

ROLE OF AUTOMATED SYMBOLIC GENERATION OF EQUATIONS OF MOTION TO ENHANCE ROBOTICS EDUCATION

Hrishi Shah, Sumit Tripathi, Leng-Feng Lee, and Venkat Krovi
Department of Mechanical and Aerospace Engineering
State University of New York at Buffalo

Abstract

In recent years there has been a significant increase in the variety and complexity of Articulated-Multi-Body-Systems (AMBS) used for various applications. There is also increased interest in the model-based design-refinement and controller-development, which is critically dependent upon availability of underlying plant-models.

Kinematic and dynamic plant-models for AMBS can be formulated by systematic application of physics postulates. This process, in its various variants, forms the basis of various mechanisms/robotics courses. However, the type and complexity of the example systems is often limited by the tractability of first generating and subsequently analyzing system equations-of-motion (EOM). Nonetheless, it should be noted that using simpler examples alone may sometimes fail to capture important physical phenomena (e.g. gyroscopic, coriolis). Hence, we examine the use of some contemporary symbolic- and numeric-computation tools to assist with the automated symbolic equation generation, sensitivity analysis and development of model based controllers to enhance various courses.

From the design and analysis perspective, we examine a host of examples starting with basic examples like the single pendulum, double pendulum; building up to intermediate examples like the four-bar mechanism and culminating with the implementation of 3-PRR and 3-RRR planar parallel platform mechanisms to showcase the modeling and analysis aspects. From a control perspective, we focus on the Furuta pendulum example. This allows us to showcase the emergence of model-complexity, even in relatively-simple two-jointed mechanical system and yet to study various

aspects of model-creation, model-linearization and model-based controller development.

The principal underlying philosophy of our effort is to establish linkage between traditional modeling approaches and use of these contemporary tools. We also try to make a case for use of automatic symbolic computation and manipulation as a means for enhancing understanding of both basic and advanced AMBS concepts. Lastly, we document our efforts towards creation of self-paced tutorials and case-studies that serve to showcase the benefits.

Introduction

Over the past few decades, several seminal textbooks [1-4] have addressed the mathematical modeling and analysis of kinematics, dynamics and control of articulated multibody systems (AMBS). In their simplest form, the governing equations-of-motion (EOM) take the form of a system of Ordinary Differential Equations (ODEs). However, there are many factors that can quickly introduce complexity in these governing equations. First, and perhaps the greatest, source of complexity comes from the effects of finite rotations in two- and three-dimensions, which introduces trigonometric complexity (in the form of sine and cosine terms). This modeling complexity is amplified in the transition from linear (1D) systems to planar (2D) systems and ultimately to spatial (3D) systems. Second, most real-life multibody systems possess one or more closed kinematic loops, typically to enhance their stiffness and payload capacity. Such closed-kinematic loops can help reduce overall actuation requirements by creating constraints within system degrees-of-freedom. However, these algebraic constraints interact with the underlying systems of ODEs of the

unconstrained systems to create systems of Differential Algebraic Equations (DAEs). Hence, both initial modeling as well as subsequent performance analysis and controller development tend to be difficult in such multibody systems.

Nevertheless, effectively modeling and analyzing the kinematics and dynamics of all such systems within a computational framework is critical for the apriori prediction of the overall system response. From a design perspective, accurate and computationally-efficient simulation models are vital for rapid design-refinement through iterative simulation-based parametric studies. From the control perspective, the same model can also help implement more effective model-based nonlinear control strategies. The recent trend towards larger multi-DOF articulated mechanical systems, operating in 3D task spaces typically at higher operating speeds has created more stringent performance requirements of the developed controllers. This, in turn, paves the way for development and deployment of model-based non-linear controllers to satisfy performance criteria.

Thus, the fundamental challenge in such systems remains: “Given a description of a mechanical system in terms of the relative physical layout, interconnections, and mechanical properties, how we can formulate the kinematic or dynamic equations of motion (EOM), characterize the system response and exercise control over its environmental interaction?”

The variety of formulations that exist for multibody systems can be daunting. Such variety arises from the interplay between (i) the multitude of problem tasks that can be addressed, (ii) the varying levels of analysis, and (iii) the numerous possible system configuration descriptions. The designer may seek to address forward or inverse problems for such systems, operating in the kinematic or dynamic regimes, with system configurations modeled in terms of a variety of coordinates (absolute, relative, mixed). Oftentimes, selection of specific coordinate descriptions for systems

offers unique advantages and disadvantages for specialized problem tasks/analyses. For example, in systems with joint-based actuation, a relative joint-coordinate-based formulation simplifies the determination of the (external) actuation forces for inverse-dynamics problems. However, additionally determining the internal pin-reaction forces is easier in some form of extended coordinate system (for example, absolute Cartesian coordinates of each link) with suitable constraints within an augmented Lagrange formulation.

Traditionally, the ability to select and switch the formulation, depending on the task at hand has created challenges – oftentimes requiring a reformulation of the EOM from scratch. It is also worth noting that practical limits on system-size are often encountered when using certain existing formulations (such as the Lagrangian formulation) to derive EOM of increasingly-complex systems. For all but the simplest problems, however, this task can be laborious and error-prone. The complexity encountered in real-life multibody systems can very easily take many man-months of effort to develop and validate by hand.

In our work, we explore building upon the capabilities of MapleSim [5], a Maple toolbox to facilitate the rapid and automated creation of the symbolic EOMs of large-scale articulated multibody models. We exploit MapleSim ability to create kinematic and dynamic EOMs, using user’s preferred formulation and coordinates within a systematic and automated symbolic implementation. Eventually, we also showcase the connection to other software like Simulink and its real time controller implementation.

However, there remains a question of applicability and accuracy of this new found capability which we will systematically examine. We will study this in the context of several case studies, beginning with simple pendulum, double pendulum; building up to intermediate examples like the four-bar mechanisms, and finally examine the implementation of 3-PRR, 3-RRR planar parallel platform mechanisms. These case studies of articulated systems with active/

passive joints and multiple closed kinematic loops engender many of the complicating factors entering the equations of motion (EOMs).

The rest of the paper is organized as follows. We discuss the background as well as challenges in conventional approach and how contemporary tools can help alleviate some of these challenges. Then, we layout the staged implementation of our tutorials, followed by several case studies used in the tutorials. In the next section, we discuss the Furuta pendulum which serves as an advanced model including effects of under-actuation. We discuss controller development and subsequent validation by simulation and hardware-in-the-loop (HIL) testing. Finally, we conclude with a discussion of critical issues faced during implementation of these plans.

Background & Challenges

Traditionally, many of the concepts and ideas for AMBS (including the study of kinematics and dynamics), are delivered in a classroom-based lecture. In this setting, mathematical formulations of the mechanism are usually emphasized and students are required to formulate the equations governing the kinematics and dynamics of some simple mechanisms and then solve these using algebraic techniques. The main advantage of this approach is that it permits the student to understand the fundamental theory underlying the analysis as well as get a handle on the formulation that forms the basis for the analysis of more complicated mechanisms. Thus, with a grasp of the basic concepts and formulations, students can implement the techniques algorithmically by suitably programming [6,7]. However, the complexity of the analyzed mechanism imposes limitations for the analytical method. For example, the formulation of a set of equations for a simple four bar mechanism is manageable for links with center of mass along the link lengths. However, if the shape of the linkage is complex or the number of links increases, the formulation becomes more complicated and time consuming. Thus, the analytical method is most often limited to

simple two-dimensional mechanisms and links with relatively simple geometries. Similarly, control courses, limited by the student's skill and time within the curriculum, rely mostly on popular, though simple, systems for which model equations are readily available.

Many of these problems can be alleviated by using computer aided analysis software tools (e.g. ADAMS, VisualNastran, etc.) which are available to support the simulation based study of AMBS. The main benefits of this method are that the students can analyze more complex mechanisms with detailed link geometries, obtain quick results and compare many possibilities prior to selecting best mechanism by permitting the detailed visualization of virtual mechanisms, giving the student a better understanding of the motion of the mechanism, the path of a specific point, and the functionality of the mechanism. The principal disadvantage is that the formulations of the kinematic and dynamic analysis of the mechanism are completely hidden from the student. The black box approach to the underlying governing equations can in many cases hinder understanding of the concepts behind many of the mechanisms. Furthermore, implementation of advanced model-based controllers requires the plant model either in the form of equations or a sensor based virtual implementation that could provide all the data required for model identification and design. However, the latter approach introduces an additional model identification step and also takes more time for sensitivity analyses, since the model needs to be re-run every time. Hence, there is a great need for symbolic computation of the plant equations.

It is to overcome the limitation of both these approaches that we examined the third automated symbolic modeling approach. Automated symbolic approaches derive their many advantages by eliminating the need for manual EOM computation and subsequent manipulation which was error prone and time consuming. Two trends that have favored the adoption and rapid proliferation of the symbolic computation are: (i) the availability of low-cost PC based symbolic manipulation tools like Maple; and (ii) the capability of integrating

multiple functionalities into a unified environment like MapleSim/Connector.

However, there are several pedagogic issues that hinder the direct deployment of technological tools like MapleSim:

A. Currently the use of MapleSim needs “expert users” who can not only model, but also analyze the results for their correctness. While tutorials are made available by the vendors of these tools, they are targeted at a more experienced user (typically with a graduate level knowledge of AMBS). These traditional tutorials may assume a certain level of both mathematical sophistication and engineering experience from the user.

B. Novitiate robotics students may tend to have difficulty understanding both technical (theoretical) concepts as well as their technological implementation. Moreover, it is crucial that student gleans a greater insight into the problem and is better equipped to make engineering judgments from the information obtained from the use of such software.

It is to promote this type of greater understanding that we are creating a series of self-paced MapleSim tutorials deployed in two contexts. First, we targeted the course MAE 413/513: “Robotic Mobility and Manipulation” at the State University of New York at Buffalo. Simultaneously, we also sought to examine the efficacy of these tools and to provide support to develop a self-paced MAE501: "Independent Study" course focusing on model-based controller development for Furuta pendulum.

MAE413/513: "Robotic Mobility and Manipulation" Tutorials Development [8].

Traditionally, many of the concepts and ideas behind articulated multibody systems (AMBS) (including the study of kinematics and dynamics), are delivered with simple examples in a didactic classroom setting. In MAE413/513 we begin with the formulation of dynamic EOMs of various simple mechanical systems (such as single pendulum) using the Newton Euler and Euler Lagrange methods. This is

followed by the extraction of dynamic equations in matrix form and structural properties. After obtaining the EOMs, students proceed to perform various forward and inverse dynamics simulations in MATLAB using numerical integration routines (ode5 for fixed time-step solver and ode45 for variable time step solver).

While addressing the basic formulations, many assumptions were made in order to simplify the calculations. However, it was unclear as to how situations involving more complex link geometries could be handled. For example, if the given links are not slender (shown in Figure 1) and/or if the center of mass is not at the geometric centroid of the link, then the problem cannot be directly handled using the methods taught in class.

Hence we sought to create a set of tutorials to supplement the class. The goal is to reinforce the ideas and concepts originally presented in the course by paralleling the course material with these tutorials (Figure 2). The lecture coverage of the course provided the student exposure to the use of traditional simplified analytical modeling and computational analysis methods. Independent exploration by the student with the tutorial was intended to promote interactive experiential learning. The desired outcomes included improving the overall understanding of system modeling by the students and accelerating their learning experience without increasing the lecture hours.

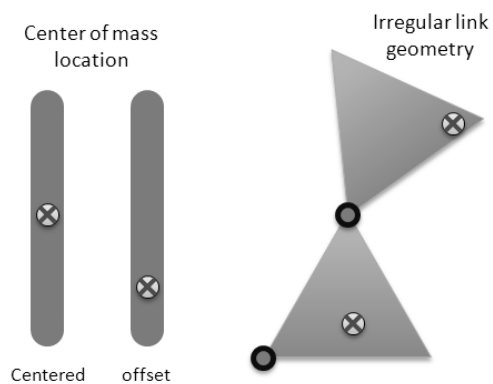


Figure 1: Traditional formulation of equations of motion often rely on simplified model and cannot be directly applied to irregularly shaped linkage parts.

Traditional Approach

Lagrangian of a system is defined as

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \quad \dots (1)$$

Where;

T = Kinetic Energy, V = Potential Energy

EOM:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) - \frac{\partial L}{\partial q} = \frac{\partial \Pi}{\partial \dot{q}} - \frac{\partial \Delta}{\partial q} \quad \dots (2)$$

Where;

$\Pi(q, \dot{q})$ = Power Supplied, $\Delta(q, \dot{q})$ = Dissipation

EOM of Double Pendulum:-

l_i = length of Link i , where $i=1, 2$

C_{G_i} = Locations of the center of gravity, $i=1, 2$

x_i = Distance from joint i to C_{G_i} , $i=1, 2, 3$

$a = x_1$, $b = x_2$, $c = x_3$ m_i = Mass of Each Link

I_i = M.I. of Link i about C_{G_i} , where $i=1, 2$

Displacement of C_{G_i} w.r.t. the fixed frame:

Link 1:

$$\vec{r}_{C_{G_1}} = a \sin \theta_1 \hat{i} - a \cos \theta_1 \hat{j} \quad \dots (3)$$

Link 2:

$$\vec{r}_{C_{G_2}} = (L_1 \sin \theta_1 + b \sin \theta_2) \hat{i} + (-L_1 \cos \theta_1 + b \cos \theta_2) \hat{j} \quad \dots (4)$$

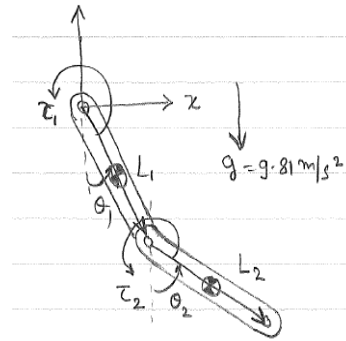
Differentiate Equation (3) and (4):

Link 1: $\dot{\vec{r}}_{C_{G_1}} = (a \cos \theta_1 \dot{\theta}_1) \hat{i} + (a \sin \theta_1 \dot{\theta}_1) \hat{j} \quad \dots (5)$

Link 2:

$$\dot{\vec{r}}_{C_{G_2}} = (L_1 \cos \theta_1 \dot{\theta}_1 + b \cos \theta_2 \dot{\theta}_2) \hat{i} + (L_1 \sin \theta_1 \dot{\theta}_1 + b \sin \theta_2 \dot{\theta}_2) \hat{j} \quad \dots (6)$$

Link



Power supplied to the system

$$\Pi(\dot{q}) = \tau_1 \dot{\theta}_1 + \tau_2 \dot{\theta}_2 \quad \dots (22)$$

And:-

$$\frac{\partial \Pi}{\partial \theta_1} = \tau_1, \quad \frac{\partial \Pi}{\partial \theta_2} = \tau_2 \quad \dots (23)$$

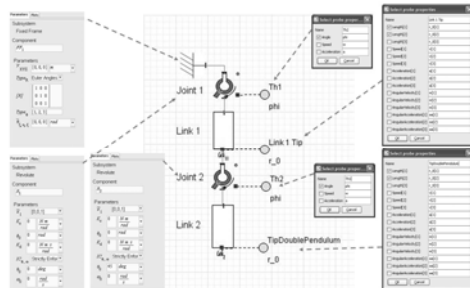
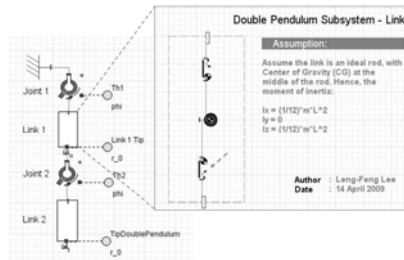
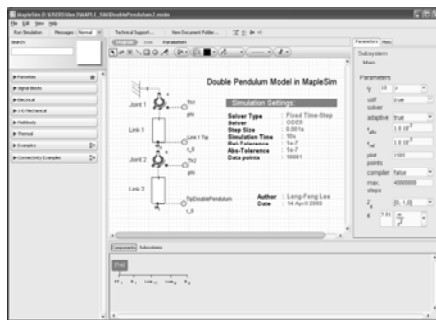
Dissipation function; $\Delta = 0 \Rightarrow \frac{\partial \Delta}{\partial \dot{q}} = 0$

Dynamics Equation of Double compound Pendulum:

$$[m_1 a^2 + m_1 L_1^2 + I_1] \ddot{\theta}_1 + m_2 L_1 b \cos(\theta_2 - \theta_1) \ddot{\theta}_2 - m_2 L_1 b \sin(\theta_2 - \theta_1) \dot{\theta}_1 \dot{\theta}_2 + m_2 L_1 b \sin(\theta_2 - \theta_1) (\dot{\theta}_1 - \dot{\theta}_2) \dot{\theta}_1 + m_2 g a \sin \theta_1 + m_2 L_1 \sin \theta_1 \dot{\theta}_1^2 = \tau_1 \quad \dots (25)$$

$$m_2 L_1 b \sin(\theta_2 - \theta_1) \dot{\theta}_1 \dot{\theta}_2 + [m_2 b^2 + I_2] \ddot{\theta}_2 + m_2 L_1 b \sin(\theta_2 - \theta_1) (\dot{\theta}_1 - \dot{\theta}_2) \dot{\theta}_1 + m_2 L_1 b \sin(\theta_2 - \theta_1) \dot{\theta}_1 \dot{\theta}_2 + m_2 g b \sin \theta_2 = \tau_2 \quad \dots (26)$$

Automated Symbolic Approach



$$\left[\left(\frac{1}{4} l_1^2 m_1 + l_1 m_2 l_2 \cos(\theta_2) + J_1 + J_2 + l_1^2 m_2 + \frac{1}{4} m_2 l_2^2 \right) \left(\frac{\partial^2}{\partial t^2} \theta_1 \right) + \left(\frac{1}{2} l_1 m_2 l_2 \cos(\theta_2) + J_2 + \frac{1}{4} m_2 l_2^2 \right) \left(\frac{\partial^2}{\partial t^2} \theta_2 \right) + \frac{1}{2} \cos(\theta_1) l_1 m_1 g + \cos(\theta_1) l_1 m_2 g - \frac{1}{2} l_1 m_2 l_2 \left(\frac{\partial}{\partial t} \theta_2 \right)^2 \sin(\theta_2) + \frac{1}{2} l_2 \cos(\theta_1) \cos(\theta_2) m_2 g - \frac{1}{2} l_2 \sin(\theta_1) \sin(\theta_2) m_2 g - l_1 m_2 l_2 \left(\frac{\partial}{\partial t} \theta_1 \right) \left(\frac{\partial}{\partial t} \theta_2 \right) \sin(\theta_2) - \tau_1 \right]$$

$$\left[\left(\frac{1}{2} l_1 m_2 l_2 \cos(\theta_2) + J_2 + \frac{1}{4} m_2 l_2^2 \right) \left(\frac{\partial^2}{\partial t^2} \theta_1 \right) + \left(\frac{1}{2} l_1 m_2 l_2 \cos(\theta_2) + J_2 + \frac{1}{4} m_2 l_2^2 \right) \left(\frac{\partial^2}{\partial t^2} \theta_2 \right) + \frac{1}{2} l_2 \cos(\theta_1) \cos(\theta_2) m_2 g - \frac{1}{2} l_2 \sin(\theta_1) \sin(\theta_2) m_2 g + \frac{1}{2} l_1 m_2 l_2 \left(\frac{\partial}{\partial t} \theta_1 \right)^2 \sin(\theta_2) - \tau_2 \right]$$

Figure 2: Linkage between traditional approach and automated symbolic approach to obtain EOM of a double pendulum.

Other considerations behind the selection and implementation included: (1) the accessibility of the software within the class; (2) the ease of learning (within a semester or less); and (3) the unified modeling environment provided by these tools. Pre-constructed virtual models were also made available to the entire class to facilitate further exploration of many of these concepts on an individual basis.

As shown in Figure 3, in the *first* phase, students begin with a series of simple case studies that are intended to familiarize the student with some of the basic functions in the MapleSim environment with the help of the examples and theory they learn in classroom. In the *second* phase, given some basic models of AMBS examples, students make advanced model in MapleSim and use what they have learnt in the previous steps to study the functional performance of these mechanisms (see Figure 4). Finally, in the *third* phase, students use the knowledge gained from this tutorial to support the final course project. The final project requires the student to use the software to explore different options in their designed model, such as interactively alter

parameters and location of actuation, of their specific designed models and come up with the final design, which meets the specifications. These case-studies have a natural hierarchical staging in the form of increasing “problematic” components at subsequent levels as students gain mastery at the initial levels [1].

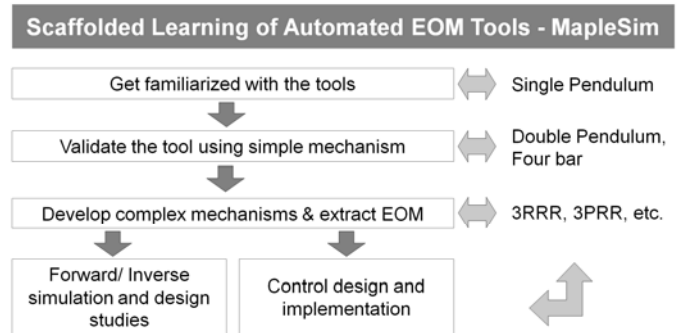


Figure 3: The tutorial is intended for scaffolded learning of the tool (MapleSim) with examples range from simple pendulum to complex planar parallel mechanism, complementing the course lectures.

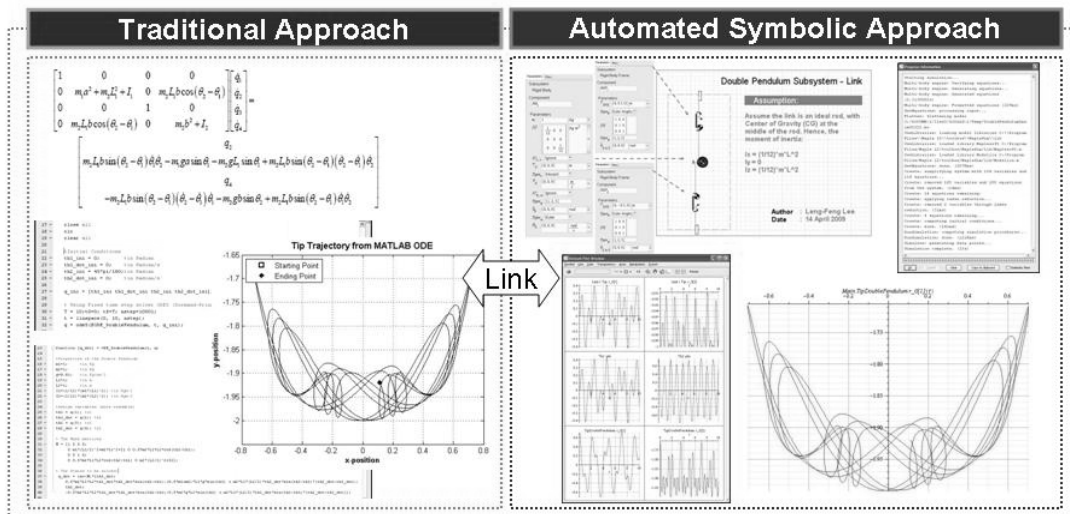


Figure 4: Comparison of forward dynamic simulation in traditional approach and automated symbolic approach, with double pendulum as an example.

In addition, we also incorporated erroneous (but “intuitive”) directives at several places within the tutorials. This is intended to force students to make common but undesirable mistakes and to then benefit from the experiential learning process. From this perspective, our approach is closer to Linn’s framework for scaffolded knowledge integration [7] which emphasizes the merits of such trial-and-error learning as well as the applicability of multiple equally-valid alternate approaches to problem solution. Ultimately, the principal desired outcome of these tutorials is to promote the development of *cognitive inquiry* within the student and accelerating their learning experience without increasing the lecture hours. At the same time the created framework also helps us address the more immediate goals of reinforcement of concepts being presented in the course by paralleling the course material in the case-studies.

Case Study 1: Single Pendulum

The first case study is a pendulum shown in Figure 5 (a). In this first tutorial, the goal is allow the student to (i) get familiarized with the interface (i.e. how to connect various blocks to create a system); (ii) explore various features provided by MapleSim (i.e. how to set the initial conditions); and (iii) analyze and visualize the result. The tutorial is introduced as a problem statement:

“A pendulum of length of L , as shown in Figure 5(a), with a mass m is pulled back to reach an initial angle of θ_0 from the vertical reference line and then released from rest. Determine the velocity and the reaction force over the entire period of the mass.”

This problem was selected to be the first example both from the viewpoint of its simplicity as well as its familiarity to the students. We demonstrate the process of modeling and the solution first by the traditional analytical approach, and then demonstrate the same process in MapleSim.

In the analytical approach, we discuss the following staged solution process:

- (1) Simplify the problem by appropriate assumptions, such as treating the pendulum inertia is same as an idealized rod.
- (2) Draw the free body diagram of the simplified model.
- (3) Develop the appropriate governing EOM using Newton’s Laws of Motion as well as using Lagrangian method.
- (4) Solve the EOM to obtain the desired solution.

In the MapleSim approach, the students are required to convert the simplified model into a block-based model as shown in Figure 5 (b). The tutorial guides the students to create the parts, and subsequently assemble it at the desired initial configuration in MapleSim environment. The model can now be simulated in order to visualize its motion and extract the time history of important parameters.

However, it is important to motivate the students to exercise good engineering judgment while analyzing many of these virtual models. For example, since the simulations are done numerically, students were required to use their engineering knowledge to detect any inconsistencies, if present. In this pendulum example, the expected velocity vs. angle graph is well understood (by even the novice students). Other intuitive cross validations such as equilibrium positions or resulting Cartesian trajectory were also incorporated.

We also adopted the approach of first creating scenarios that caused errors, and then working the students through the process of resolving these errors. For example, a physical link needs to be modeled as a compound pendulum (with both mass and inertia) whereas students would typically forget to model the rotational inertia. The tutorial guided the students to recognize this difference, and work their way to simulate its effects hence understand it better.

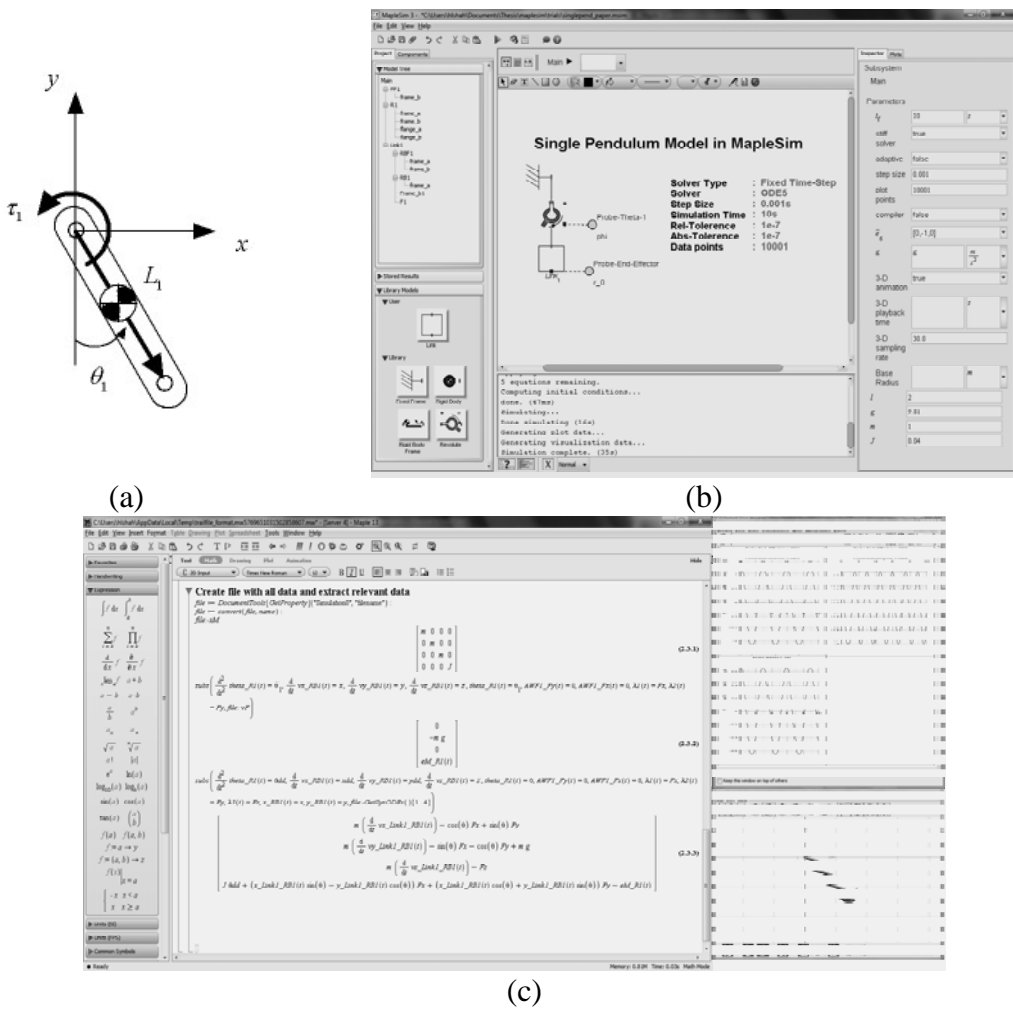


Figure 5: (a) Single pendulum problem; (b) modeled in MapleSim environment; (c) generated EOMs, plot of results from forward dynamic simulation, and visualization of the pendulum in motion.

Case Study 2: Planar Four-Bar Mechanism

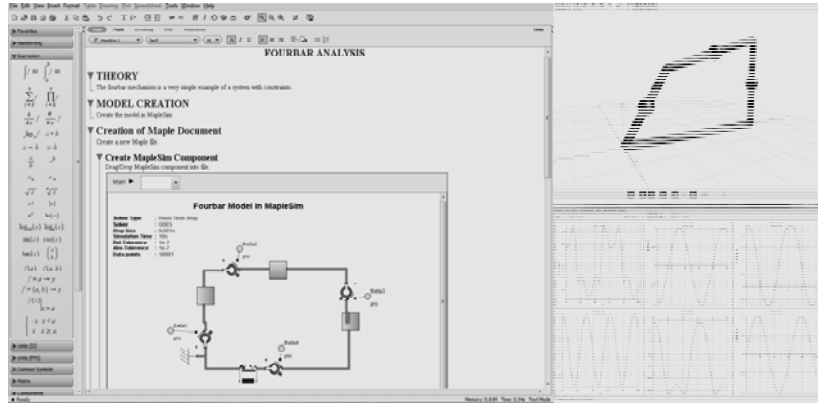
In the second phase, we examine more complicated examples of different planar mechanisms (double pendulum and four-bar, which are usually assigned to student as homework). The four-bar introduces a constrained mechanical system with dynamics couplings among the linkages. A four-bar, as shown in Figure 6 (a), with the problem statement described as the following:

“Given a four bar mechanism, formulate the dynamic equations of motion using Lagrangian method and compare the results with MapleSim automated EOMs generation.

“Perform forward dynamic simulation of the four bar system under the effect of gravity.

Compute the required torque for input angle to follow a sine wave.”

Using MapleSim, it is also possible to cross check and reinforce results obtained from different formulations against each other. E.g. we can compare symbolic expression for constraint forces obtained by hand calculation with the MapleSim result. Thus, at every stage of the kinematic and dynamic analysis, the correspondence between the analytical approach and the MapleSim approaches are emphasized. Finally, it is also important to have students aware of the trade-off between the symbolic equation generation and corresponding computation power by the tools and the accuracy of the results.



(a)

Lagrangian Eq. w.r.t. θ_1

$$\frac{1}{4} m_1 (\dot{\theta}_1)^2 + \frac{1}{2} m_2 [2(\dot{\theta}_1)^2 + 2L_1 \dot{\theta}_1^2 \cos \theta_2 - 2L_1 \dot{\theta}_1 L_2 \sin \theta_2 \dot{\theta}_2]$$

$$+ L_1 L_2 \dot{\theta}_2^2 \cos^2 \theta_2 - L_2^2 \dot{\theta}_2^2 \sin^2 \theta_2 + \frac{1}{2} L_2^2 \dot{\theta}_2^2 + \frac{1}{2} L_2^2 \dot{\theta}_2^2 + I_1 \dot{\theta}_1 + I_2 (\dot{\theta}_1 + \dot{\theta}_2) + \frac{1}{2} m_2 g L_1 \cos \theta_1 + m_2 g (L_1 \cos \theta_1 + \frac{1}{2} L_2 \cos(\theta_1 + \theta_2))$$

Lagrangian Eq. w.r.t. θ_2

$$\frac{1}{2} m_2 [L_1 \dot{\theta}_1 L_2 \cos \theta_2 - L_1 \dot{\theta}_1 L_2 \sin \theta_2 \dot{\theta}_2 + \frac{1}{2} L_2^2 \dot{\theta}_2^2 + \frac{1}{2} L_2^2 \dot{\theta}_2^2] + I_2 (\dot{\theta}_1 + \dot{\theta}_2) - \frac{1}{2} m_2 [-L_1 \dot{\theta}_1^2 L_2 \sin \theta_2 - L_1 \dot{\theta}_1 L_2 \sin \theta_2 \dot{\theta}_2] + \frac{1}{2} m_2 g L_2 \cos(\theta_1 + \theta_2)$$

Lagrangian Eq. w.r.t. θ_4

$$\frac{1}{4} m_3 \dot{\theta}_4^2 + I_3 \dot{\theta}_4 + \frac{1}{2} m_3 g L_3 \cos(\theta_4 + \theta_3)$$

Collecting \ddot{q} terms in above equations:-

$$M = \begin{bmatrix} \frac{1}{4} m_1 L_1^2 + m_2 L_1^2 + m_2 L_1 L_2 \cos \theta_2 & \frac{1}{2} m_2 L_1 L_2 \cos \theta_2 & 0 \\ \frac{1}{2} m_2 L_1 L_2 \cos \theta_2 & \frac{1}{2} m_2 L_2^2 + I_2 & 0 \\ 0 & 0 & \frac{1}{4} m_3 L_3^2 + I_3 \end{bmatrix}$$

Collecting \ddot{q} and gravity terms:-

$$Cq + F = \begin{bmatrix} -m_2 L_1 \dot{\theta}_1 L_2 \sin \theta_2 \dot{\theta}_2 - \frac{1}{2} m_2 L_2^2 \dot{\theta}_2^2 \sin \theta_2 \\ \frac{1}{2} m_2 g L_1 \cos \theta_1 + m_2 g L_1 \cos \theta_1 + \frac{1}{2} m_2 g L_2 \cos(\theta_1 + \theta_2) \\ \frac{1}{2} m_2 L_2 (L_1 \dot{\theta}_1^2 \sin \theta_2 + g \cos(\theta_1 + \theta_2)) \\ \frac{1}{2} m_3 g L_3 \cos(\theta_4 + \theta_3) \end{bmatrix}$$

(b)

Comparison for Fourbar.mw
 Test Math Drawing Plot Animation
 C Text Test New Roman

Comparison of Various Matrices with Lagrangian Formulation

Mass Matrix
 Now we copy the xM from the Lagrangian Modeling file attached with this model

$$xM = \begin{bmatrix} \frac{1}{4} m_1 L_1^2 + m_2 L_1^2 + m_2 L_1 L_2 \cos(\theta_2) + \frac{1}{4} m_2 L_2^2 + J_1 + J_2 & \frac{1}{2} m_2 L_1 L_2 \cos(\theta_2) + \frac{1}{4} m_2 L_2^2 + J_2 & 0 \\ \frac{1}{2} m_2 L_1 L_2 \cos(\theta_2) + \frac{1}{4} m_2 L_2^2 + J_2 & \frac{1}{4} m_2 L_2^2 + J_2 & 0 \\ 0 & 0 & \frac{1}{4} m_3 L_3^2 + J_3 \end{bmatrix}$$

Get it to the required format by substitution
 $xM = \text{subs}(\theta_1 = \theta_1, \theta_2 = \theta_2, \theta_4 = \theta_4, xM)$

Now we find the difference in the two obtained matrices; `error, missing operator or '.'`
 $\text{diff}xM = \text{file}xM - xM$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (3.5.1.1)$$

And the matrices are a perfect match; `error, missing operator or '.'`

Force Vector
 Now we copy the vF from the Lagrangian file

$$vF = \begin{bmatrix} -m_2 L_1 \left(\frac{\partial}{\partial t} \theta_1 \right) L_2 \sin(\theta_2) \left(\frac{\partial}{\partial t} \theta_2 \right) - \frac{1}{2} m_2 L_1 L_2 \left(\frac{\partial}{\partial t} \theta_2 \right)^2 \sin(\theta_2) + \frac{1}{2} m_2 g L_1 \cos(\theta_1) + m_2 g L_1 \cos(\theta_1) + \frac{1}{2} m_2 g L_2 \cos(\theta_1 + \theta_2) \\ \frac{1}{2} m_2 L_2 \left(L_1 \left(\frac{\partial}{\partial t} \theta_1 \right)^2 \sin(\theta_2) + g \cos(\theta_1 + \theta_2) \right) \\ \frac{1}{2} m_3 g L_3 \cos(\theta_4 + \theta_3) \end{bmatrix}$$

We convert to the required form by substitution and take the difference for verification
 $vF = \text{subs}(\theta_1 = \theta_1, \theta_2 = \theta_2, \theta_4 = \theta_4, vF)$
 $\text{diff}vF = \text{comp}hfy(vF + \text{file}vF)$

$$\begin{bmatrix} eM_{\theta_1}(t) - eM_{\theta_1}(t) \\ eM_{\theta_2}(t) - eM_{\theta_2}(t) \\ eM_{\theta_3}(t) + eM_{\theta_3}(t) \end{bmatrix} \quad (3.5.2.1)$$

(c)

Figure 6: (a) Fourbar mechanism modeled in MapleSim and its corresponding forward dynamic analysis; (b) analytical EOMs generated by using Lagrangian method (selectively shown only the EOMs and Mass matrix); and compare the EOMs with (c) EOMs generated using MapleSim.

Case Study 3: Complex Planar Parallel Manipulator

In the final phase (course final project), students were asked to model and analyze variants of planar parallel mechanisms (e.g. 3PRR , 3RRR , 3RPR). These systems introduce multiple closed loop constraints as compared to a four bar system. They are required to formulate the EOMs of the given system, both analytically (using Lagrangian method) and using MapleSim. After formulation of the EOMs, they were required to design a trajectory tracking controller. Additionally, the students were required to analyze system level performance (such as manipulability and workspace).

Figures 7 (a) and (c) show a 3PRR parallel manipulator example while Figures 7 (b) and (d) show a 3RRR example. The students are required to first solve the forward and inverse kinematics of the system, followed by formulating the EOMs analytically using Lagrangian method. Parallel to this work, students also create the constrained multibody

system in MapleSim, followed by generating EOMs symbolically. Figure 8 shows the EOMs generated by (a) analytical approach and compares the result with (b) automated MapleSim result.

Using such model, students could then compute and analyze manipulator end-effector motion for various inputs. E.g. Figure 9(a) and (f) shows the results for a 3RRR manipulator with a constant -0.5N/m torque applied at three active joints for 5 seconds. Various joint angles can also be plotted to study the range of motion (workspace) of the manipulator parametrically, as shown in Figure 9 (g).

MAE501: "Independent study" Deployment[2]

The Rotary Inverted pendulum (ROTPEN) [3], also known as a Furuta pendulum (shown in Figure 10) is a commonly used project example in various control cases. It is a typical multivariable, underactuated, nonlinear and unstable system.

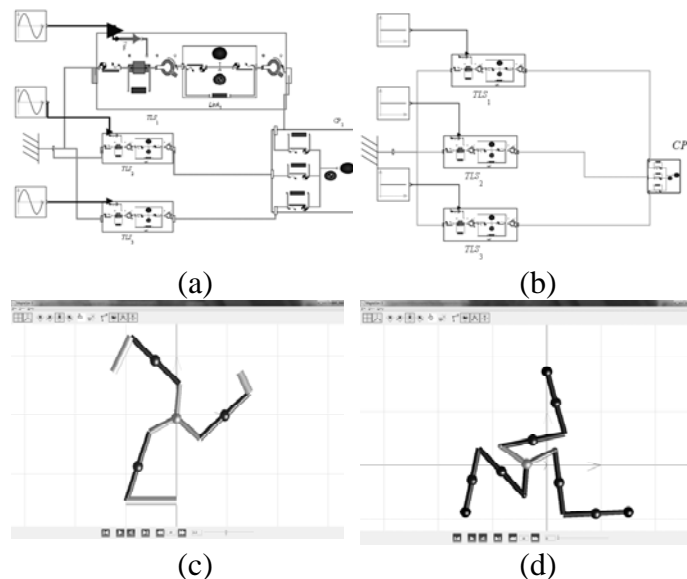


Figure 7: MapleSim model of (a) 3PRR and (b) 3RRR manipulator and their corresponding visualizations shown in (c) and (d).

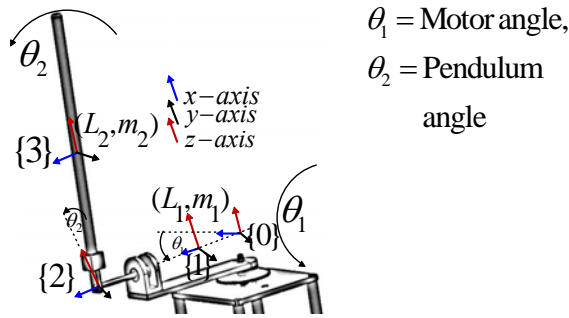
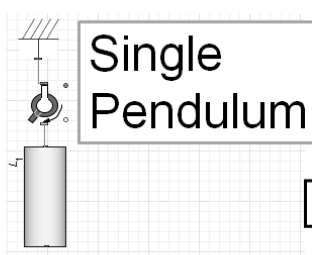


Figure 10: The Rotary Inverted Pendulum (ROTPEN), also called the Furuta pendulum.



Single Pendulum

```

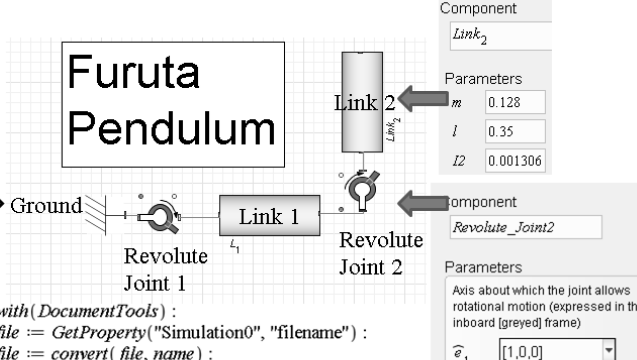
with(DocumentTools):
file := GetProperty("Simulation0", "filename"):
file := convert(file, name):
M := file:-xM:

```

Mass Matrix

$$M := \text{simplify}(\text{subs}(\text{theta_R3}(t) = \text{th2}, M), \text{trig})$$

$$\begin{bmatrix} I1 + \frac{1}{4} L1^2 m1 \end{bmatrix}$$



Furuta Pendulum

```

with(DocumentTools):
file := GetProperty("Simulation0", "filename"):
file := convert(file, name):
M := file:-xM:

```

Mass Matrix

$$M := \text{simplify}(\text{subs}(\text{theta_R3}(t) = \text{th2}, M), \text{trig})$$

$$\begin{bmatrix} I2 + \frac{1}{4} m2 L2^2 & -\frac{1}{2} \cos(\text{th2}) L2 m2 L1 \\ -\frac{1}{2} \cos(\text{th2}) L2 m2 L1 & I1 + \frac{1}{4} L1^2 m1 + L1^2 m2 + \frac{1}{4} m2 L2^2 - \frac{1}{4} m2 L2^2 \cos(\text{th2})^2 \end{bmatrix}$$

(a)

(b)

"Dynamic analysis complete."
 "Saving model for future use..."
 "Model saved to: F:/Maple_sim/Pendulum/Pendulum.msmlib"
 "Use `Model := MapleSim:-Multibody:-GetModel("F:/Maple_sim/Pendulum/Pendulum.msmlib"):` to retrieve stored model"

```

with(DocumentTools):
file := GetProperty("Simulation0", "filename"):
file := convert(file, name):
M := file:-xM:
M := simplify(subs(theta_R3(t) = th2, M), trig)

```

Difference of Lagrangian modeling and MapleSim auto generated Mass matrix: {0,0,0,0}

$$\begin{bmatrix} I2 + \frac{1}{4} m2 L2^2 & -\frac{1}{2} \cos(\text{th2}) L2 m2 L1 \\ -\frac{1}{2} \cos(\text{th2}) L2 m2 L1 & I1 + m2 L1^2 + \frac{1}{4} m2 L2^2 - \frac{1}{4} m2 L2^2 \cos(\text{th2})^2 + \frac{1}{4} L1^2 m1 \end{bmatrix}$$

#Equations Calculated From Lagrangian Modeling

$$M1 := \begin{bmatrix} I2 + \frac{1}{4} m2 L2^2 & -\frac{1}{2} m2 L1 L2 \cos(\text{th2}) \\ -\frac{1}{2} m2 L1 L2 \cos(\text{th2}) & I1 + m2 L1^2 + \frac{1}{4} m2 L2^2 - \frac{1}{4} m2 L2^2 \cos(\text{th2})^2 + \frac{1}{4} m1 L1^2 \end{bmatrix} : \{ \}$$

$\text{simplify}(M - M1)$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

(c)

Figure 11: Mass matrix generated from MapleSim template (a) single pendulum; (b) single pendulum modified to Furuta pendulum; and (c) comparison of MapleSim generated mass matrix and hand coded result of mass matrix.

Modeling

We guide the student through the model deployment with the approach discussed earlier. Converting a planar single pendulum model into spatial Furuta pendulum was easily accomplished by attaching a copy of the original model to its base and changing the copies' parameters in the MapleSim GUI as shown in Figure 11. The rest of the steps for extracting model equations were unaltered.

Although a virtual model can be actuated with a simple idealized torque driver, we explored incorporation of real world actuators e.g. DC motor. The modularity of MapleSim components allows us to create a DC motor model separately and attach it to Furuta pendulum model. The DC motor equations shown in Figure 12 get included implicitly into the system model and need not be calculated separately.

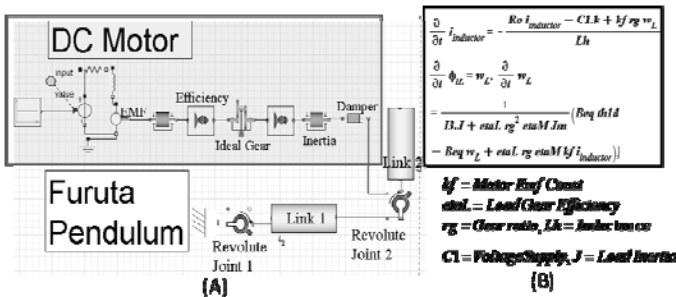


Figure 12: (a) DC motor model attached to Furuta pendulum (b) DC motor equations.

Controller

Linear controller design techniques require a model expressed in linear state space format. The conversion process may be automated by attaching a simple Maple template, like the one shown in Figure 13, to the model. The tutorials guide the user's controller design process with linear controller design in order to facilitate subsequent comparative study between linear and model based controllers. An outline of the adopted approach for LQR control design is illustrated in Figure 14. However, a detailed discussion is beyond the scope of the current paper.

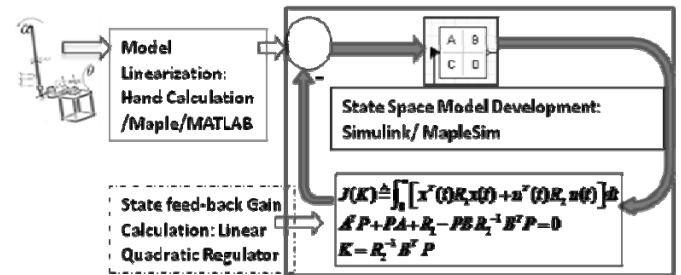


Figure 14: Steps to design conventional linear controllers.

```

Linearization
#f is vector of symbolic expression for Dynamics Equation of the system
#Taylor series expansion with respect to q and qdot
Ass := subs(f, q, Matrix(q_length, 1, 0)) + subs(jacobian(f, q), q, Matrix(q_length, 1, 0)).q :
Ass := subs(Ass, qdot, Matrix(qdot_length, 1, 0)) + subs(jacobian(f, qdot), qdot, Matrix(qdot_length, 1, 0)).qdot :
Asym := Matrix(q_length, 2*q_length) :
X := (q, qdot) :
for j from 2*q_length by -1 to 1 do for i from 1 by 1 to q_length do
Asymx[i, j] := select(has, Ass[i, 1] + ignore, X[j, 1]) :
Ass[i, 1] := Ass[i, 1] - Asymx[i, j] : Asym[i, j] := simplify(Asymx[i, j] / X[j, 1]) : od od :
A := Matrix([[Matrix([Matrix(q_length, q_length, shape = identity), Matrix(q_length, q_length, 0)]), [Asym]])]
B := Matrix([[Matrix(q_length, 1, 0)], [Ass]]) :
    
```

Figure 13: Automated script for linearization of AMBS to obtain model in linear state-space format.

For the given single-input multi-output system, we use a cascaded control approach [4]. This implies that we have separate sets of gains for motor states and pendulum states. To allow for a comparative study, a configuration was chosen so that it would be stable with both controllers. Thereafter, gains for motor states were fixed and performances of two controllers with same ranges of gains of pendulum states were compared. The performance functional was selected as:

$$\text{Performance functional} = \frac{1}{\int_0^t (X^T R_1 X + u^T R_2 u) dt}$$

where $X = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$, $u =$ control law.

$R_1 = \text{diag}(0.25, 4, 0.01, 1)$, $R_2 = 0.2$
 R_1 & R_2 are state and input weighting matrices.

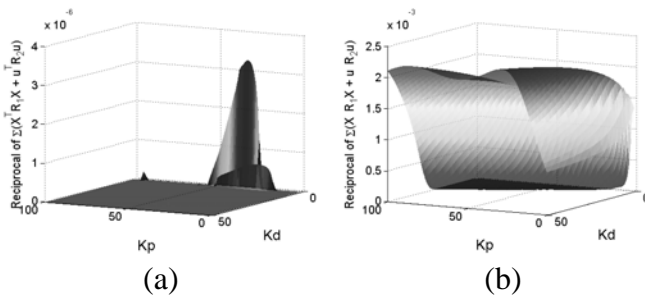


Figure 15: Performance comparison between LQR and model based controller with respect to change in K_p and K_d (a) LQR (b) model based controller (Input-output linearization); \int_0^t Norm of performance functional = (a) 0.000013; and (b) 0.0949.

Figure 15 compares the performance functionals of both controllers for variation in gains of the pendulum states (proportional- K_p and derivative- K_d gains). We observe that the model based controller has a wider range of controller gains as compared to the LQR based controller.

Implementation of Model Based Controller

As an example, feedback linearization control methodology was deployed using two copies for the same model in MapleSim. One served as the inverse dynamics model to create the required torque for a given motion profile while the other served as the physical plant being controlled with the generated torque profile. This arrangement canceled out non-linearities in the system thereby allowing control of the system using an additional PD controller, which is designed easily using linear control techniques. Figure 16 (a) highlights the details of the controller and the plant model.

This control technique was extended to Simulink by using MapleSim Connector Toolbox, which exports any subsystem of MapleSim to an S-function in Simulink. The controller developed in MapleSim was validated against a SimMechanics model of the system, as shown in Figure 16 (b).

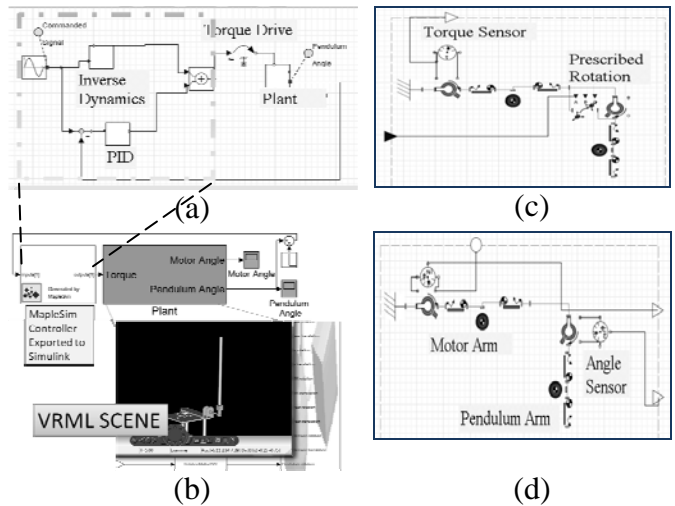


Figure 16: (a) Input-Output Linearization with MapleSim blocks (b) SimMechanics model controlled with MapleSim controller. (c) MapleSim inverse dynamics (d) plant model.

In Figure 17 (a) and (b), we observe that the commanded signal was tracked as required. However, the motor angle response, as seen in Figure 17 (c), becomes unstable since it is a part of the internal dynamics (and not accounted for

in the current control model). Thus, while the functionality of sensor-based inverse dynamics is available in SimMechanics, MapleSim provides better insight into the system equations, which become relevant in the subsequent analysis.

Unstable Internal Dynamics

The Equations generated from MapleSim and subsequent symbolic computation based on input-output linearization [5] yields the form of controller (without resorting to controller gain tuning techniques) and thus achieve control over the internal dynamics.

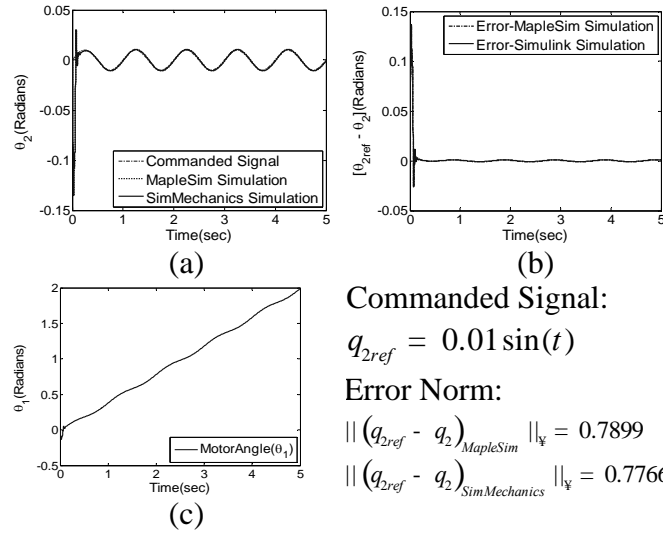


Figure 17: (a) Response of θ_2 , a comparison between MapleSim and SimMechanics Model response, (b) error between reference θ_2 and system θ_2 response, and (c) response of θ_1 .

The automatically generated equations from MapleSim are shown below for the readers benefit.

$$M = \begin{bmatrix} \frac{1}{3}m_1L_1^2 + m_2L_1^2 + \frac{1}{4}m_2L_2^2 - \frac{1}{4}m_2L_2^2c_2^2, & -\frac{1}{2}m_2L_1L_2c_2 \\ -\frac{1}{2}m_2L_1L_2c_2, & \frac{1}{3}m_2L_2^2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{1}{2}m_2L_2^2 \cos(\theta_2) \sin(\theta_2) \dot{\theta}_2, & \frac{1}{2}m_2L_1L_2 \sin(\theta_2) \dot{\theta}_2 \\ -\frac{1}{4}m_2\dot{\theta}_1L_2^2 \cos(\theta_2) \sin(\theta_2), & 0 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$G = \begin{bmatrix} 0 \\ -\frac{1}{2}m_2gL_2 \sin(\theta_2) \end{bmatrix} = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

The overall dynamic EOM was written as:

$$M_{11}\ddot{\theta}_1 + M_{12}\ddot{\theta}_2 + C_{11}\dot{\theta}_1 + C_{12}\dot{\theta}_2 + G_1 = \tau_1 - Beq\dot{\theta}_1 \quad (1)$$

$$M_{21}\ddot{\theta}_1 + M_{22}\ddot{\theta}_2 + C_{21}\dot{\theta}_1 + C_{22}\dot{\theta}_2 + G_2 = 0 \quad (2)$$

where “*Beq*” is motor damping constant. An explicit relation between input τ_1 and output θ_2 can be obtained from Eqn. (1) and (2) as:

$$\ddot{\theta}_2 = \frac{\tau}{\left(M_{12} - \frac{M_{11}}{M_{21}}\right)} + F(\theta) \quad (3)$$

$$F(\theta) = \frac{\frac{M_{11}}{M_{21}}(C_{21}\dot{\theta}_1 + C_{22}\dot{\theta}_2 + G_2) - C_{11}\dot{\theta}_1 - C_{12}\dot{\theta}_2 - G_1 - Beq\dot{\theta}_1}{\left(M_{12} - \frac{M_{11}}{M_{21}}\right)}$$

The control input is chosen to be of the form:

$$\tau = \left(M_{12} - \frac{M_{11}}{M_{21}}\right) \{v - F(\theta)\} \quad (4)$$

where v is the secondary input, which creates the linear system:

$$\ddot{\theta}_2 = v \quad (5)$$

This secondary input v can be designed by standard linear pole placement techniques to drive error dynamics to zero. We choose $v = \ddot{\theta}_{2d} - K_{p2}e - K_{d2}\dot{e}$, where $e = \theta_2 - \theta_{2d}$ is the tracking error, θ_{2d} is desired pendulum angle and K_{p2} , K_{d2} is positive constants, which results in $\ddot{e} + K_{p2}e + K_{d2}\dot{e} = 0$, a *Hurwitz Polynomial* that represents exponentially stable error dynamics.

However, the error dynamics is 2nd order and only account for part of the 4th order system dynamics. Thus, a part of the system dynamics, called *internal dynamics*, is rendered unobserved in the input-output linearization approach. We implemented cascade-control for achieving internal dynamics stabilization. And details are presented in [13]. The cascade control law, as shown in Figure 18, is a two part design with an inner loop:

$$\tau = \left(M_{12} - \frac{M_{11}}{M_{21}} \right) \{v - F(\theta)\}, \quad (6)$$

$$v = 0 - K_{p2}e_2 - K_{d2}\dot{e}_2$$

where $e_2 = \theta_2(t) - \theta_{2d}(t)$ and an outer loop:

$$\theta_{2d} = K_{p1}e_1 + K_{d1}\dot{e}_1 \quad (7)$$

where $e_1 = \theta_1 - \theta_{1d}$. High gains were selected to converge $\dot{\theta}_2$ and $\ddot{\theta}_2$ to zero and θ_2 to θ_{2d} in a very short time interval.

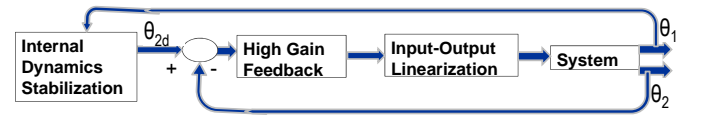


Figure 18: Cascade control.

Output Linearization

```
Matrix([[th1dd], [th2dd]]) #Define joint acceleration vector
Matrix([[th1d], [th2d]]) #Define joint velocity vector
f.thdd + C.thd + G #M=Mass Matrix, C=Cristoffel Matrix,
l2=0 (There is no motor at second joint)
```

```
m := simplify( (select(has, tou[2, 1], th1dd) ) / th1dd ):
m := -remove(has, tou[2, 1], th1dd) :
l := th1dd_num / th1dd_den # Calculate d^2 theta_1 in terms of d^2 theta_2
sing d^2 theta_1 in tou1, we know tou1=U
l := collect(subs(tou[1, 1], [th1dd], [th1ddsym]), th2dd) :
m := select(has, temp_tou1, th2dd) :
m := th2dd
m := U - Beq.th1d - remove(has, temp_tou1, th2dd) :
nbine( simplify( (th2dd_num - U) ) ) :
th2dd_den * (v - fx) #Calculate Control Law for theta_2 stabiliza
```

(a)

$$\frac{1}{64} \frac{1}{m^2 L_1 L_2 \cos(\theta_2)} (64 B e q \theta_1 d m^2 L_1 L_2 \cos(\theta_2) - 16 m^2 L_2 \sin(\theta_2) m_1 L_1^2 g + 8 v m_1 L_1^2 m_2 L_2^2 - 4 m^2 L_2^2 m_1 L_1^2 \theta_1 d^2 \sin(2 \theta_2) + 8 m^2 L_2^3 \theta_2 d \theta_1 d L_1 \sin(3 \theta_2) + 8 m^2 L_2^3 \sin(\theta_2) \theta_2 d \theta_1 d L_1 - 2 m^2 L_2^4 \theta_1 d^2 \sin(2 \theta_2) + m^2 L_2^4 \theta_1 d^2 \sin(4 \theta_2) + 4 m^2 L_2^3 g \sin(3 \theta_2)] - 16 m^2 L_2^2 I_1 \theta_1 d^2 \sin(2 \theta_2) - 16 m^2 L_2^2 L_1^2 \theta_1 d^2 \sin(2 \theta_2) + 16 m^2 L_2^2 L_1^2 \theta_2 d^2 \sin(2 \theta_2) - 64 m^2 L_2 \sin(\theta_2) I_1 g - 64 m^2 L_2 \sin(\theta_2) L_1^2 g + 4 v m^2 L_2^4 + 128 v I_1 I_2 + 32 v I_1 m^2 L_2^2 + 128 v m^2 L_1^2 + 16 v m^2 L_1^2 L_2^2 + 16 v m^2 L_2^2 I_2 + 32 v m_1 L_1^2 I_2 - 4 v m^2 L_2^4 \cos(2 \theta_2) - 12 m^2 L_2^3 \sin(\theta_2) g - 16 v m^2 L_2^2 I_2 \cos(2 \theta_2) - 16 v m^2 L_2^2 L_1^2 L_2^2 \cos(2 \theta_2))$$

(b)

Figure 19: (a) Input output linearization script for Furuta pendulum (b) symbolic expression for Control law to stabilize q_2 .

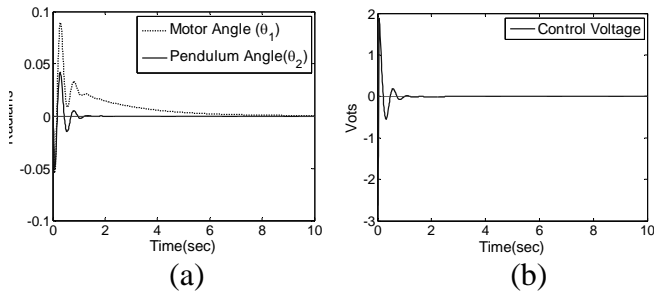


Figure 20: Controller response data from MapleSim, (a) joint angles (b) control effort required in the form of motor voltage.

An automated script, as shown in Figure 19, was programmed within Maple and attached to a MapleSim model to make the controller design reusable for other similar systems, e.g. planar double pendulum. A custom component

for the controller was generated in MapleSim and simulation results are shown in Figure 20.

Thus, the full cycle of modeling and controller design was achieved within Maple/MapleSim. Students can augment their learning by following these step by step procedures without hand calculating the system model.

Real Time Experimental evaluation

Following controller design and validation using simulation, HIL testing was conducted. The controller developed in MapleSim was exported to a Simulink S-function block for real time application using the MapleSim Connector Toolbox. The results of implementation of this real-time controller on the physical Furuta pendulum setup are shown in Figure 21.

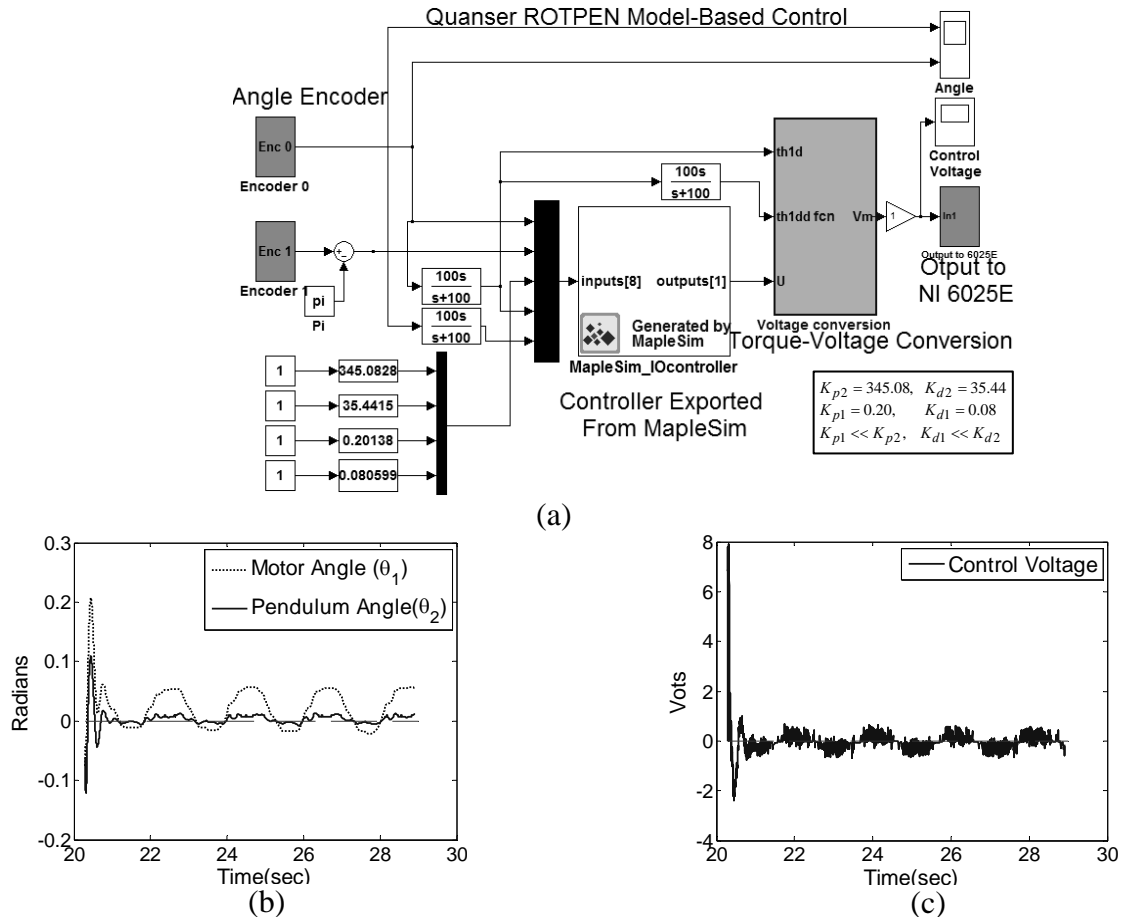


Figure 21: (a) Simulink block diagram with MapleSim controller; (b) Furuta pendulum configurations; and (c) control voltage

Most of the students were able to complete and check against the kinematics simulation. This was mainly because the dynamic equations were very cumbersome to calculate manually and generally contained errors. Detailed symbolic manipulation could not be explored due to lack of tutorials explaining how equations could be extracted and interpreted. This is a shortcoming that our future tutorials will seek to address.

In addition to the ease of model generation and model based control design, we also explored other benefits of symbolic computation. Sensitivity analyses of controller performance require accurate (non-linear) simulation model which tend to be computationally expensive. This can be significantly enhanced with symbolic computation, e.g. calculating torque sensitivity with respect to change in mass of pendulum.

Thus, from the instructional viewpoint this proved to be a viable vehicle for bridging the gap between the conventional classroom-based approaches for teaching mechanisms and robotics and an experiential approach. In terms of instructional support, very little was required to support the course (after the initial investment of effort and time in the tutorial).

Video recordings of these tutorials were also created and available to student [8, 12]. These videos show step-by-step instructions on the creation of a MapleSim model, extracting EOMs, to performing various simulations. The level of detail in the step-by-step instructions is a factor that we will be investigating further in future work. We also plan to gradually increase the number and complexity of intermediate “mini-projects” to permit the students to get hands-on practice in engineering problems with different levels of complexity. Finally, a careful quantitative evaluation of the effectiveness of these tutorials is an important issue which still needs to be addressed.

Acknowledgments

We gratefully acknowledge the support from the Research Foundation of State University of New York, National Science Foundation

CAREER Award (IIS-0347653) and CNS-0751132.

References

1. U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations*. Philadelphia, PA: SIAM, 1998.
2. A. A. Shabana, *Computational Dynamics*. New York, NY: Wiley, 2001.
3. K. J. Waldron and G. L. Kinzel, *Kinematics, Dynamics, and Design of Machinery*. New York, NY: Wiley, 2003.
4. J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, N.J.: Prentice Hall, 2006.
5. Maplesoft, "MapleSim," ed. Waterloo, Ontario, Canada., 2009.
6. G. Adams and I. Jong, "Using Matlab to Animate the Generation of a Space Centrode in Kinematics," in *ASEE 1997 Annual Conference*, Milwaukee, WI, 1997.
7. M. Linn, "Designing Computer Environments for Engineering and Computer Science: The Scaffolded Knowledge Integration Framework," *Journal of Science Education and Technology*, vol. 4, pp. 103-106, 1995.
8. H. Shah and S. Tripathi. (2010, April 10). *MAE 501 Tutorials*. Available:<http://www.buffalo.edu/education/maplesim>
9. J. Hiebert, *et al.*, "Problem Solving as a Basis for Reform in Curriculum and Instruction: The Case of Mathematics," *Educational Researcher*, vol. 25, pp. 12-21, 1996.
10. Quanser. (2010, *Rotary Pendulum Experiment*. Available:http://www.quanser.com/english/downloads/products/Rotary/SR_V02_ROT PEN_PIS_031108.pdf

11. K. Guemghar, *et al.*, "Predictive control of fast unstable and nonminimum-phase nonlinear systems," in *Proceedings of 2002 American Control Conference, 8-10 May 2002*, Danvers, MA, USA, 2002, pp. 4764-9.
12. H. Shah. (2010, April 10). *MAE 513 MapleSim Tutorials*. Available:<http://sites.google.com/site/maplesimtutorial/>
13. S. Tripathi, "Role of Symbolic Computation in Linear and Model-Based Controller Development," Mechanical & Aerospace Engineering, University at Buffalo, Buffalo, 2010.

Biographical Information

Hrishi Shah is an M.S. student in mechanical engineering at the State University of New York at Buffalo with research interests in various aspects of multibody dynamics and control. He graduated with a bachelor's degree in Mechanical Engineering from National Institute of Technology, Calicut, India in 2007.

Sumit Tripathi received his Bachelor of Engineering in marine engineering from Jadavpur University, India, in 2003. He is pursuing an M.S. in mechanical engineering at the State University of New York at Buffalo. His research interests include modeling and nonlinear control of robotics and autonomous systems, and the development of real time software and mechatronics applications.

Leng-Feng Lee is currently working towards his Ph.D. degree in Mechanical Engineering at the State University of New York at Buffalo with a focus on design and control of novel articulated multibody systems for various applications. He graduated with a B.S. and M.S. degree in mechanical engineering from the same institution in 2003 and 2005 respectively.

Venkat Krovi (S'97–M'99) received the M.S. and Ph.D. degrees from the University of Pennsylvania, Philadelphia, in 1995 and 1998, respectively, both in mechanical engineering. In 2001, he joined the Department of Mechanical and Aerospace Engineering, State University of New York, Buffalo, where he is currently an Associate Professor. His research interests include design, analysis, and prototyping of novel articulated mechanical systems. Dr. Krovi was the recipient of the 2003 National Science Foundation CAREER Award.