4M25- Final Report

System Identification and Model-Based Reinforcement Learning for Control of a Soft Robotic Manipulator

Candidate Number 5526D

Abstract

Soft robotics have a large variety of applications as a result of their desirable properties. However, these properties give most soft robotics infinite degrees of freedom and non-linearities that can prove challenging for control methods. This report presents a data-efficient method utilizing reinforcement learning (RL) methods to control a soft robotic pendulum. A neural network is deployed to learn the characteristic of the system from observation data. This then allows quick reinforcement learning without the need for high-volume testing on the physical system.

Introduction

Background and Motivation

Bio-inspired softness in robotics can offer many advantages over classical rigid designs, such as the utilization of system dynamics for more energy-efficient motion with minimal adverse disruption to the environment. As a specific example, soft surgical manipulators can be compliance matched to abdominal viscera to minimize the risk of damage to internal organs. This way the controller can work with deflections of the manipulator by the environment to achieve efficient motion, such as winding around obstacles.

The challenges of our soft robotic pendulum system can be split into two different categories, both stemming from the infinite degrees of freedom of a continuously soft manipulator. First, redundancy in the infinite state-space of the manipulator, and sensitivity to external influences (namely, gravity and contact forces), present difficulty in developing analytic kinematic models. Second, is the challenge of designing generalizable controllers with limited information on both the state of the manipulator and heterogeneity in the material properties of the manipulator and of the environment. We propose that statistical learning approaches may serve as fruitful research avenues for both problems.

Outline

A simple soft manipulator - a continuously soft rod with actuation only at its base - has been developed to explore these models. System identification is first employed, with full information and then with limited information, to derive the dynamics of the manipulator. Then a controller is trained via model-based RL with the objective of maintaining the manipulator in the upright position. This approach circumvents the inefficiency of collecting data for model-free reinforcement learning, by generating simulated data with the learned dynamics. The goal is that by training on simulated dynamics, reinforcement can achieve faster than real-time training, increasing the data efficiency. Both model-free and model-based approaches are compared to test this. Reinforcement learning is then also compared to the efficacy of traditional linear-quadratic regulator (LQR) control and proportional integral derivative (PID) control. As an external influence and comparison, there is PILCO ((Deisenroth & Rasmussen, 2011)), a control learning scheme based on Gaussian processes for system identification and policy search for control between trials.

A multi-pendulum system was made in simulation with torsional springs and dampers between rigid members to model softness. The information available to the neural network used for system identification was reduced to the angle at the base and the position of the end-effector relative to the base.

The control method is summarised below:

- Create a soft robotic pendulum using Simulink
- Gather observation data via motor babbling
- Learn system dynamics via a Neural network
- Utilize reinforcement learning to train a controller on the learned dynamics

- Compare model-free vs model-based approach
- Compare performance to more classical control methods
- Obtain data from a real-world setup and apply RL control methods

Methods

Simulation

To approximate a soft robot in simulation a pseudo rigid body (PRB) simulation in MathWorks Simulink (see Figure 1) was made to train our system before access to physical data. This facilitated rapid iteration of the design necessary for the reward function and accuracy of the controller further down the line.

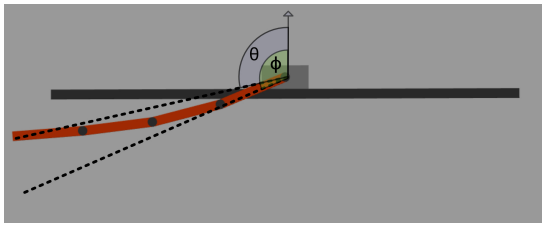

Figure 1: Pendulum system for N=4

The simulation of continuum manipulators is challenging due to the multiple degrees of freedom of the system and the non-linearity associated with soft materials. PRB models provide a numerical approximation of these flexible elements through a series of rigid links and rotational joints, for which torsional springs and dampers at each joint capture the material's compliance (Venkiteswaran, Sikorski, & Misra, 2019).

Each simulation outputs the tip coordinates (e_x, e_y), tip rotation (e_r), base rotation ϕ ,and their first derivatives for use in dynamics prediction. The simulation has several tunable parameters as summarised in Table 1.

Parameter	Description	Unit
L	Pendulum length	m
n	Number of rigid links	$-$
m	Pendulum mass	g
W	Link width	m
D	Link depth	m
ζ	Damping constant	$Nm/(deg/s)$
k	Spring constant	Nm/deg

Table 1: Simscape simulation parameters

System Identification

To learn the dynamics of the system, using the same method when observing real data, the state of the simulation was measured with random torque inputs (motor babbling). The variables recorded are the base angle (ϕ), the end effector angle (θ), and the position of the tip relative to the base (e_x, e_y). The full state vector is $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$, where $\mathbf{q} = [\phi, \theta, e_x, e_y]$. The dynamics model takes the state and the torque (τ) as inputs, and returns the state at the next time step: $\mathbf{f}(\mathbf{x}(t), \tau(t)) = \mathbf{x}(t+1)$.

Data was gathered by simulating the pendulum movements with a small time step, $dt = 0.01$, measuring the pendulum positions at each step, and then using numerical differentiation to get the first derivative of each observed variable. Under the assumption of the Markov Property (see equation 1), a neural network with 3 hidden layers was employed to learn the dynamics, each with 16 neurons. Multiple layers help approximate non-linear functions more accurately and efficiently.

$$p(s_{t+1}|s_t, \tau_t) \tag{1}$$

Motor babbling ran for 60 trials, each lasting 5 seconds to give 499 data points per trial. This gave 5 minutes of training data for our system. The model is reset between trials to prevent excessive motor velocities; the initialization torque is drawn from a normal distribution for the same reason. Data was predominantly gathered in the same space as that which the pendulum exists during the control task i.e. near the top equilibrium.

Reinforcement Learning

In order to balance the system in an upright position a controller is trained, using either the simulation system or using the neural network previously trained.

This was done using the Deep Q-network algorithm which learns an optimal policy for an environment. This policy determines discrete actions by using a deep neural network to approximate the optimal action-value function. As such DQN is suitable for the non-linear complex model we are attempting to simulate. DQN learns from raw sensory inputs making the process more generalizable across different tasks. Hence if it works to balance and control the pendulum vertically it should be applicable to any less complex tasks.

The agent's action space is defined as the continuous space $\tau \in [-\tau_{max}, \tau_{max}]$ acting on the motor where τ_{max} is the maximum allowed torque from the

system. The action is selected according to a policy $pi(\tau, s)$, defined as:

$$pi(\tau, s) = p(\tau_t = \tau | s_{\text{t-1}} = s) \quad (2)$$

In this equation, $pi(\tau, s)$ acts as a lookup table of states to actions. An action is selected according to

$$pi(\tau, s) \begin{cases} 1 \text{ if } \tau = argmax_\tau[\tilde{Q}(\tau, s; \psi)] \\ 0 \text{ otherwise} \end{cases} \quad (3)$$

In this $\tilde{Q}$ is the critic: an approximation of the Q-value function. If the agent has access to the precise Q-value function then the agent will receive the maximum reward. Ψ is the parameter that improves the critic's approximation. In DQN-learning critic improvement is achieved using a neural network to minimize mean squared error.

The reward function given to the agent is shown in equations 4 and 5.

$$r \propto min(max(cos(\theta)/|sin(\theta), 50), -50) + r_t^* \quad (4)$$

$$r_t^* \propto \begin{cases} 1 \text{ if } \theta_t < \varepsilon \\ r^*_{\text{t-1}} e_{\text{-kt}} \text{ otherwise} \end{cases} \quad (5)$$

As visualised in figure 2 2, the reward increases as the pole reaches the upright position. A bonus r^* is given if the pendulum end-effector is in the small region at the top. This small addition provides a rapid reward increase when the system reaches a precise position. This helps ensure the pendulum gets as close to the desired position without reaching a minima only close to the target. The parameter ε tunes the precision of this reward and k is the decay rate to prevent undesired spinning behavior.

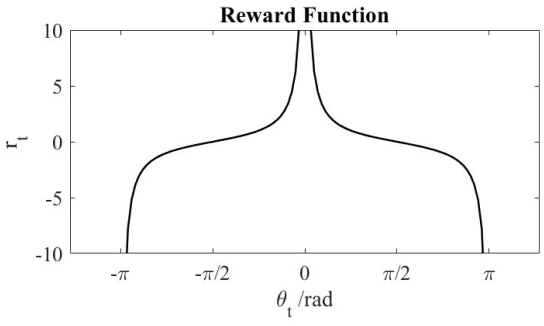

Figure 2: Reward function for DQN

Reinforcement Learning- Model Based

Utilizing the data from the trained neural network, a model-based approach can be taken with the goal of achieving similar performance with less training on the actual system. This was first done with the original network from system identification however later developments saw an improvement with using DQN-MBPO (Model based policy optimization). In the latter case, a base agent DQN explores the environment, optimizing the critic for value function approximation, while the MBPO agent uses a neural network for system identification. The architecture of this network is almost identical to the one established in 'system identification'.

The controller was trained with 9 simulated rollouts for every 1 real experience with the Simulink model. This regime helps reduce model bias as it is rectified by the inclusion of real Simulink experiences. Thus similar performance should be achieved with fewer episodes than the model-free approach.

Results

System Identification

As seen in Figure 3, the neural network accurately models the system dynamics. By minimizing the mean squared error (MSE) between the predicted next state and the actual next state the network achieved an error of $\approx 3e-9$. This performance is reflected in the rollout seen in Figure 3.

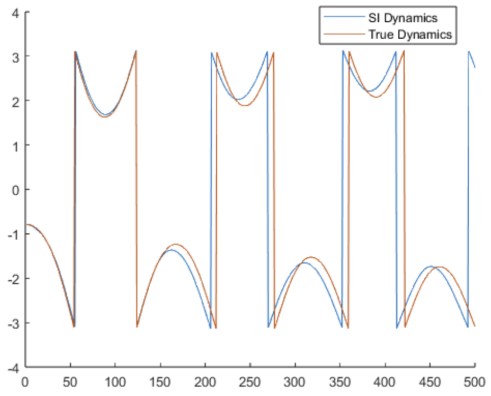

Figure 3: System identification compared to simulation rollout

Reinforcement Learning-Model Free

The model-free controller proved to successfully balance the pole reaching the asymptotic reward in approximately 150 episodes of training, equivalent to around 4 hours of simulated time. This can be seen by the blue line in Figure 4. This approach optimizes the controller with direct observations of the system.

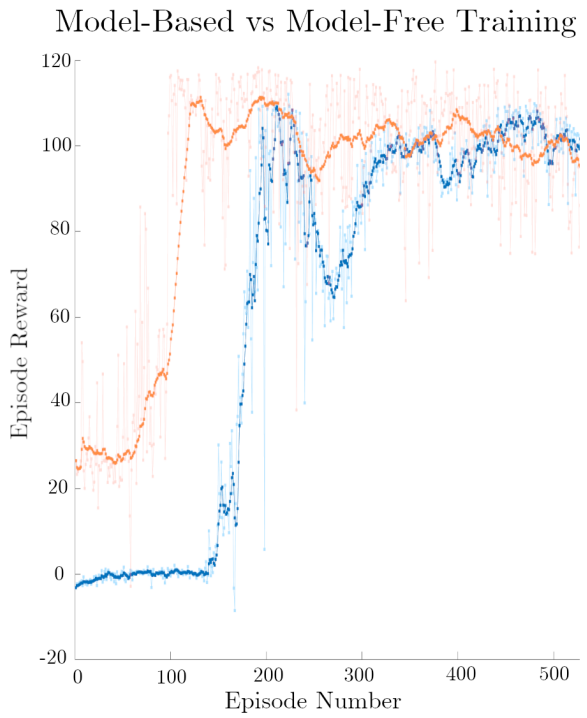

Figure 4: Model-Free vs Model-Based training episodes

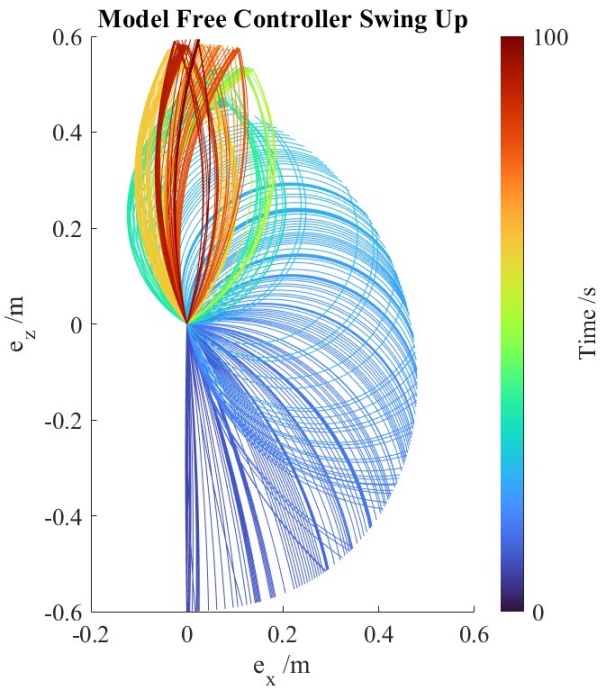

Figure 5: Time course of compliant rod swing under model-free RL control using constant curvature approximation

Typically the pendulum will reach steady state oscillatory behavior after 10 seconds. After swinging up the rod remained within a standard deviation of 0.2 radians from the upright position for an arbitrary time. To help with visualization of the motion Figure 5 shows the swing up and subsequent movement of the pendulum as it is balanced.

Model-Based reinforcement learning

Now utilizing the dynamics from system identification the model-based controller was trained on a combination of real and simulated experiences. The agent reached asymptotic performance in approximately 100 episodes of training as seen by the orange line in Figure 4 (2 hours 45 minutes of simulated time). As seen in Figure 6, the final performance is near-identical to the performance of the model-free controller, reaching a standard deviation of 0.2 radians from the upright position. Note that the time-averaging smooths rapid changes in error, thereby masking small differences in the swing-up velocity, where the model free-controller shows a slight advantage. Unlike with a purely system identification-driven controller the model-based approach has minimal model bias. Therefore this method offers savings in the overall time interacting with the environment but with an increased computational cost.

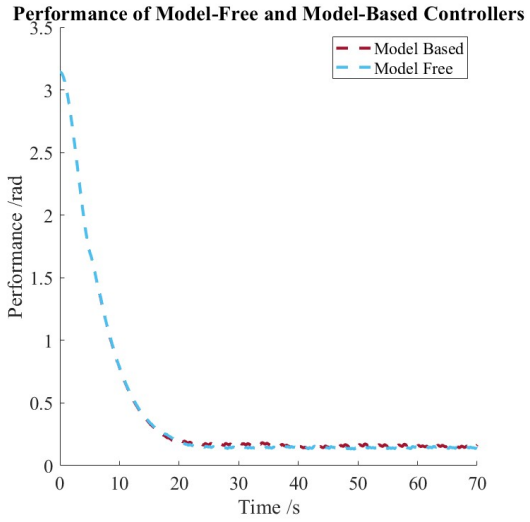

Figure 6: Performance of model-free and model-based RL trained controllers

Classical control methods

To establish the success of our controller we compared the performance with the alternate methods LQR and PID. These control methods, while only partially applicable to our problem, can provide some benchmarks. LQR should be optimal for a known system with limited energy input and PID without torque limits should be similarly quick.

LQR Linear Quadratic Regulator, as established in lectures, can be used for multi-link pendulum problems. In an attempt to test the feasibility of our problem, the system was tested against the lab provided by the lectures. By adjusting the initial starting position, target angles, and torque the method converges to a steady-state upright position using minimal energy input. Figure 7 shows how the LQR method compares with PID (discussed later). For this method, the pendulum starts closer to the final position as convergance becomes more difficult as the distance from the target increases for a non-linear system. This is because the method requires linearization about the state space and hence becomes less accurate for large jumps.

While the controller is robust on a fully known model, it is not suitable for our application for multiple reasons. It requires a full state-space model of the system (matrices A, B, C, D) so it will not perform simply with observed data of motor babbling with a physical system.The LQR method solves the algebraic Riccati equation associated with the linear quadratic cost function (using matrices Q and R) so a suitable cost function must be chosen. In the virtual lab provided the steady state associated with target torque and joint angles is coupled such that if incorrectly set can cause odd convergence and high torque behavior.

LQR may still be useful in conjunction with system identification. If we were able to match observed data to the parameters of a model and accurately estimate the system then an LQR controller could be derived for this system and applied to the true model.

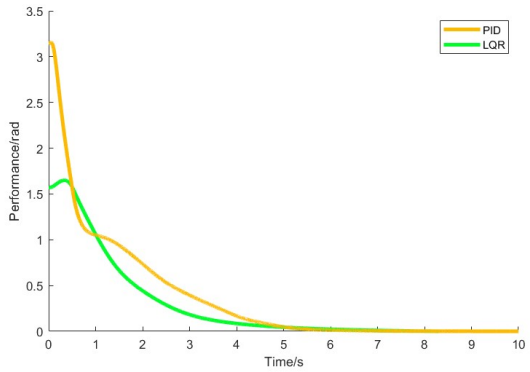

Figure 7: Error of $|\theta|$ for LQR and PID

PID The Proportional Integral Derivative control method is applicable to both linear and non-linear systems and worked with some success on our system. As the model is non-linear parameter optimization is required. As a result, optimal PID parameters change with input and target state. Furthermore, adding more links to the system increases the overall complexity making PID control hypersensitive to its parameters. The performance of PID can be seen in figure 7. However, a very high torque limit was given to the controller meaning it naturally moved the pendulum upright much quicker than the reinforcement-based methods.

For our application, PID parameters could be optimized for the model obtained from system identification and then applied to the real system. This enabled testing on the actual system to be avoided, whilst still yielding reasonable results, albeit to a limited degree.

Future Work

To coincide with the simulated system a physical model was made. Figure 8 shows the final design of the soft robot manipulator. The flexible beam is mounted to the drive shaft using a single 3D printed piece. This was fitted to a high precision to reduce any backlash introducing noise to the system. The beam itself is made from a flexible TPU which was iteratively designed to provide sufficient flex with the end mass was attached. A long and deep beam promotes bending in the minor axis removing perturbation in the unobserved plane (when using computer vision). The system itself also works with

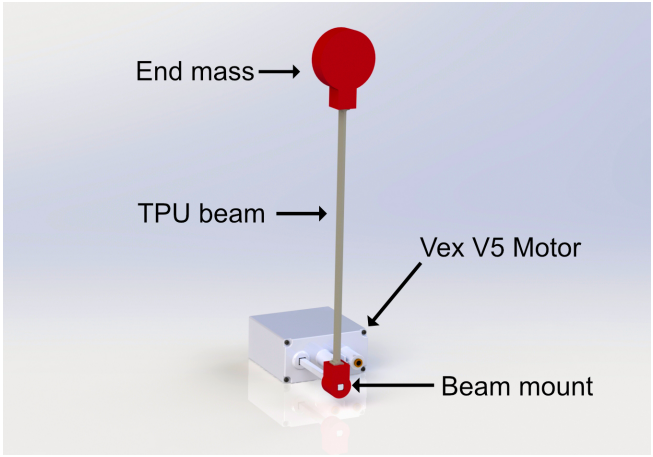

Figure 8: CAD Design

affine camera projection as all movement is within a single plane, this made computer vision an ideal method for determining the end-effector position. In the context of surgical manipulators, an IMU may be preferred.

Hardware Control

The system utilizes VEX hardware which can provide direct torque and motor position values to the controller. This enabled us to synchronize the time series data of motor position to the pixel coordinates of the pendulum. The controller was designed in Simulink and deployed on the VEX hardware via code generation. While the program is running, base angle phi is measured using the built-in rotary encoder.

Computer Vision

To obtain the end-effector position computer vision tracking was used. This was implemented on a raspberry pi module to remove possible sources of lag. To extract the end-effector co-ordinates (x, y) and angle θ a blue dot at the end of the pendulum was tracked using OpenCV dictated by Algorithm 1.

Algorithm 1 Extracting x, y coordinates

1: Capture camera frame
2: Extract regions within the specified colour range to form a binary image
3: Apply a 3x3 Gaussian Blur
4: Extract Contours
5: Sort contours by area size, and select the largest as the tracked object
6: Compute the center of the region to yield x, y coordinates in pixels.

Figure 9 shows object detection in practice. The data is time-stamped which allows synchronizing with the MATLAB readings to train system dynamics.

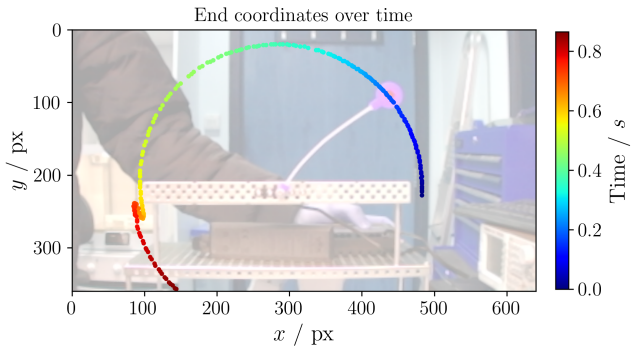

Figure 9: End-effector co-ordinate extraction with camera view overlay

Conclusion

Overall the reinforcement learning methods proposed provided an interesting insight into the possible control of soft robotics and performed well in balancing the system with limited torque. The model-based approach successfully converged its performance in fewer episodes than model-free RL showing the benefits of this method. There remains much potential for these models to be optimized to achieve a very high level of performance – something we were unable to see due to the initial shortened deadline of the 21st, which was extended too late for us to complete further work.

While not directly comparable with PILCO, it is worth noting some differences in observations. PILCO took less than 90 seconds of real experience to learn to stabilize the double pendulum cart model, with computation time between trials. While our method requires more time on the system, there is potential to reach values closer to this. A deeper and longer-trained neural network paired with increased simulation training time per episode would allow rapid convergance of the DQN algorithm on real simulation data without introducing significant model bias.

The generalizability of RL also makes it a useful solution in potentially more complex situations where traditional PID control would be much less suited. Adjustments to the reward function can also promote certain movement sequences and provide constraints to the system.

References

Deisenroth, M., & Rasmussen, C. (2011, 01). Pilco: A model-based and data-efficient approach to policy search. , 465-472. Retrieved from https://mlg.eng.cam.ac.uk/pub/pdf/DeiRas11.pdf

Venkiteswaran, V. K., Sikorski, J., & Misra, S. (2019). Shape and contact force estimation of continuum manipulators using pseudo rigid body models. *Mechanism and Machine Theory*, *139*, 34-45. doi: https://doi.org/10.1016/j.mechmachtheory.2019.04.008