Nonlinear Model Predictive Control for the Swing-up of a Rotary Inverted Pendulum

Sooyong Jung* and John T. Wen†

Abstract

This paper presents the experimental implementation of a gradient-based nonlinear model predictive control (NMPC) algorithm to the swing-up control of a rotary inverted pendulum. The key attribute of the NMPC algorithm used here is that it only seeks to reduce the error at the end of the prediction horizon rather than finding the optimal solution. This reduces the computation load and allows real-time implementation. We discuss the implementation strategy and experimental results. In addition to NMPC based swing-up control, we also present results from a gradient based iterative learning control, which is the basis our NMPC algorithm.

1 Introduction

Model predictive control (MPC) is a feedback control scheme that generates the control action based on a finite horizon open loop optimal control from the initial state. In addition to its intuitively appeal – choosing action based on its impact in the future rather than just reacting to the present – MPC also offers the possibility of incorporating control and state constraints, which few feedback control methods can claim to do. MPC was first proposed for linear systems [1, 2] and later extended to nonlinear systems (called NMPC) [3–8]. It has been especially popular in process control where slow system response permits the on-line optimal control computation. However, due to the computation load, application to systems with fast time constant is still elusive. This is especially true for open loop unstable systems. Instead of solving the complete optimal control problem in each sampling period, we have proposed

*School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, sjung@sununo.me.gatech.edu
†Electrical, Computer, & Systems Eng., Rensselaer Polytechnic Institute, Troy, NY 12180, wen@cat.rpi.edu
an NMPC scheme [9] that only seeks to reduce the predicted state error. The reduced computation load points to the potential applicability to systems with fast dynamics. Simulation results thus far have been promising. In this paper, we present the first experimental results of this NMPC scheme, applied to the swing-up of a rotary inverted pendulum.

Before implementing our NMPC scheme, we first test on the experiment an iterative learning control that is based on the same gradient iteration approach [10]. An fixed horizon open loop control is refined in each run by using the measured error from the physical experiment in the previous run and the gradient matrix from an analytical model. The swing-up and balancing (using a linear controller) is consistently achieved after only a few iterations.

For the NMPC implementation, the analytical model is used to refine the control sequence in each iteration. From the open loop control experiment, a significant mismatch between the model and experiment is observed. However, by tuning the design parameters, including the sampling rate, prediction horizon, control penalty function, and update parameter, we are able to achieve consistent swing-up experimentally. For the real-time implementation, we have used MATLAB xPC Target as the real-time computation platform. For the rotary inverted pendulum (which has 4 states) we are able to achieve 5ms sampling time for an 80-step look-ahead horizon.

The rest of the paper is organized as follows: The iterative learning control is discussed in Section 2. The NMPC control is presented in Section 3. The real-time computation platform is discussed in Section 4. The experimental results are shown in Section 5.

2 Iterative Learning Swing-up Control

An open loop control strategy was proposed in [10] for the path planning of nonholonomic systems. This strategy has served as the basis for the later extension to NMPC implementation [9]. Since this strategy is gradient based, the complete model information is needed. When this open loop control is applied experimentally, large error results due to the modeling error. In this section, we describes an extension to the iterative learning framework by using the end point error obtained experimentally to update the open control using the assumed model.
2.1 Open Loop Control

We first briefly review the gradient based control law used in [10]. Consider a discrete nonlinear system (obtained through, for example, finite difference approximation of a continuous time system)

\[ x_{k+1} = f(x_k) + g(x_k)u_k \]  

where \( x_k \in \mathbb{R}^n \) and \( u_k \in \mathbb{R}^m \). The goal is to find \( u = [u_0^T \ u_1^T \ \cdots \ u_{M-1}^T]^T \) to drive \( x_k \) from a given initial state \( x_0 \) to a given final state \( x_d \). We use \( \phi_M(x_0, u) \) to denote the state \( x_M \) obtained by using the control \( u \), starting from the initial state \( x_0 \).

We will approximate the control trajectory \( u \) by using the basis functions \( \{\psi_{ni} : i = 1, \ldots, N, n = 1, \ldots, N\} \):

\[ u_k = \sum_{n=1}^{N} \lambda_n \psi_{nk} = \Psi_k \lambda \]  

where \( \Psi_k = \begin{bmatrix} \psi_{1k} & \cdots & \psi_{Nk} \end{bmatrix} \) and \( \lambda = \begin{bmatrix} \lambda_1 & \cdots & \lambda_N \end{bmatrix}^T \). For the entire control trajectory, we have

\[ u = \Psi \lambda \]  

where \( \Psi = \begin{bmatrix} \Psi_0^T & \cdots & \Psi_{M-1}^T \end{bmatrix}^T \).

There are many possible choices of the basis functions. The standard pulse basis is not used due to the high sampling rate requirement. We have tried Fourier and Laguerre functions with the latter giving the better performance. This is due in part to the fact that Laguerre functions decay exponentially toward the end of the horizon allowing faster control action in the beginning of the horizon and avoiding control peaking at the end of the horizon.

Continuous time Laguerre functions, which form a complete orthonormal set in \( L_2(0, \infty) \), have been used in system identification [11] because of their convenient network realizations and exponentially decay profiles. The discrete time version of Laguerre functions has been used in identification for discrete time systems in [12]. Laguerre functions have also been proposed in the model predictive context [13, 14]. The details of the Laguerre functions used in our implementation is described in Appendix A.
Define the final state error as
\[ e(x_0, u) = \phi_M(x_0, u) - x_d. \] (4)

The coefficients \( \lambda_n \) in (2) can be updated using a standard gradient type of algorithm to drive \( e \) to zero. For example, the following is the Newton-Raphson update
\[ \lambda^{(i+1)} = \lambda^{(i)} - \eta_i \left( \frac{\partial \phi_M(x_0, \Psi \lambda^{(i)})}{\partial u} \right)^+ e(x_0, \Psi \lambda^{(i)}). \] (5)

The gradient \( \frac{\partial \phi_M(x_0, \Psi \lambda)}{\partial u} \) can be obtained from the time varying linearized system of (1) about the trajectory generated by \( u = \Psi \lambda \). Define
\[ L(\lambda) = \frac{\partial \phi_M(x_0, \Psi \lambda)}{\partial u} \Psi. \]

Eq. (5) is implemented as
\[ \lambda^{(i+1)} = \lambda^{(i)} - \eta_i v, \quad L(\lambda^{(i)}) v = e(x_0, \Psi \lambda^{(i)}) \] (6)
where \( v \) is solved using LU decomposition. The update parameter \( \eta_i \) is chosen based on the Amijo’s rule [15] (the step size continues to be halved until either the prediction error, \( e \), decreases or the minimum step size is reached). The end point error \( e \) converges to zero if \( L \) is always of full row rank. The issue of singularity (configurations at which \( L \) loses rank) has been addressed for continuous \( u \) [16, 17] and will not be addressed here.

The actuator \( u \) is typically bounded: \(|u| \leq u_{\text{max}}\) (shown here as a single input for simplicity). It can be incorporated in the update law through an exterior penalty function, e.g.,
\[ h(u) = \begin{cases} 0 & |u| \leq u_{\text{max}} \\ \gamma(u - u_{\text{max}})^2 & u > u_{\text{max}} \\ \gamma(u + u_{\text{max}})^2 & u < -u_{\text{max}} \end{cases} \] (7)

The constraint is imposed at each time instant, so the overall penalty function is
\[ z(\lambda) = \sum_{i=0}^{M-1} h(\Psi_i \lambda). \] (8)
The update law for $\lambda$ now needs to be modified to drive $(e, z)$ to zero:

$$
\lambda^{(i+1)} = \lambda^{(i)} - \eta_i v, \quad \begin{bmatrix} L \\ G \end{bmatrix} v = \begin{bmatrix} e \\ z \end{bmatrix}
$$  \tag{9}

where $G$ is the gradient matrix of $z$ with respect to $\lambda$:

$$
G = \frac{\partial z}{\partial \lambda} = \sum_{i=0}^{M-1} \left. \frac{\partial h(u)}{\partial u} \right|_{u=\Psi_i} \Psi_i.
$$  \tag{10}

### 2.2 Iterative Control Refinement

Due to the inevitable mismatch between the model and physical experiment, open loop control will result in possibly large end point error in the physical system. To address this issue, we have applied an iterative learning strategy which updates the control in (9) by using the error $(e, z)$ measured from the physical experiment but the gradient $L$ from the model. If the gradient mismatch between the physical system and the model is sufficiently small, then the updated control will result in a smaller end point error. The learning control scheme that we have used can be summarized as follows:

1. Find the open loop swing-up control based on simulation using the analytical model (described in Appendix B).
   
   The parameters that need to be chosen are control horizon $T$, sampling time $t_s$ (this affects the gradient operator approximation), number of Laguerre functions $N$, and the decaying factor in Laguerre functions $a$.

2. Apply the open loop control trajectory to the physical system and save the resulting state trajectory.

3. Update the control by using the actual final state error. The gradient matrix is computed by linearizing the analytical model along the measured state trajectory and the applied control trajectory.

4. Iterate steps 2 and 3 until the end point error $e$ is sufficiently small.

For the practical implementation of this learning algorithm, there are a number of parameters that need to chosen; the key ones are:

1. **Number of Basis Function, $N$.** Large $N$ means better approximation of $u$ but it also increases the computation load (this becomes more important for NMPC).
2. **Length of Horizon, M.** M needs to be sufficiently large so that the end point can be reached with the specified bound on \( u \).

3. **Decaying Factor in Laguerre Functions, \( a \).** This parameter (between 0 and 1) determines the decay rate of \( \psi_n \). Smaller the \( a \), faster is the convergence of \( u \) to zero. However, if \( a \) is chosen too small, the control constraint may be difficult to satisfy.

The experimental results are given in Section 5.1.

### 3 Nonlinear Model Predictive Control (NMPC)

This section presents the NMPC algorithm that we have implemented for the pendulum swing-up experiment. The algorithm is based on the result in [9]. The basic idea is simple: the open loop control law iteration is executed with the current state as the initial state and the current control trajectory as the initial guess of the control. Then a fixed number of Newton-steps is taken and the resulting control is applied. The process then repeats at the next sampling time.

#### 3.1 Description of the NMPC Algorithm

For the NMPC implementation, due to the moving horizon implementation, we use the pulse basis for discretization. To describe the algorithm analytically, again consider the discrete nonlinear system (1). Let the prediction horizon be \( M \). Denote the predictive control vector at time \( k \) by \( u_{k,M} \):

\[
\mathbf{u}_{k,M} = [u_1^{(k)}, \ldots, u_M^{(k)}], \quad \mathbf{u}_{k,M} \in \mathbb{R}^{m \cdot M}.
\]  

(11)

Let \( \phi_M(x_k, u_{k,M}) \) be the state at the end of the prediction horizon, starting from \( x_k \) and using the control vector \( u_{k,M} \). The predicted state error is then

\[
\mathbf{e}_{M,k} = \phi_M(x_k, u_{k,M}) - x_d.
\]  

(12)

The main idea of the algorithm is to simultaneously perform the open loop iteration over the prediction horizon and apply the updated control to the system at the same time. The implementation of the algorithm can be summarized as
follows:

1. At the initial time with the given initial state $x_0$, choose the initial guess of the predictive control vector $u_{0,M}$.

   Also compute the equilibrium control, $u_d$ from

   $$f(x_d) + g(x_d)u_d = x_d$$

   (13)

2. For $k \geq 0$,

   (a) Calculate one Newton-step control update:

   $$v_{k,M} = u_{k,M} - \eta_k (\nabla u_{k,M} \phi_M(x_k, u_{k,M}))^\top e_{M,k}.$$  

   (14)

   The gain, $\eta_k$, is found based on Amijo’s rule to ensure predicted error is strictly decreasing. This can be done as long as the gradient matrix is non-singular.

   (b) Shift the predictive control vector by 1 step (since the first element will be used for the actual control) and add the equilibrium control at the end of the vector:

   $$u_{k+1,M} = \Gamma v_{k,M} + \Phi u_d$$

   (15)

   where $\Gamma \in \mathbb{R}^{mM \times mM}$ and $\Phi \in \mathbb{R}^{mM \times m}$ are defined as:

   $$\Gamma = \begin{bmatrix} 0_{m(M-1) \times m} & I_m(M-1) \\ 0_{m \times m} & 0_{m \times m(M-1)} \end{bmatrix}, \quad \Phi = \begin{bmatrix} 0_{m(M-1) \times m} \\ I_m \end{bmatrix}$$

   (c) Compute the control, $u(k)$ to be applied as

   $$u(k) = \Lambda v_{k,M}$$

   (16)

   where $\Lambda = [I_m \ 0_{m \times (M-1)m}]$.

   (d) Repeat Step 2a–2c at the next time instant.
Note that for stability, the non-singularity condition of the gradient matrix is needed as in the open-loop case. For a more detailed discussion of the stability condition, see [18].

Remarks:

1. The state and control constraint can be incorporated through exterior penalty functions as in (8).

2. The parameters that affect the performance of this algorithm are:
   - Prediction horizon $M$: This is determined from $T/t_s$ where $T$ is the horizon in time and $t_s$ is the sampling rate. Large $T$ is beneficial in keeping $u$ within the constraint but implies heavier computation load in real-time. Small $t_s$ is important to keep the approximation error (due to discretization) small, but it also leads to large $M$ and heavier real-time computation load.
   - Initial predictive control vector, $u_{0,M}$: Without any a priori insight, this can just be chosen as a zero vector. If some off-line optimization has already been performed, it can be used as the initial guess.

4 Real-Time Implementation

The learning control and NMPC algorithms have been applied to a rotary inverted pendulum testbed. The system is nonlinear, underactuated, and unstable at the target state (vertical up position), making the system a challenging candidate for control. The horizontal link is controlled to bring the vertical link upright. This is called the swing-up control. The system model is included in Appendix B. In this section, we describe the physical hardware and software environment that we have used to implement the learning and NMPC algorithms.

4.1 Hardware

The experiment was constructed as part of a kit in support of control education [19]. A MicroMo motor is used in the driving joint. A 1024-count encoder is mounted on each joint. The open-loop control law is implemented on a DSP control board made by ARCS, Inc. This board includes a TI C30 class processor and on-board A/D, D/A, encoder, and digital I/O interfaces. The feedback NMPC control is implemented using MATLAB xPC Target with Real-Time Workshop and Simulink Toolbox from MathWorks, Inc. The incremental encoder board is APCI-1710 from ADD-DATA, Inc., supporting 4 channels of single-ended/differential encoder with 32-bit resolution. The D/A
board is PCIM-DAS1602/16 from Measurement Computing, Corp., which supports 2 16-bit D/A channels. The real-time controlling PC is an AMD Athlon 1.4GHz with 512MB RAM.

4.2 Software

4.2.1 Simulation Model

Both learning control and NMPC have been extensively tested on the simulation model before applying to the physical experiment. Simulink is used for the simulation study as well as for the real-time code generation by using the Real-Time Workshop. In the Simulink model, the NMPC algorithm was programmed in C and converted into a Simulink S-Function block by using the C-MEX S-Function Wrapper. This simulation environment allows us to tune the controller parameters.

4.2.2 Real-Time Model

For the iterative learning control, an open-loop controller is coded using Simulink and compiled into C-code by using the Real-Time Workshop (RTW). The state variables are logged for each run for the off-line control update. The NMPC algorithm is also coded using Simulink and compiled using RTW.

4.2.3 Velocity Estimation

For the physical experiment, only the position variable is available (from the incremental encoders), and velocities have to be estimated from the position measurement. For the open-loop control, velocities are estimated by simple Euler approximation. For the NMPC control system, the washout filter is used. When the pendulum is near the vertical equilibrium, a full order linear observer is used. A more detailed study of the effect of velocity estimation on the performance of NMPC can be found in [20].
5 Experimental Results

5.1 Iterative Learning Control

Since for the learning control, the input is open-loop, real-time computation load is not an issue. Hence we choose a longer prediction horizon:

\[ T = 1.5 \text{s}, \quad t_s = 5\text{ms}, \quad M = T/t_s = 300. \]

The number of Laguerre functions used is \( N = 10 \) and the Laguerre decaying factor, \( a \), is chosen to be 0.9. For the control penalty function, we choose

\[ \gamma = 0.45, \quad u_{\max} = 0.45\text{N-m}, \quad u_{\min} = -0.45\text{N-m} \]

We have observed that these parameters may vary significantly and still achieve swing-up and capture. The key consideration is that \( t_s \) should be chosen small enough to ensure sufficiently small discretization error and \( T \) should be chosen large enough to allow swing-up with the specified control bound. Though the analytical model is not good enough for direct open loop control, it appears to be adequate for generating a descent direction.

For the physical experiment, the initial control sequence is obtained using the algorithm described in Section 2.1. However, the model/plant mismatch prevents the pendulum to be captured by the linear controller. After three iterations using the gradient algorithm in Section 2.2, the linear controller is able to capture and balance the pendulum. Figure 1 shows the experimental results using the control sequence obtained after the third iteration. The pendulum link is shown to be swung up (i.e., \( x_2 \) goes from \(-90^\circ\) to \(90^\circ\)) after two swings, and then captured and balanced by the linear controller. Note that the control effort is well within the saturation limit.

5.2 NMPC

After tuning the controller parameters using simulation, we have settled with the choice of

\[ M = 80, \quad T = 0.4\text{s}, \quad t_s = 5\text{ms}. \]
Again, these parameters may be adjusted and still achieve consistent swing-up. The parameter guide line is similar to the iterative learning control case, $t_s$ should be small enough to limit the discretization error, $T$ should be large enough for the control bound to be satisfied. We have observed consistent swing-up and capture for $T$ varying between 0.3 sec. and 0.45 sec., and $t_s$ varying between 4 ms and 6 ms. The number of swings (and hence the time to reach the swing-up position) does vary with the choice of these parameters. The initial control vector is set to zero to minimize the use of a priori information. The control constraint is imposed via a quadratic exterior penalty function described in (7), with the parameters

$$\gamma = 0.45, \quad u_{max} = 0.1N\cdot m, \quad u_{min} = -0.1N\cdot m.$$ 

Note that the exterior penalty function does not ensure a hard bound on the input; therefore, a tighter control constraint is chosen to satisfy the physical constraint of 0.45N-m. In both simulation and physical experiments, a hard saturation constraint of ±0.45N-m is imposed for the control input.

The simulation result using the analytical model is shown in Figure 2. Note that the pendulum link initially moves in the opposite direction as the iterative control result in Figure 1, since the initial control sequence is chosen to be zero and it takes time for the control sequence to evolve through gradient modification in (15). After about 0.3 sec., the control and state trajectories become more similar to the open loop result in Figure 1. Four sample experimental runs using the same controller parameters are shown in Figures 3-6. Even though the controller parameters are the same, the response behaves differently after the pendulum is swung up. This is due to the lack of robustness in the balanced configuration. In Figure 3-4, the pendulum remains balanced after swing-up, but in Figure 5-6, the pendulum falls down at about 3 sec., in different directions, and then swings up again within 1 sec.

In all these experiments, the state trajectories in the swing-up phase match reasonably well with the simulation prediction shown in Figure 7. The control trajectories show the same trend within the first second, but the experimental responses contain much more chattering between the saturation levels when it is near the swung-up position (see Figure 8). In general, the NMPC controller is very sensitive to disturbance during the balancing phase. It appears that the 80-step horizon is too long at the balanced position for the controller to react effectively to the destabilizing effect of the gravity. We have recently tried to adjust the length of the prediction horizon near the balancing position and had some success [20]. This inability to maintain the balancing position may be attributed to several factors: modeling error due to the vibration of the fixed support, the relatively poor velocity estimation (estimated through washout filter...
in this implementation), and poor estimate of the Coulomb friction. When a linear observer based controller is used near the balancing state, the experimental results shown in Figure 9–10 now very close to the simulation.

6 Conclusions

This paper presents the experimental implementation and results of iterative learning control and nonlinear model predictive control applied to the swing-up of a rotary inverted pendulum with input torque constraint. In both cases, computation load is reduced by using a Newton-step control update instead of solving the complete optimal control problem. In the real-time control case with NMPC, MATLAB xPC-Target is used as the real-time computation platform, and 5ms sampling rate is achieved with an 80-step look-ahead horizon. Coupled with a linear control law that captures and balances the pendulum in the neighborhood of the vertical equilibrium, both iterative learning control and NMPC control can swing up and balance the pendulum consistently. Current work involves on improving the robustness of NMPC and applying NMPC to the observer implementation.

Acknowledgment

This work is supported in part by the National Science Foundations under grant CMS-9813099, and in part by the Center for Advanced Technology in Automation Technologies under a block grant from the New York State Science and Technology Foundation.
References


A  Laguerre Functions used in Learning Control

In continuous time, the Laguerre function set is defined as:

\[ f_i(t) = \sqrt{2^p} \frac{e^{pt}}{(i-1)!} \frac{d^{i-1}}{dt^{i-1}}[t^{i-1}e^{-2pt}] \]  \hspace{1cm} (17)

where \( i \) is the order of the function set \( i = 1, 2, \ldots, N \) and \( p \) is Laguerre pole. The corresponding Laplace transform for this set is:

\[ F_i(s) = \sqrt{2^p} \frac{(s-p)^{i-1}}{(s+p)^i}. \] \hspace{1cm} (18)

The \( z \)-transform of the discrete Laguerre function set, \( l_i(k), \ i = 1, 2, \ldots, N \) is given in [12]:

\[ L_i(z) = \sqrt{1-a^2} \left[ \frac{az-1}{z-a} \right]^{i-1}. \] \hspace{1cm} (19)

Based on the \( z \)-transform, it is possible to the discrete Laguerre function set satisfies the difference equation

\[ L_{k+1} = \Omega L_k \] \hspace{1cm} (20)

where

\[ L_k = \begin{bmatrix} \psi_{1k} & \psi_{2k} & \cdots & \psi_{N_k} \end{bmatrix}^T \]

\[ \Omega = \begin{bmatrix} a & 0 & \cdots & 0 \\ a^2 - 1 & a & \cdots & 0 \\ a(a^2 - 1) & a^2 - 1 & \cdots & 0 \\ a^2(a^2 - 1) & a(a^2 - 1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a^{N-2}(a^2 - 1) & a^{N-3}(a^2 - 1) & \cdots & a \end{bmatrix} \]

with initial condition

\[ L_0 = \sqrt{1-a^2} \begin{bmatrix} 1 & a & a^2 & \cdots & a^{N-1} \end{bmatrix}^T \]
and discrete Laguerre pole $a$.

B Model of Rotary Inverted Pendulum

The rotary inverted pendulum used in this study is shown in Figure 11. The equation of motion is of the following form:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + F(\dot{\theta}) + G(\theta) = Bu$$  \hspace{1cm} (21)

where $\theta$, $\dot{\theta}$, $\ddot{\theta}$ are the joint angle, velocity and acceleration vectors, $M(\theta)$ is the mass-inertia matrix, $C(\theta, \dot{\theta})\dot{\theta}$ is the centrifugal and Coriolis torques, $F(\dot{\theta})$ is the friction, $G(\theta) \in \mathbb{R}^2$ is the gravity load, and $u$ is the applied torque.

The inertia matrix is given by

$$M(\theta) = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$  \hspace{1cm} (22)

with

$$m_{11} = m_1 l_1^2 + I_{331} + m_2 \left( I_1^2 + l_2^2 \cos^2 (\theta_2) \right) + I_{222} \sin^2 (\theta_2) + I_{332} \cos^2 (\theta_2) + 2 \sin (\theta_2) \cos (\theta_2) I_{232}$$

$$m_{12} = m_{21} = -m_2 l_1 l_2 \sin (\theta_2) + I_{122} \sin (\theta_2) + I_{132} \cos (\theta_2)$$

$$m_{22} = m_2 l_2^2 + I_{112}$$

The Coriolis matrix is given by

$$C(\theta, \dot{\theta}) = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$  \hspace{1cm} (23)

with

$$c_{11} = -2m_2 l_2^2 \sin (\theta_2) \cos (\theta_2) \dot{\theta}_2 + 2 \sin (\theta_2) \cos (\theta_2) \left( I_{222} - I_{332} \right) \dot{\theta}_2 + I_{232} \left( 4 \cos^2 (\theta_2) - 2 \right) \dot{\theta}_2$$

$$c_{12} = -m_2 l_1 l_2 \cos (\theta_2) \dot{\theta}_2 + I_{122} \cos (\theta_2) \dot{\theta}_2 - I_{132} \sin (\theta_2) \dot{\theta}_2$$

$$c_{21} = m_2 l_2^2 \sin (\theta_2) \cos (\theta_2) \dot{\theta}_1 - I_{222} \sin (\theta_2) \cos (\theta_2) \dot{\theta}_1 + I_{332} \sin (\theta_2) \cos (\theta_2) \dot{\theta}_1 + I_{232} \left( 1 - 2 \cos^2 (\theta_2) \right) \dot{\theta}_1$$

$$c_{22} = 0$$
The friction is modeled as Coloumb + viscous:

$$F(\dot{\theta}) = \begin{bmatrix} F_{v1}\dot{\theta}_1 + F_{c1}sgn(\dot{\theta}_1) \\ F_{v2}\dot{\theta}_2 + F_{c2}sgn(\dot{\theta}_2) \end{bmatrix}$$  \quad (24)$$

where $F_v$ represent the coefficient of viscous friction for each joint, and $F_c$ of Coulomb friction. For the NMPC computation, we ignore the Coulomb friction terms due to the difficulty that discontinuity poses in gradient calculation.

The gravity term is

$$G(\theta) = \begin{bmatrix} 0 \\ m_2g l_{c2} \cos(\theta_2) \end{bmatrix}$$ \quad (25)$$

The torque is applied only on the first link, therefore

$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$ \quad (26)$$

For the state space representation, we define the state vector as

$$x = \begin{bmatrix} \theta \\ p \end{bmatrix}$$

where $p \overset{\Delta}{=} M(\theta)\dot{\theta}$ is the generalized momentum. The equation of motion in the state space form is then

$$\dot{x} = \begin{bmatrix} M^{-1}(\theta)p \\ (\dot{M}(\theta,M^{-1}p) - C(\theta,M^{-1}p))M^{-1}p - F(M^{-1}p) - G(\theta) \end{bmatrix} + \begin{bmatrix} 0_{2\times1} \\ B \end{bmatrix} u$$ \quad (27)$$

The physical parameters are determined through direct measurements:

$m_1 = 0.11409$ kg  \quad $l_1 = 0.169$ m  \quad $l_{c1} = 0.1137$ m  

$m_2 = 0.04474$ kg  \quad $l_2 = 0.162$ m  \quad $l_{c2} = 0.0511$ m

where subscripts 1 and 2 denote link 1 and link 2, respectively.
The moment of inertia for each link can be computed using the above values:

\[ I_{11_1} = 2.62e^{-5} \text{ kg m}^2 \quad I_{22_1} = 3.59e^{-5} \text{ kg m}^2 \quad I_{33_1} = 3.43e^{-5} \text{ kg m}^2 \]

\[ I_{12_1} = I_{21_1} = 0 \quad I_{13_1} = I_{31_1} = 1.29e^{-5} \text{ kg m}^2 \quad I_{23_1} = I_{32_1} = 0 \]

\[ I_{11_2} = 1.87e^{-5} \text{ kg m}^2 \quad I_{22_2} = 4.82e^{-6} \text{ kg m}^2 \quad I_{33_2} = 1.86e^{-5} \text{ kg m}^2 \]

\[ I_{12_2} = I_{21_2} = 0 \quad I_{13_2} = I_{31_2} = 0 \quad I_{23_2} = I_{32_2} = 5.24e^{-6} \text{ kg m}^2 \]

Identification of friction parameters of the inverted pendulum system was performed and are given as follows

\[ F_{v_1} = 0.11409 \text{ N-m/rad/sec} \quad F_{c_1} = 0.008 \text{ N-m} \]

\[ F_{v_2} = 0.0001 \text{ N-m/rad/sec} \quad F_{c_2} = 0.0 \text{ N-m} \]
Figure 1: State and Control trajectories on the Open-Loop Control System: After Third Iteration
Figure 2: Simulation Response with NMPC

Figure 3: Experimental Response with NMPC: sample run #1
Figure 4: Experimental Response with NMPC: sample run #2

Figure 5: Experimental Response with NMPC: sample run #3
Figure 6: Experimental Response with NMPC: sample run #4

Figure 7: Comparison of State Response
Figure 8: Comparison of Control Trajectory

Figure 9: Experimental Response with NMPC + linear balancing control: sample run #1
Figure 10: Experimental Response with NMPC + linear balancing control: sample run #2

Figure 11: The Rotary Inverted Pendulum Experiment