2024 7th International Conference on Soft Robotics (RoboSoft)
San Diego, CA, USA. April 14-17, 2024

Model Evolutionary Gain-Based Predictive Control (MEGa-PC) for Soft Robotics*

Spencer Jensen, John L. Salmon, Marc D. Killpack

Abstract— **This paper details a reliable control method for highly nonlinear dynamical systems such as soft robots. We call this method model evolutionary gain-based predictive control or MEGa-PC. The method uses an evolutionary algorithm to optimize a set of controller gains via model predictive control. We demonstrate the performance of MEGa-PC in simulation for a single-link inverted pendulum and a three-link inverted pendulum, and on physical hardware for a three-joint continuum soft robot arm with six degrees of freedom. MEGa-PC is compared to prior work that used Nonlinear Evolutionary Model Predictive Control or NEMPC. The new method performs similarly to NEMPC in terms of accumulated cost over the entire trajectory, however, MEGa-PC generalizes better to real-world applications where safety is paramount, the dynamic model is uncertain, the system has significant latency, and where the previous sampling-based method (NEMPC) resulted in significant steady-state error due to model inaccuracy.**

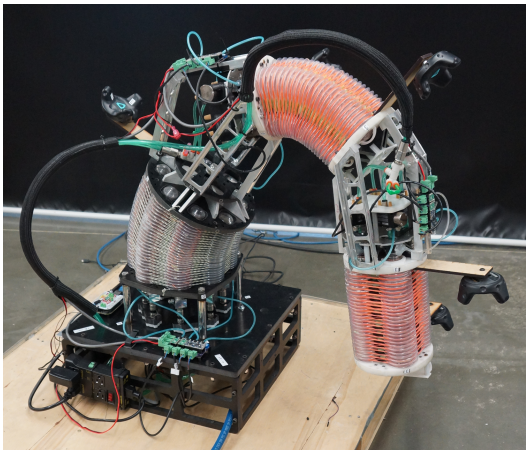

Fig. 1. The three-link, three-joint soft continuum robot arm used to test the control method introduced in this work. Each joint of the robot is able to apply torque about two different axes. The joint angles are estimated using HTC Vive trackers placed at top and bottom of each joint.

I. Introduction

Model-based optimal control methods for soft robots can enable more dynamic and aggressive performance. Specifically, model predictive control (MPC) is a class of optimal control algorithms that use a dynamic model to forward predict how different inputs will affect a system's state over time. However, this method often relies on an accurate dynamic model (which is difficult for soft robots) and can be intractable for complex models. Instead of solving for direct control inputs at every time step, the method presented in this paper uses a nonlinear model predictive controller formulation to generate a gain matrix that optimizes performance over a given time horizon. This method of generating a gain matrix is especially advantageous and potentially more robust because of the inherent uncertainties present when developing model-based optimal controllers for soft robots.

We refer to the method in this paper as model evolutionary gain-based predictive control (MEGa-PC) and it is a direct improvement upon nonlinear evolutionary model predictive control (NEMPC) (see [1]). Specifically, MEGa-PC generates low-level controller gains instead of direct actuation inputs. Generating a gain matrix instead of direct control inputs results in several additional benefits as outlined next.

First, MPC for complex, high-dimensional systems (such as soft robots) can be computationally intensive and may not find a solution (depending on constraints, initial conditions, and nonlinear cost functions). If the high-level optimization controller finds a poor local solution or does not converge for any reason, using a gain matrix allows the low-level controller to ensure that the system stays stable and performs the desired task. This can be critical for applications such as the task described in [2], where we plan to use a large-scale soft robot such as the one presented in this paper for the application of carrying a stretcher with a human partner.

Second, the high-level optimization for MEGa-PC can maintain stable control while running at slower update rates than past work, because the resulting gain matrix can be utilized in a high-rate, low-level controller. Because we can run the MPC algorithm at a lower rate, this allows additional time for MEGa-PC to generate the gain matrix at each iteration, thus allowing a longer time horizon, evaluation of a more complicated nonlinear model, or finding a more optimal trajectory. The control method can also be easily tailored to different low-level control laws. For example, in this work, we implement two different low-level control laws to adapt to different scenarios and overcome different problems such as model inaccuracies causing steady-state error.

In this paper, we show that MEGa-PC gives similar performance in simulation scenarios when compared to NEMPC. However, we also show that MEGa-PC outperforms NEMPC when real-world effects are added such as latency, optimization failure to converge, and model inaccuracies for three different systems, including a continuum joint, three-link soft robot arm as seen in Figure 1. These systems were chosen because they all have nonlinear dynamics and are the same systems as those analyzed in [1].

*This research was supported by the National Science Foundation (National Robotics Initiative) under grant number 2024792.

Mechanical Engineering Department, Brigham Young University (Provo, UT, USA)

2024 IEEE 7th International Conference on Soft Robotics (RoboSoft) | 979-8-3503-8181-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ROBOSOFT60065.2024.10521954

II. Related Works

This work builds most directly on another sampling-based MPC algorithm as described in [1] in which the authors show that nonlinear evolutionary model predictive control (NEMPC) performed better than other comparable state-of-the-art optimal control methods, namely Model Predictive Path Integral (MPPI) control and Differential Dynamic Programming (DDP) on a single-link and triple-link inverted pendulum. *Because NEMPC outperformed MPPI and DDP*, in this paper, we focus on comparing model evolutionary gain-based predictive control (MEGa-PC) to NEMPC as a direct benchmark.

Hyatt et al.'s work is similar to [3], [4], and [5]. These works focus on different machine-learned models but all use the same evolutionary model predictive control format where the optimization produces an optimal trajectory input. Abughalieh and Alawneh, [6], give an overview of the then-current parallelized or sampling-based MPC techniques.

Iterative linear quadratic regulator (iLQR) [7], a variant of DDP, is similar to this work in that it uses the model to output a gain instead of specific control inputs. Many papers use this method as a baseline state-of-the-art controller in addition to extending its utility by adding elements of machine learning [8], [9], [10]. By comparison, because of required assumptions, iLQR may only be able to find a local optimum, whereas MEGa-PC has the ability to at least perform a global search over the entire input space (albeit without guarantees of finding a global solution). In other words, MEGa-PC can explore the entire multi-modal or non-linear function space heuristically, through the implementation of a genetic algorithmic process, whereas iLQR finds the optimal in a local region by approximating and linearizing the local space. In addition, iLQR should perform the same or worse than DDP (see [11]) and we have already seen that NEMPC performs better, in terms of lower realized cost in simulation, than DDP [1].

In [12] the authors developed a method that is the most similar to MEGa-PC because they use a genetic algorithm to optimize the cost function weightings for a linear quadratic regulator (LQR) and augment an LQR with an integrator. In contrast to this controller, MEGa-PC does not require a linearized model or a quadratic cost. By enforcing the constraints of a linearized model and quadratic cost function on a nonlinear model, the solution provided by LQR will likely be less performant.

MEGa-PC is also similar to the concept of gain-scheduling (see [13]), where gains are selected for different areas of the workspace, using interpolation between these areas to achieve acceptable performance. One of the significant drawbacks of a gain-scheduling algorithm is the tedious process of tuning a controller for each part of the workspace. In contrast, MEGa-PC only has a single cost function to tune, but can output different gains depending on the current state and goal state.

Colombo and Da Silva in [14], compare gain-scheduled LQR with model predictive control (MPC). Although it was more difficult to definitively quantify stability for MPC, they found that MPC had higher performance in terms of RMS error. One of the main ideas of MEGa-PC is to combine the high performance of MPC with the straightforward stability verification of a state feedback control law.

III. Methods

A. Model Evolutionary Gain-Based Predictive Control

The essence of Model Evolutionary Gain-Based Predictive Control (MEGa-PC) is a genetic algorithm working to optimize a set of gains for a low-level controller by sampling the future predicted states of the system. At every time step, the predicted future states are propagated forward in time (using either a learned discrete-time model, or a discretized analytical model), starting with the current measured states. The low-level gain matrix that results from the MPC optimization is then used to multiply the current states and generate the control input at the next time step. Figure 2 shows a process flow diagram for NEMPC and MEGa-PC. Because [1] provides an in-depth description of the genetic algorithm used to solve the MPC problem, in this paper we present only the form of the optimization. The optimization problem can be expressed in equation form as:

$$\begin{aligned} \underset{K}{\text{minimize}}\ & J(x,u) \\ \text{subject to}\ & \vec{x}_{i+1} = f(\vec{x}_i, \vec{u}_i) \\ & \vec{u}_i = -K\vec{x}_i \\ & u_{min} <= \vec{u}_i <= u_{max} \\ \text{where}\ & i = t, t+1, t+2, ..., t+T \end{aligned} \tag{1}$$

Here, standard notation is used where x and u are the states and inputs of the system respectively, K is a $\mathbb{R}^{mxn}$ gain matrix, and J is a general cost function (not necessarily linear, or convex). Additionally, t is the current discrete time step being evaluated and T is the length of the time horizon (for both state prediction and control). Although we assume a state feedback control law where $u = -Kx$, any low-level control law that is gain-based would work for this algorithm. For example, on the soft robot arm, we instead use the cost function $u = K(x_{goal} - x) + u_{prev}$ which is essentially a state feedback control law with an integral term. For the cost function in this paper, we use a standard quadratic cost function as follows:

$$\begin{aligned} J(x,u) &= \sum_{i=t}^{T-1} \left[\tilde{x}_i^T Q \tilde{x}_i + \sum_{j=0}^{N-1} R_j \tilde{u}_{i,j} \right] + \tilde{x}_T^T Q_f \tilde{x}_T \\ \text{where}\ & \tilde{x} = x_{goal} - x \\ & \tilde{u} = u_{goal} - u \end{aligned} \tag{2}$$

where, Q, Q_f, and R are diagonal weighting matrices specific to each system, x_{goal} is the desired end state, and u_{goal} is the desired final input (if it is needed). Although we use a standard quadratic cost function in this paper for the purpose of familiarity and defining a simple benchmark, this control technique is not constrained to quadratic or even differentiable cost functions.

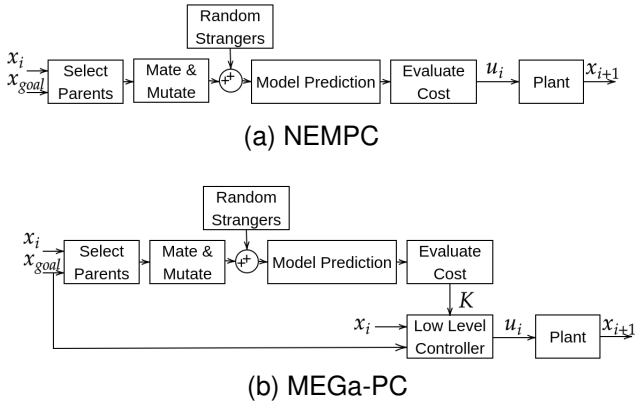

Fig. 2. Flow diagrams for NEMPC, **(a)**, and MEGa-PC, **(b)**. MEGa-PC outputs a gain matrix for a low-level controller whereas NEMPC outputs a direct actuation input. i refers to the current time step. At each new time step, the previous time step's x_{i+1} becomes the new x_i.

One of the significant ways that MEGa-PC improves upon NEMPC is that there is no need to parameterize the outputs of the controller such as in [1] where they pick several "knot points" between which they linearly interpolate over the time horizon. These knot points are not needed in MEGa-PC because the same gain is used for the entire horizon as opposed to NEMPC which defines a set of inputs, which could be different at each knot point. Although some parameterization may be beneficial for extremely complex or dramatically changing systems, the work in this paper has no time horizon parameterization of the gain matrix inside the controller. Expressed in another way, the controller has the task of finding the single optimal gain matrix for the current time horizon. However, because we are using model predictive control, or receding horizon control, we can also find a new gain matrix at every time step to adapt to changing situations or state estimation.

Generating a complete gain matrix, $K \in R^{m \times n}$, increases the required memory storage of the optimization script. However, this can be significantly reduced by only optimizing a small subset of the matrix elements. In this paper, this is done by optimizing only the diagonal elements of the matrix, which assumes zero cross-coupling between the states. Beyond this important difference in representing the control input (or optimization variables) as a set of gains for the output of this MPC algorithm, the procedure for forward prediction given the control input, the procedure for cost calculation, and the general optimization using a genetic algorithm follows the process from the authors' previous work.

It should be noted that NEMPC and MEGa-PC, as part of their common optimization method, rely on a large population (or sample) size that can be evaluated in parallel (multi-threading on a CPU/GPU) to produce optimal control trajectories that are similar in performance to other methods. This means that they are almost always run on a hardware-accelerated platform. The results detailed in this paper use Pytorch [15] for easy, rapid, parallelized execution on the graphics processing unit (GPU) although any hardware-accelerated linear algebra package would work.

B. Simulated and Hardware Test Systems

The three systems used to compare and evaluate MEGa-PC in this paper are: 1) the single and 2) triple-link inverted pendulum, and 3) a soft continuum robot arm. For forward propagation of the dynamic models of these systems, we use a discretized analytical model for both the single pendulum and the soft continuum arm while we use a discrete-time learned model (DNN) for the triple-link inverted pendulum.

The single-link inverted pendulum is modeled with standard dynamics of the form

$$I\ddot{\theta} + b\dot{\theta} - mg\sin(\theta) = \tau \tag{3}$$

where the moment of inertia, I, is calculated as ml^2 and θ, $\dot{\theta}$, $\ddot{\theta}$ are the angular position, velocity and acceleration respectively. The variables l, b, m, g, and τ are the length, viscous friction coefficient, mass, gravity constant, and applied torque, respectively which have the following values for our test cases: $1\ m$, $0.1\ \frac{Nm}{s}$, $0.2\ kg$, $9.81\ \frac{m}{s^2}$ and a maximum allowable torque of $1 Nm$. These values will result in a controller that has to drive energy into the system by swinging back and forth to be able to get to the upright position. We use a straightforward Runge-Kutta 4th-order method with an integration step size of 0.01 seconds for the single-link inverted pendulum for both simulation and control without hardware acceleration.

For the three-link inverted pendulum, we use the deep neural network (DNN) architecture as shown in [1] to approximate the dynamics to enable a direct comparison with their results. The DNN model outputs the next state of the system at a time step of 0.01 seconds. The original continuous-time dynamics (which were used to train the DNN model) for this system are of the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{4}$$

where the mass matrix, M, Coriolis matrix, C, and gravity vector, G, are calculated according to standard formulas and q, $\dot{q}$, and $\ddot{q}$ are a vector of each links' angular position, velocity, and acceleration. Each link in the three-link system is parameterized by a mass of $0.5\ kg$, a length of $0.5\ m$, a simplified moment of inertia of $0.1\ kg\,m^2$ and a maximum allowable torque of $10\,Nm$. The maximum allowable torque is such that the three-link system cannot swing or move directly to an upright configuration (similar to the single-link system).

The last system used to test and analyze MEGa-PC in this paper is the soft robot arm as seen in Figure 1. This arm has three inextensible, compliant, continuum joints that can each bend about the plane at the base of the joint defined by two orthogonal axes. This gives each joint two controllable degrees of freedom. This means that we model the soft robot arm as having six kinematic degrees of freedom. Each joint in the soft robot arm is actuated by either four or eight antagonistic pneumatic chambers. The pressure in each

chamber can be controlled to change the differential pressure between each pair of antagonistic chambers. This gives the system an 18-dimensional state space and six-dimensional input space (two differential pressures for each joint). The controller is tested on the soft robot hardware to show that this control method works in the physical world as well as in simulation. Soft robot arms such as the one used in this paper exhibit extremely complex dynamic effects such as hysteresis, nonlinear friction and stiffness, fluid dynamics, and coupling effects. There are many ways to model these robots, but none are as simple and accurate as models for traditional rigid robots. The analytical model that we use to predict the motion of this robot arm for control is equivalent to the dynamic analysis given in [16]. The model takes the standard form

$$M\ddot{q} + C\dot{q} + G = K_{prs}p - K_{spring}q - K_d\dot{q} \tag{5}$$

where q, $\dot{q}$, and $\ddot{q}$ are the joint angles, velocities, and accelerations of the arm. M, C, and G are the mass matrix, Coriolis matrix, and gravity vector. K_{prs} is the pressure to torque mapping. $K_{spring}q$ accounts for the parasitic spring-like force that causes the joint to settle at an equilibrium position, and $K_d\dot{q}$ is the damping force for modeling frictional effects. This model is complicated enough that the DNN architecture used to model the three-link system struggles to capture the dynamic effects for this soft robot arm. To enable controller comparison with this model, we use the Tustin or bilinear discretization to get a discrete-time affine model at each time step [17]. This model is then used to approximate the system for the duration of the control horizon and is evaluated on the GPU for rapid parallelized computation.

IV. Results

A. Inverted Pendulum

To begin with, the new model evolutionary gain-based predictive controller (MEGa-PC) is compared to NEMPC on the inverted pendulum system (where the state x is θ and $\dot{\theta}$, and u, the inputs, is torque τ). Because the pendulum system is relatively simple, the cost function is also simple with

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q_f = \begin{bmatrix} 0 & 0 \\ 0 & 5 \end{bmatrix}, \quad R = \begin{bmatrix} 0 \end{bmatrix}, \tag{6}$$

where the parameters above are manually tuned.

The zero weighting on velocity in both Q and Q_f (top left corner of the matrices) are because we do not want to penalize high velocities in this case, but rather we want the pendulum to get to its final position as quickly as possible. Figure 3 shows the pendulum being controlled to the vertical position, using NEMPC and MEGa-PC with the exact same parameters. NEMPC and the high and low-level controllers in MEGa-PC are run at 100 Hz. Both controllers have a population of 1000 model prediction samples, each with a horizon length of 50 time steps. As can be seen, the two methods yield very similar results. This is confirmed in their almost identical integral of time-weighted absolute error

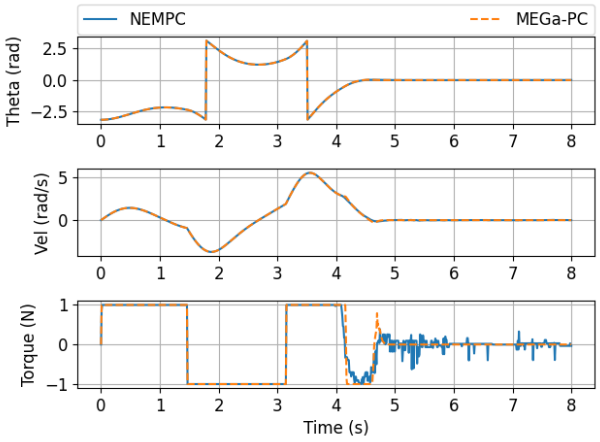

Fig. 3. From top to bottom, graphs of the positions, velocities, and applied torques of the inverted pendulum system controlled by NEMPC and MEGa-PC with identical parameters. The controllers visually perform almost identically. NEMPC is shown in blue and MEGa-PC is shown in orange. One difference of note is that MEGa-PC has smooth input transitions at steady state in comparison with NEMPC. The objective, or target angle and velocity, in all cases, is 0 rad and 0 rad/s respectively which is not shown in figures to increase legibility.

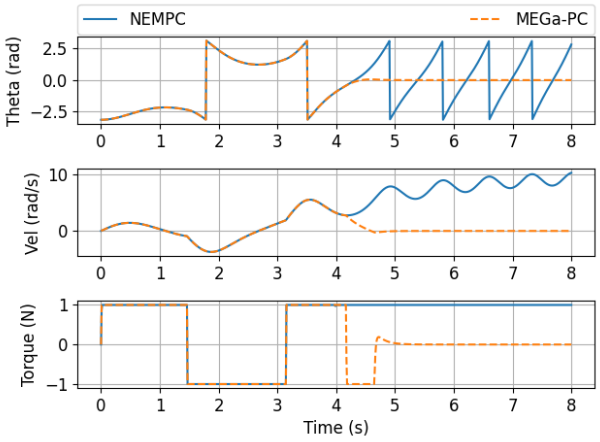

Fig. 4. From top to bottom, graphs of the angular positions, velocities, and applied torques of the inverted pendulum system controlled by NEMPC and MEGa-PC with identical parameters. Both optimization controllers are stopped at 4 seconds, after which the control inputs change with respect to a constant gain (For MEGa-PC) and constant torque for NEMPC). It should be noted that the linear low-level controller for MEGa-PC (which uses the last K matrix generated by the controller) is still running at every time step.

(ITAE) values of 16.40 and 16.35 for NEMPC and MEGa-PC respectively, which is a difference of 0.3 percent in ITAE.

Figure 4 shows the same inverted pendulum system as Figure 3. However, in contrast to Figure 3, Figure 4 shows the systems when the optimization controller is stopped after four seconds, after which the control inputs change with respect to a constant gain (For MEGa-PC) and constant torque for NEMPC). The motivation of stopping the two controllers is to show the effect of using a gain-based controller in the case of a current optimization becoming infeasible, or not converging to a good solution.

The inverted pendulum, being controlled by NEMPC,

TABLE I

A COMPARISON OF THE TOTAL ACCUMULATED COST OF NEMPC, MEGA-PC, AND MPPI FOR 500 TRIALS. EACH TRIAL IS PERFORMED THE SAME EXCEPT FOR A DIFFERENT INITIAL RANDOM SEED. SD STANDS FOR STANDARD DEVIATION.

Controller Costs	Median	Mean	Min	Max	SD
NEMPC	966	968	947	1069	12.35
MEGa-PC	1031	1031	937	1244	33.08
MPPI	1059	1060	1019	1149	18.44

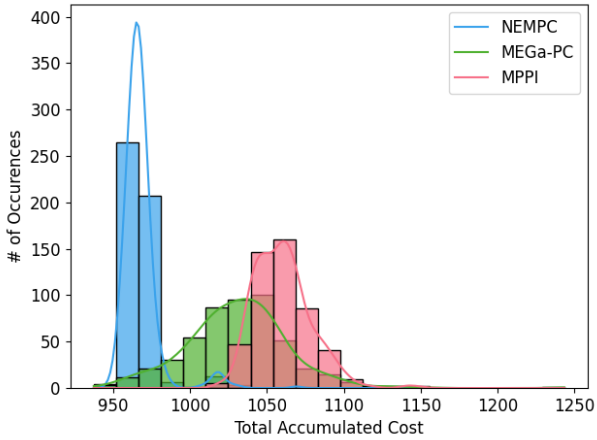

Fig. 5. A histogram of 500 different runs for the three-link inverted pendulum under MPPI, MEGa-PC and NEMPC. NEMPC is shown in blue, MEGa-PC is shown in green, and MPPI is shown in pink. Each run is exactly the same except for the seed of the random number generator used in the controller optimization. MEGa-PC seems to be more sensitive to the initial seeding, despite finding the best overall minimum.

begins to spin continuously in an uncontrolled manner. However, MEGa-PC maintains its performance and this is supported by the ITAE values of 49.01 and 16.39 for NEMPC and MEGa-PC respectively. This is possible because a single gain matrix is able to maintain performance over a larger range of angles as the pendulum approaches the goal state. Indeed, the ITAE performance confirms that the gain matrix towards the end of the trajectory when the optimization stops is already very close to being optimal. The modified trajectory has an ITAE value of 16.39 versus the original ITAE value of 16.35, a difference of 0.2 percent, where MEGa-PC comes up with a new gain matrix at every step. It should be noted that if MEGa-PC is stopped much sooner than four seconds, it also cannot reach the goal orientation since the pendulum is not close enough to its final goal state. However, the system does not tend to go unstable with a fixed gain matrix (unlike in the case of NEMPC). To encourage the system to remain stable at all times, a cost on any positive, real eigenvalues of the matrix $A - BK$, was added to the cost function. This should help avoid introducing instability.

B. Three-link Inverted Pendulum

To further compare NEMPC and MEGa-PC, we evaluated the two controllers on the three-link pendulum which triples the dimension of the state space. The cost function for NEMPC and MEGa-PC was again identical and the weights for the weighting matrix, Q, were set at 3, 2, and 1 for position. These same weights were retained for the final weighting matrix, Q_f, and a cost in R is also set on each of the three inputs equal to 0.0001. The high-level MPC controllers are nominally run at 100 Hz with a horizon length of 50 time steps.

For the three-link inverted pendulum, the seed for the random number generator which dictates the initial population of input trajectories, strangers, and mutation noise during the optimization for NEMPC and MEGa-PC had a noticeable effect on the final cost. The effect of this randomness is quantified in Table I which shows the median, mean, minimum, maximum, and standard deviation for the accumulated cost over 500 different trials. NEMPC performed slightly better on average than MEGa-PC, while MEGa-PC found a control policy that gave a lower overall minimum cost.

In addition, statistics for model predictive path integral (MPPI) control were added to Table I to give a better idea of the relative performance difference between NEMPC and MEGa-PC. MPPI was specifically selected as Hyatt and Killpack [1] found that NEMPC performed better than MPPI, and both MPPI and NEMPC performed better than differential dynamic programming (DDP). We use the same MPPI controller from [1] based on the method presented in Williams et al. [18]. MPPI was given the same horizon length, control rate, number of samples, and cost function as NEMPC and MEGa-PC. Figure 5 shows a histogram of the accumulated cost over the entire trajectory for the three controllers for the same set of 500 different random initial seeds. Overall, NEMPC performs the best with a mean difference of 6.5 % below MEGa-PC and 9.5 % below MPPI. We expect that with a better initialization algorithm such as latin hypercube sampling or a better matrix parameterization, the MEGa-PC samples would move further left in Figure 5. This sampling is particularly promising because MEGa-PC does find the overall minimum cost of the three controllers at 937. However, improvements in the quality of sampling-based methods remain an open question. Still, even if MEGa-PC were to always perform with slightly higher actual cost than NEMPC, one of the main benefits is robustness to the issues with both the optimization convergence, and errors in the dynamic model (given that the output is a gain-based controller that should be more robust than a direct torque or pressure control policy). The seeds for NEMPC and MEGa-PC that gave the smallest cost were selected for the remainder of this paper.

Figure 6 shows graphs of the three-link inverted pendulum under control by the NEMPC algorithm and the MEGa-PC method. For these trajectories, the accumulated cost is 947 and 937 for NEMPC and MEGa-PC respectively. The two algorithms perform similarly, reaching the target at approximately the same time despite generating different trajectories in the global search space to reach the goal state.

Figure 7 shows the same three-link pendulum under con-

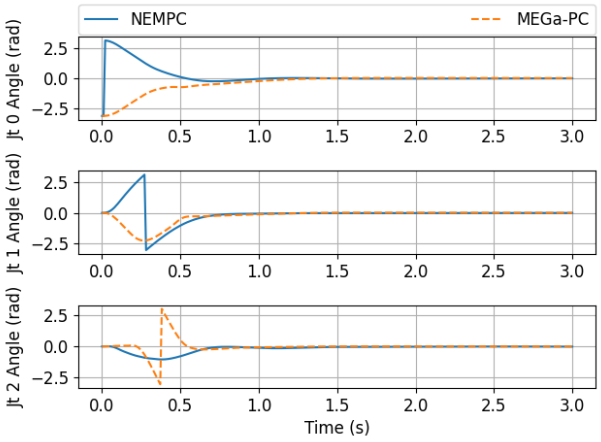

Fig. 6. Graphs of the joint angles of the three-link inverted pendulum. These plots show a comparison between the lowest cost trajectories from NEMPC and MEGa-PC with the same cost functions. NEMPC is shown in blue and MEGa-PC is shown in orange. The controllers perform similarly in terms of rise and settling time, as well as in terms of accumulated cost.

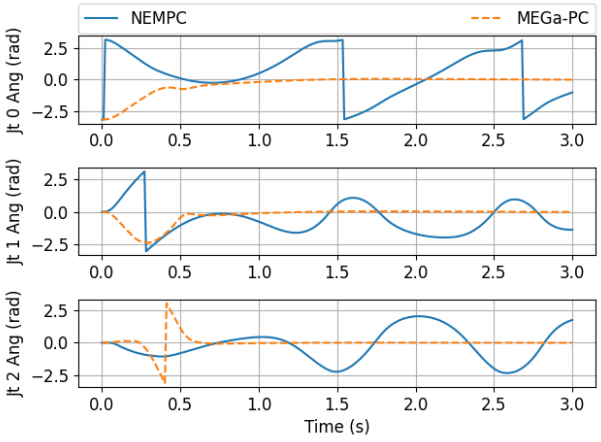

Fig. 7. Graphs of the joint angles of the three-link inverted pendulum under the control of NEMPC and MEGa-PC with the same cost functions. Both optimization controllers are stopped at a half-second and are no longer able to change their output. It should be noted that the linear low-level controller for MEGa-PC is still running at every time step.

trol of NEMPC and MEGa-PC. The optimization portion of both controllers is stopped at a half-second into the trial, although after that time, the current gain matrix from MEGa-PC is still used to generate state-dependent control inputs. NEMPC's past control output is not sufficient to give good performance, resulting in a total accumulated cost of 4461. The low-level portion of the MEGa-PC controller still achieves a total accumulated cost of 955 versus the optimal accumulated cost of 937 based on the controller running continuously. MEGa-PC is also able to control the pendulum, even though the optimization script stopped prematurely at 0.5 seconds. This again shows that MEGa-PC finds gains that function for a wide range of the state space. We note that if the optimization script is stopped sooner than 0.5 seconds, MEGa-PC is also unable to perform the desired

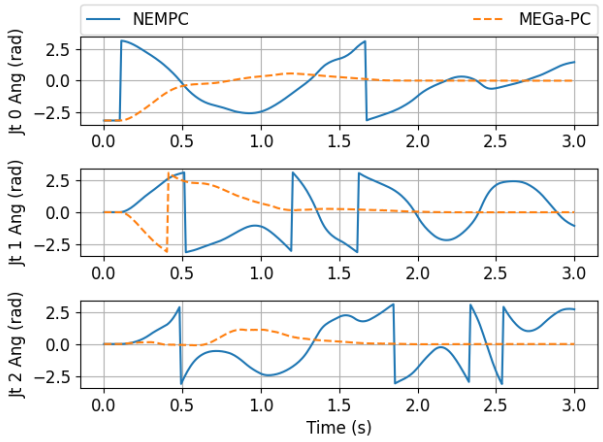

Fig. 8. Graphs of the joint angles of the three-link inverted pendulum under the control of NEMPC and MEGa-PC with the same cost functions. NEMPC is shown in blue and MEGa-PC is shown in orange. Both controllers run at a significantly reduced rate of 10 Hz. Importantly, the linear low-level controller for MEGa-PC ($u = -Kx$) is still running at every time step.

task. However, the gain matrix should at least ensure a stable response (which is not true with NEMPC).

To validate the hypothesis that a single gain matrix works well for a sufficiently wide range of the state space, Figure 8 shows the three-link inverted pendulum system running at a control frequency of 10 Hz, an order of magnitude lower than its previous rate of 100 Hz. MEGa-PC is able to perform the task successfully despite the drastically reduced rate of updating the K gain matrix, especially when compared to the resulting performance from NEMPC. MEGa-PC achieves an accumulated cost of 1572 versus the higher cost for NEMPC which was 5849. This is in part due to the fact that MEGa-PC is able to keep running low-level control at 100 Hz and only needs to output a gain matrix every 10 Hz to keep the system performing well.

Interestingly, this means that MEGa-PC could intentionally take longer to solve the optimal control problem in order to extend the control time horizon, evaluate a better model of the plant, calculate a more computationally expensive cost function, or increase the number of generations or population size in the genetic algorithm to achieve a more optimal control input trajectory. All of this would be possible without sacrificing performance because we have a low-level controller running at a high rate.

To further test the hypothesis that MEGa-PC can extend its control horizon to solve more challenging problems than NEMPC, we made the problem more complex by reducing the maximum allowable torque in the three-link system by one-fourth to 2.5 Nm. This significantly extends the required horizon length to be able to successfully predict and perform the swing-up task. If we reduce the control frequency from 100 Hz to 25 Hz, we can quadruple the horizon length to 200 time steps without increasing the latency of the controller. Figure 9 shows the behavior of the two control methods under these conditions. It should be noted that the random number generator seed again had a significant effect on the

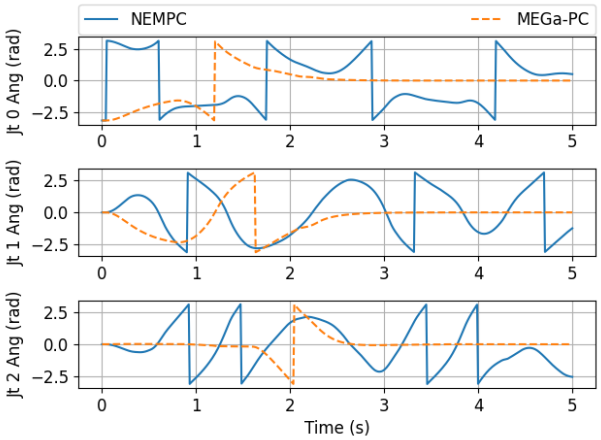

Fig. 9. Graphs of the joint angles of the three-link inverted pendulum under the control of NEMPC and MEGa-PC with the same cost functions. For this experiment the maximum allowable applied torque was cut by one-fourth. NEMPC is shown in blue and MEGa-PC is shown in orange. Both controllers are run at 25 Hz instead of 100 Hz which allows the optimizations to predict states four times as far into the future. MEGa-PC is still able to perform the task successfully, as opposed to NEMPC, as seen by closer match to the target objective (reaching the zero position for all three joints). The linear low-level controller for MEGa-PC is still running at every time step, which is one of the main benefits of this type of control.

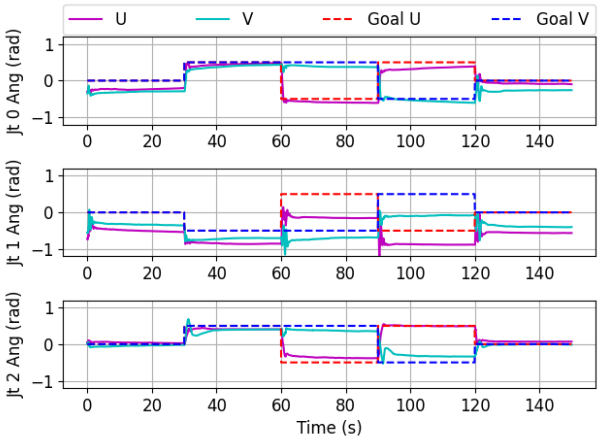

Fig. 10. Graphs showing the performance of NEMPC on for the soft robot. The plots show the rotation about the X and Y axis, denoted as U and V respectively, for each of the joints. Given the difficulty in modeling soft robot dynamics, the model of the arm that NEMPC uses to predict behavior results in significant steady state error.

cost and we selected the best seed out of 10 random seeds. As shown, MEGa-PC is able to complete the task while NEMPC attempts something that looks like it might work, but in fact, cannot make the proper adjustments in time to perform the task. For this test, MEGa-PC received an accumulated cost of 4336 compared to the cost of NEMPC which was 10 301. A video showing both the three-link and single-link inverted pendulum experiments can be seen here: `https://youtu.be/TvptLL6eHtE`.

C. Three-link, Six-DoF, Continuum, Soft Robot Arm

We also validated MEGa-PC on a physical soft robot arm to show this method works for real-world applications with complex, nonlinear dynamics. This further increases the difficulty of the problem by increasing the dimensionality of the system, and by dealing with the inherent model inaccuracies present in our current soft robot model. Figure 10 shows NEMPC controlling the three-DoF continuum-joint robot arm. Each row of plots shows the joint angles of each joint from joint zero to joint two in top-to-bottom order. The prediction model does not match the real system and thus suffers from significant steady-state error. The middle joint, referred to as joint one, specifically has unacceptable steady state error with a maximum error of approximately 65 percent of the step size.

To overcome the steady state error problem, MEGa-PC uses the control law $u = K(x_{des} - x) + u_{prev}$, which allows MEGa-PC to act like an integration scheme. Figure 11 shows the arm under the control of MEGa-PC with the updated control law. The system has zero steady-state error, while also performing well in terms of transient response. MEGa-PC is specifically tuned to get smooth non-oscillatory perfor-

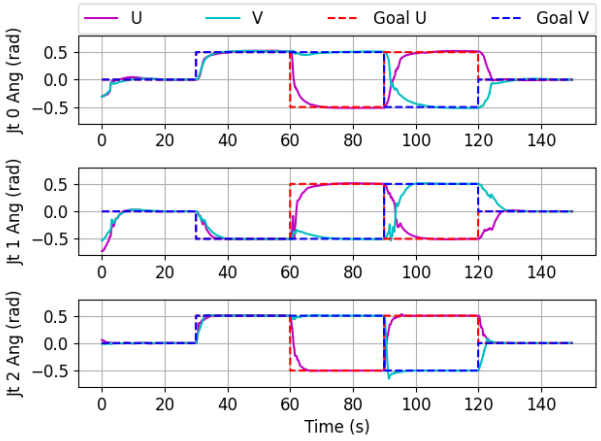

Fig. 11. Graphs showing the performance of MEGa-PC on the orange robot arm. The plots show the rotation about the X and Y axis, denoted as U and V respectively, for each of the joints. Because the model of the arm that MEGa-PC used has significant steady state error, we used the control law $u = K(x_{des} - x) + u_{prev}$. This allows the controller to perform well even with a suboptimal model of the system.

mance. During the tuning process, we found that multiplying the weighting R by the squared inverse of the position error drastically improved the oscillations and transient behavior of the system. This is in part because we set u_{goal} in the cost function (shown in Eqn. 2) for the soft robot arm problem to be u_{prev}. A video showing the arm under the control of MEGa-PC can be found at the following link: `https://youtu.be/AwVW288Czoo`.

The gains produced by MEGa-PC for the hardware experiment shown in Figure 11 are presented in Figure 12. MEGa-PC varies the gains intelligently by raising the values of the gains when the system is far away from the goal and lowering them to a steady state value when it gets close to the goal state. Of note are the potential issues that may be caused by the oscillation in these control inputs. However, in practice,

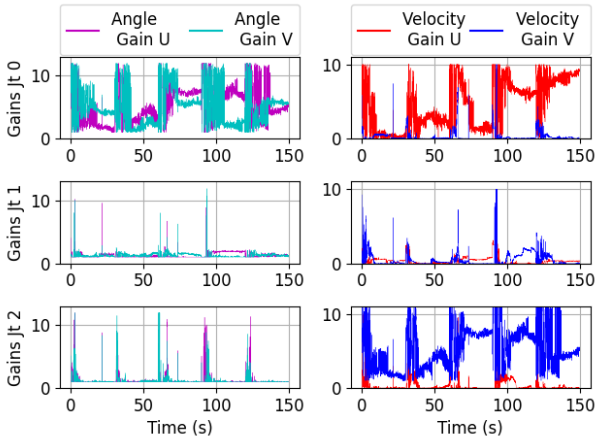

Fig. 12. Graphs show the gain values output by MEGa-PC during the hardware experiment with the soft robot. Each row shows the gains for each joint. The first column shows the gains on the joint angles and the second column shows the gains on the angular velocity of each joint.

this did not cause unwanted behavior. Slew constraints could also easily be added to the optimization to limit how quickly the gain matrix can change. Currently, we constrain the overall control input but do not limit the values for the K matrix directly.

V. Conclusion and Future Work

In section IV, we found that the initial random seed had a non-negligible effect on the performance of all three algorithms, but had a larger effect on MEGa-PC. We assume that this effect could be mitigated by starting with a better initial population method such as Latin hypercube sampling or by a better parameterization of the gain-matrix. Indeed, future work should consider the impact of using a non-diagonal gain matrix (and any required parameterization of that matrix) that could serve to better handle coupling effects for example. In addition, because this method does not require a quadratic or differentiable cost, exploration of more effective cost functions should also be explored.

In conclusion, MEGa-PC is shown to be more adaptable than NEMPC to model failures or model uncertainty for applications like soft robot control. It also performs well on complex control problems using the generated gain matrices to maintain performance using high-rate low-level controllers.

References

[1] P. Hyatt and M. D. Killpack, "Real-Time Nonlinear Model Predictive Control of Robots Using a Graphics Processing Unit," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1468–1475, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8954784/

[2] E. Mielke, E. Townsend, D. Wingate, and M. D. Killpack, "Human-robot co-manipulation of extended objects: Data-driven models and control from analysis of human-human dyads," *arXiv:2001.00991 [cs, eess]*, Jan. 2020, arXiv: 2001.00991. [Online]. Available: http://arxiv.org/abs/2001.00991

[3] A. Zhang, Z. Lin, B. Wang, and Z. Han, "Nonlinear Model Predictive Control of Single-Link Flexible-Joint Robot Using Recurrent Neural Network and Differential Evolution Optimization," *Electronics*, vol. 10, no. 19, p. 2426, Jan. 2021, number: 19 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/10/19/2426

[4] E. Ademir Morales Perez and H. Iba, "LSTM Neural Network-based Predictive Control for a Robotic Manipulator," in *2021 International Symposium on Electrical, Electronics and Information Engineering*, ser. ISEEIE 2021. New York, NY, USA: Association for Computing Machinery, Feb. 2021, pp. 128–135. [Online]. Available: https://doi.org/10.1145/3459104.3459127

[5] C. C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. D. Killpack, "Using First Principles for Deep Learning and Model-Based Control of Soft Robots," *Frontiers in Robotics and AI*, vol. 8, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2021.654398

[6] K. M. Abughalieh and S. G. Alawneh, "A Survey of Parallel Implementations for Model Predictive Control," *IEEE Access*, vol. 7, pp. 34 348–34 360, 2019, conference Name: IEEE Access.

[7] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system," *International Journal of Control*, vol. 80, no. 9, pp. 1439–1453, Sep. 2007, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207170701364913. [Online]. Available: https://doi.org/10.1080/00207170701364913

[8] G. Lee, S. S. Srinivasa, and M. T. Mason, "GP-ILQG: Data-driven Robust Optimal Control for Uncertain Nonlinear Dynamical Systems," arXiv, Tech. Rep. arXiv:1705.05344, May 2017, arXiv:1705.05344 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1705.05344

[9] Z. Cheng, J. Ma, X. Zhang, F. L. Lewis, and T. H. Lee, "Neural Network iLQR: A Reinforcement Learning Architecture for Trajectory Optimization," arXiv, Tech. Rep. arXiv:2011.10737, Aug. 2021, arXiv:2011.10737 [cs] type: article. [Online]. Available: http://arxiv.org/abs/2011.10737

[10] T. Zong, L. Sun, and Y. Liu, "Reinforced iLQR: A Sample-Efficient Robot Locomotion Learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 5906–5913, iSSN: 2577-087X.

[11] J. N. NGANGA, H. Li, and P. Wensing, "Second-order differential dynamic programming for whole-body mpc of legged robots," in *Embracing Contacts-Workshop at ICRA 2023*, 2023.

[12] M. H. Karim Roni and M. S. Rana, "Genetic Algorithm Based Data-Driven ILQR Control Scheme for Three-Phase Microgrid System," in *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Feb. 2021, pp. 1–6.

[13] K. Wu and G. Zheng, "Fem-based gain-scheduling control of a soft trunk robot," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3081–3088, 2021.

[14] F. T. Colombo and M. M. da Silva, "A comparison between gain-scheduling linear quadratic regulator and model predictive control for a manipulator with flexible components," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, p. 09596518221087805, Apr. 2022, publisher: IMECHE. [Online]. Available: https://doi.org/10.1177/09596518221087805

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[16] S. W. Jensen, C. C. Johnson, A. M. Lindberg, and M. D. Killpack, "Tractable and intuitive dynamic model for soft robots via the recursive newton-euler algorithm," in *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, 2022, pp. 416–422.

[17] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback control of dynamic systems*. Prentice hall Upper Saddle River, 2002, vol. 4.

[18] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *arXiv preprint arXiv:1509.01149*, 2015.