectory over a specific horizon [3].

2 Methods

2.1 Nonlinear Model Predictive Control (NMPC)

Model predictive control [2] and its nonlinear variation rely on an algorithm of several steps.

1. Create an optimized trajectory from current to target states over the specified horizon
2. Follow the first discrete state within the larger trajectory
3. Repeat with the new state

Derivation from [7]

$$J(x,u) = \int_0^\infty L(x(t),u(t),t)dt. \tag{1}$$

$$\begin{gathered} x^\star, u^\star = \arg\min_{x,u} J(x,u) \text{ such that} \\ \dot{x}(t) = f(x(t),u(t)) \text{ for all } t \quad \text{(Dynamics constraint)} \\ x(0) = x_0 \quad \text{(Initial state)} \end{gathered} \tag{2}$$

2.2 Physics Informed Neural Networks (PINNs)

2.3 Simulation

To create an effective simulation for system, I use Drake [5], a framework for controlling and simulating robotic systems. The specific system I control is a 7 degrees of freedom KUKA iiwa arm, documented in MIT Manipulation [8]

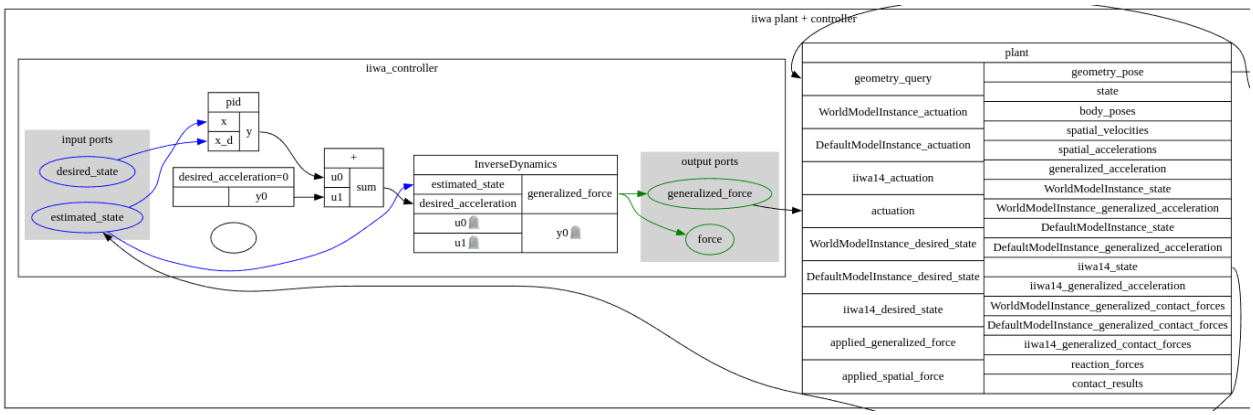

Figure 1: A flow diagram of the PID controller and system.

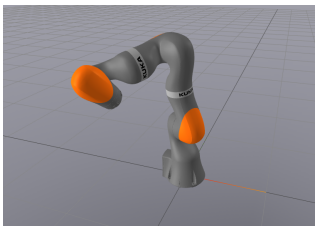

Figure 2: A visualization of the simulation, using Meshcat and the iiwa14 urdf file.

```
meshcat = StartMeshcat()
def run_simulation():
    meshcat.Delete()
    meshcat.DeleteAddedControls()

    builder = DiagramBuilder()

    # Adds both MultibodyPlant and the SceneGraph, and wires them together.
    plant, scene_graph = AddMultibodyPlantSceneGraph(builder, time_step=1e-4)
    # Note that we parse into both the plant and the scene_graph here.
    iiwa_model = Parser(plant, scene_graph).AddModelsFromUrl(
        "package://drake/manipulation/models/iiwa_description/sdf/
    iiwa14_no_collision.sdf"
    )[0]
    plant.WeldFrames(plant.world_frame(), plant.GetFrameByName("iiwa_link_0"))
    plant.Finalize()

    # Adds the MeshcatVisualizer and wires it to the SceneGraph.
    visualizer = MeshcatVisualizer.AddToBuilder(builder, scene_graph, meshcat)

    # pid controller
    kp = [100] * plant.num_positions()
    ki = [1] * plant.num_positions()
    kd = [20] * plant.num_positions()
    iiwa_controller = builder.AddSystem(
```

```
        InverseDynamicsController(plant, kp, ki, kd, False)
    )
    iiwa_controller.set_name("iiwa_controller")

    # Connect ports
    builder.Connect(
        plant.get_state_output_port(iiwa_model),
        iiwa_controller.get_input_port_estimated_state()
    )
    builder.Connect(
        iiwa_controller.get_output_port_control(),
        plant.get_actuation_input_port()
    )
    diagram = builder.Build()
    diagram.set_name("with iiwa controller")

    context = diagram.CreateDefaultContext()
    plant_context = plant.GetMyMutableContextFromRoot(context)
    q0 = np.array([-1.57, 0.1, 0, -1.2, 0, 1.6, 2])
    x0 = np.hstack((q0, 2 * q0))
    plant.SetPositions(plant_context, q0)
    iiwa_controller.GetInputPort("desired_state").FixValue(
        iiwa_controller.GetMyMutableContextFromRoot(context), x0
    )
    print(context)
    # plant.SetPositions(plant_context, [-1.57, 0.1, 0, -1.2, 0, 1.6, 0])
    # plant.get_actuation_input_port().FixValue(plant_context, np.zeros(7))

    simulator = Simulator(diagram, context)
    simulator.set_target_realtime_rate(1.0)

    meshcat.StartRecording()
    simulator.AdvanceTo(10.0)
    meshcat.StopRecording()
    meshcat.PublishRecording()

    display(
    SVG(pydot.graph_from_dot_data(diagram.GetGraphvizString())[0].create_svg())
)
```

This code

3 Discussion

In the future,

4 Conclusion

References

[1] Yugang Liu and Goldie Nejat. *Robotic Urban Search and Rescue: A Survey from the Control Perspective*. 2013.

[2] Yuval Tassa, Nicolas Mansard, and Emo Todorov. "Control-limited differential dynamic programming". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1168–1175. DOI: 10.1109/ICRA.2014.6907001.

[3] Michael G. Forbes et al. "Model Predictive Control in Industry: Challenges and Opportunities". In: *IFAC-PapersOnLine* 48.8 (2015). 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015, pp. 531–538. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.09.022. URL: https://www.sciencedirect.com/science/article/pii/S2405896315011039.

[4] Sridhar Kota and Thomas C. Mahoney. *Manufacturing Prosperity: A Bold Strategy for National Wealth and Security*. Tech. rep. MForesight: Alliance for Manufacturing Foresight, 2018. URL: https://deepblue.lib.umich.edu/handle/2027.42/145156.

[5] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: https://drake.mit.edu.

[6] Mohd Javaid et al. "Substantial capabilities of robotics in enhancing industry 4.0 implementation". In: *Cognitive Robotics* 1 (2021), pp. 58–75. ISSN: 2667-2413. DOI: https://doi.org/10.1016/j.cogr.2021.06.001. URL: https://www.sciencedirect.com/science/article/pii/S2667241321000057.

[7] Kris Hauser. *Robotic Systems*. University of Illinois at Urbana-Champaign, 2023.

[8] Russ Tedrake. *Robotic Manipulation. Perception, Planning, and Control*. 2023. URL: http://manipulation.mit.edu.