Design of Inverted Pendulum System Using SysML

Kritsana Uttamang, Ph.D.
SysML Application Engineer
Abstract: In this paper, SysML (System Modeling Language) has been used to design the inverted pendulum system whose pendulum is controlled to remain upright by a feedback controller. We use a state-space model to design our feedback controller and perform simulation to ensure that the designed controller gain can be used to meet the required response. The result of this work shows that SysML is very helpful for the system engineer to design a complex system, especially a heterogeneous system.

Keywords: SysML, Inverted Pendulum, Control of Dynamic System

1. Introduction

One of the most important and classic problems of control engineering is the inverted pendulum system. Different from the normal pendulum which is stable when hanging downward, the inverted pendulum is inherently unstable. It must be actively balanced to remain upright. Therefore, the system requires a controller, which can be software or firmware. The controller receives system state information from sensors, and performs calculations for the control signal. The control signal is sent to an actuator to control the pendulum.

Many things need to be considered by the system engineer for the design of the entire inverted pendulum system. SysML (System Modeling Language) can be used to help the system engineer handle the complexity of this system.

SysML is a graphic modeling language that has been developed for specifying, analyzing, designing and verifying a broad range of systems [1]. It extends the application of UML to systems which are not purely software based [2] including electrical systems, mechanical systems, physical systems, etc. This allows it to be applied to the design of a heterogeneous system, like our inverted pendulum system that is composed of hardware and software. The objective of this work is to design the inverted pendulum system by using SysML. The SysML tool that we have selected for use in this development is the SysML plug-in for MagicDraw.

Before constructing the real system, the system engineer needs simulations to be sure that each part of a designed system can operate together as intended. For this work, we are creating the inverted pendulum system simulator using java. This simulator can simulate motion of our designed inverted pendulum system corresponding with the feedback controller gain.

2. System Overview

2.1 Inverted Pendulum System

The inverted pendulum system is composed of a pendulum, which is a light rod length ($l$) with spherical ball mass ($m$) attached at one end while the other end is connected to a cart mass ($M$), as show in Fig.1. A motor applies torque ($\tau(t)$) to move the cart horizontally and control the pendulum so it will remain upright. This torque needs to be evaluated by the feedback controller, which we have designed by using the state-space model.

![Fig. 1 Inverted Pendulum System](image-url)
2.2 State-Space

State-space is a mathematical model that represents a physical system as a set of inputs, outputs and state variables related by first-order differential equation. By given system inputs as an input vector \( u \), system outputs as an output vector \( y \), and state variables are contained inside state vector \( x \), so the state-space equations are:

\[
\dot{x} = Ax + Bu \quad (2.1) \\
y = Cx + Du \quad (2.2)
\]

Where, state matrix \( A \) and input matrix \( B \) can be obtained from dynamic equations of system. Output matrix \( C \) is used to produce an output vector from state variables, which are obtained by solving the equation (2.1). Feed-forward matrix \( D \) is used to compensate a system loss.

From the state equation (2.1), the controller is designed to make a calculation for the input vector from the state vector by using the feedback equation:

\[
u = -Kx \quad (2.3)
\]

Where: \( K \) is feedback matrix

If the state equation of the system is known, then the feedback matrix \( K \) can be evaluated with various controller designed methods (i.e. symmetrical root locus, pole-zero replacement, Ackerman’s formula) to meet the required response. Moreover, the state equation can be used to make a controller simulation of the inverted pendulum system.

By assuming the feed-forward matrix \( D \) is zero-matrix, a feedback control loop can be represented as a diagram which is shown in Fig. 2(a). A feedback control loop consists of a plant and a controller. A plant is the physical part of the system, whose dynamics are described by the state equation. It receives a control signal from the controller to produce output states. A controller is the part of the system which receives the system states and calculates for control signal. A PC is used as the controller of the inverted pendulum system in this work as shown in Fig, 2(b).

![Fig. 2 (a) State-space feedback control loop and (b) plant and controller of inverted pendulum system](image-url)
2.3 Representing the system in SysML

SysML introduces the concept of blocks. Hierarchical systems are composed of parts which are typed by blocks. SysML also provide many kinds of diagram which can be used to model and represent the system from the different point of views. This concept has been used to model the inverted pendulum system as shown in Fig. 3(a) and Fig. 3(b).

Fig. 3 (a) Feedback control loop and (b) corresponding SysML blocks

3. Modeling with SysML

3.1 Requirements Management

SysML provides a requirement diagram to help system engineer captures the requirements of the system. By considering that the main requirement is: “To control the pendulum upright by moving the cart horizontally even though a disturbance is applied to the pendulum, moreover the user can change the controller gain of the feedback controller to meet the required response.” The requirement diagram and the use case diagram are created as shown in Fig. 4 and Fig. 5 respectively.
Fig. 4 Requirement diagram

Fig. 5 Use case diagram and corresponding features in simulation software
3.2 Blocks

SysML provides blocks to define types of parts and also provides a block definition diagram to show the composition of each part inside the system. To design the inverted pendulum system with SysML, we have to define our inverted pendulum system as a block and also type of each part inside it.

The inverted pendulum system is a feedback control system. It receives electrical power from an external power supply for the operations. If the inverted pendulum block is defined to represent the inverted pendulum system, then a plant and a controller must be parts of this block. By typing plant with pendulum assembly and controller with computer, pendulum assembly and computer then are defined as blocks. The pendulum assembly also consists of parts inside including pendulum, cart, motor, encoder, power supply, tendon mechanism and amplifier. All of these parts are hardware parts and are also defined as blocks.

For the computer block which represents the controller, it consists of both software and hardware parts. The controller application is designed as the main software part. It is composed of several software modules including state recorder, timer, feedback controller, and I/O API.

The hardware parts of the designed controller are the counter card and the data acquisition output card. The counter card receives gray coding from incremental encoders and converts these codes into angular displacements. Data acquisition card receives control signal from the controller application and passes this signal into the amplifier. Therefore, we define controller application, state recorder, timer, feedback controller, I/O API, A/D counter card, and D/A output card as blocks. Fig. 6 shows a block definition diagram which represents building blocks of our designed inverted pendulum system. The stereotype <<software>> and <<hardware>> are used to indicate software blocks and hardware blocks.

![Fig. 6 Building blocks of inverted pendulum system]
3.3 Internal Structures

After the corresponding blocks of the inverted pendulum system are defined, then the relationships between each part of the system are defined by using the internal block diagram.

The inverted pendulum system has been designed to use the electrical power from the external power supply and the user can operate the system as described in the use case diagram. At this point, we can create the internal block diagram of the system context as shown in Fig. 7. The external power supply provides the electric current to the inverted pendulum system and the user and the nuisance can interface to the system.

![Fig. 7 Outline of system context](image)

A PC-power supply is designed to use as the power source of the inverted pendulum system. It receives electric current from an external power supply, which is AC, and transforms it into DC.

The inverted pendulum system consists of two parts which are the pendulum assembly and the computer. Inside the pendulum assembly block, the pendulum is connected to the cart via the revolute joint. The cart is moved by the tendon mechanism which is also driven by the motor. The motor receives power from the amplifier which is connected to the power supply and receives a control signal from the computer. Two encoders, whose type is incremental, are connected to the counter card of computer. One encoder reads the angular displacement of the motor’s rotor, which can be transformed into the position of the cart $x$. Another one is attached to one end of the pendulum rod that connects to the cart. This encoder reads the angle of the pendulum $\theta$, which is measured with respect to the vertical upright. These encoders receive power from the power supply and produce gray coding as output.

For the computer block, the A/D counter card receives gray coding from encoders and passes these codes to the controller application to calculate the output control signal. Then, the output control signal is sent to the pendulum assembly via the D/A output card.

The controller application uses the I/O API to communicate with the A/D counter card and the D/A output card. It reads the number of pulses from the A/D counter card and sends to the feedback controller module. The feedback controller uses these data and time duration from the timer to calculate for the control signal and sends this signal to the I/O API. Then, the I/O API passes this control signal to the D/A output card. The state recorder is used to record states of our inverted system. It receives state values from the feedback controller and writes them to a file.

The outline of the connection between parts inside our inverted pendulum system is shown as Fig. 8.
Fig. 8 Outline of connections between parts inside inverted pendulum system
3.4 Value Types

After we define blocks and design the internal structure of the inverted pendulum system, the ValueTypes, which are used with the inverted pendulum system, have to be created.

SysML provides a ValueType to define type of values that may be used to express information about a system, but it cannot be identified as a target of any reference [1]. It can be used to type properties, operation parameters, and potentially other elements of a system. A ValueType has an ability to carry a unit and dimension associated with a value. For our inverted pendulum, the ValueTypes and its corresponding unit and dimension are defined as shown in Fig. 9.

3.5 Standard Ports and Flowports

To clarify defined interfaces between parts, flow ports and standard ports are provided by SysML. From the SysML specification, a flow port specifies input and output items that flow between a block and its environment. Flow ports are interaction points through which data, material or energy can enter or leave the owning block. Therefore, we use this kind of port for passing data or energy between parts in our inverted pendulum system.

A standard port specifies the services that the owning block provides to its environment as well as the services that the owning block expects of its environment [1]. The standard ports are used for the relation between actors and the system, and between hardware and software.

The inverted pendulum system provides the two standard ports to communicate with the user. One is realized as iUserCommand interface. It contains operations that the user acts upon the system including start, stop, and set controller gain. Another is realized as iAmplifier interface. It contains the operation for adjust an amplifier gain. For nuisance, the system provides a standard port that is realized as iDisturbance interface that nuisance can disturb the system with its operation. Fig. 10 shows standard ports which are provided by the inverted pendulum block.
There are two relations between hardware and software. The first relation is a relation between the A/D counter card and the I/O API. The A/D counter card provides a standard port that is typed by the CounterAcquire block. This block is realized as iCounterAcquire interface. It contains a flow port to represent the flow between the A/D counter card and the I/O API. The I/O API can receive the number of pulses of encoders from this interface through these flow ports. Another relation is a relation between the D/A output card and the I/O API. The D/A output card provides a standard port that is typed by the OutputAcquire block. This block is realized as iOutputAcquire interface and also provides a flow port to represent the flow between the D/A output card and the I/O API. Fig. 11 shows standard ports which are used to connect between hardware part and software part.

After the flow ports and standard ports of the inverted pendulum system are defined, then modification to the internal structure needs to be done. Fig. 12 shows the modified internal block diagram of the system context from Fig. 7.

From the inverted pendulum internal block diagram in Fig. 8, it is also modified by adding connections between parts in the system with ports as show in Fig. 13. For relations between parts inside a computer part, they are separated into a new internal block diagram as show in Fig. 14.
Fig. 13 Internal block diagram of inverted pendulum block

Fig. 14 Internal block diagram of computer block
3.6 Parametric

SysML provides a parametric diagram to help design the calculation process of the controller. Value types and constraints must be defined before designing the feedback controller in a parametric diagram.

The feedback controller calculates for input vector, which is used to control the pendulum upright by using the feedback equation (2.3). This equation needs four state variables of $x$. Only two state variables, which are the cart’s displacement $x$ and pendulum angular displacement $\theta$, can be obtained from the encoder. The other two variables, which are cart’s velocity and the pendulum angular velocity, have to be estimated by using the variation of displacement with respect to time. The equations for estimation of the cart’s velocity and the pendulum’s angular velocity are defined as constraint blocks, whose names are VelocityEqn and AngVelocityEqn respectively. The TimeDifferentEqn constraint block is defined for calculate time different. This value is used by the VelocityEqn constraint block and the AngVelocityEqn constraint block.

The position of the cart is measured from the angular displacement of the motor. The angular displacements, which are obtained from the counter card, are the number of encoder generated pulses (gray coding). These pulses are converted into angular displacement of the encoders and the angular displacement of the motor is also converted into linear displacement of the cart. Therefore, the equations of these conversions are defined as constraint blocks. The Count2Rad constraint block is defined for conversion of gray coding into angular displacement and the LinearDisplacementEqn constraint block is defined for conversion of the motor’s angular displacement into the cart’s displacement.

The FeedbackGainEqn constraint block is created for calculation of control signal as feedback equation (2.3). It receives the controller gain $K$ and the state vector of the system. The controller gain $K$ is an attribute of the FeedbackController block and the state vector is a vector which contains all state variables of our inverted pendulum system. It is produced by the Multiplex constraint block. The output voltage from the FeedbackGainEqn is sent to the Vol2Signal constraint block. The Vol2Signal constraint block converts the output voltage to the signal and then sends this signal to the data acquisition card.

Fig. 15 shows the defined constraint blocks and the parametric diagram of feedback controller can be created as shown in Fig. 16.
4. Simulation

Before bringing the designed system into the manufacturing process, the system engineer have to be sure that each designed part can operate together correctly and the controller can control the pendulum upright with the designed controller gain because the manufacturing cost is very high. Then, the simulations are needed.

In this work, a simulation of the designed pendulum system is created using java as shown in Fig. 17. The I/O API block, which is a software part of the system, is extended to the SimIoApi class. The SimIoApi class consists of various classes which correspond to the designed inverted pendulum’s blocks.

![Simulation Diagram](image1)

**Fig. 16 Parametric diagram of feedback controller**

**Fig. 17 The inverted pendulum system simulation**
By using the reverse engineering module of MagicDraw, the reverted classes are compared with the designed SysML blocks as shown in Fig. 18.

The values of the cart’s displacement and pendulum’s angular displacement are simulated by using the control signal and the state equation of the inverted pendulum system. The simulated value of the cart’s displacement and the pendulum’s angular displacement are sent to the feedback controller. Then, the feedback controller calculates for the control signal and sends it back to the SimIoApi.

The plant of the inverted pendulum system in Fig. 2 can be replaced by the state equation (See Appendix A). By assuming that the length of rod is \( l = 1.0 \text{ m.} \), mass of pendulum ball is \( m = 0.1 \text{ kg.} \), mass of the cart is \( M = 1.0 \text{ kg.} \), and the initial pendulum’s angle is \( 10^\circ \). The fourth order Runge-Kutta is used to solve the state equation and the results of simulation show that the designed controller can control the pendulum to remain upright \( (\theta = 0) \). Fig. 19 and Fig. 20 show states of the inverted pendulum system which is controlled by using the controller gain \( K = [-0.045, -0.6275, -0.25, -0.6] \) and \( K = [-0.045, -0.6275, -0.125, -0.3] \) respectively with the same amplifier gain 2.0.
Fig. 19 State values of simulated inverted pendulum system when using $K=[-0.045, -0.675, -0.25, -0.6]$ and amplifier gain = 2.0

Fig. 20 State values of simulated inverted pendulum system when using $K=[-0.045, -0.675, -0.125, -0.3]$ and amplifier gain = 2.0
5. Conclusions

This paper shows that the graphical SysML system modeling language and the MagicDraw SysML plug-in can help us to design the inverted pendulum system. However, a limitation of the current MagicDraw SysML plug-in is that the tool does not yet provide active simulation. There are many reasons why making the simulation is very important for designing the system including safety and manufacturing cost.

About the comparison between reverted classes and designed SysML blocks, system engineer and software engineer can use MagicDraw SysML plug-in to verify the implemented model by reversing it to reverted classes. Then the implemented model can be compared with a designed SysML model.

From this work, SysML is very useful for the system engineer to handle the complexity of a system and to share understanding of the system including requirements and design.

Appendix

A. State Equation of Inverted Pendulum System

From Fig.1 and by assuming that the system has no friction then the state equation can be obtained by using Lagrange’s equation with the Lagrangian $q$ [4, 5]:

$$q(x, \theta) = \frac{1}{2} (M + m) \dot{x}^2 + m \dot{x} \dot{\theta} \cos \theta + \frac{1}{2} ml^2 \dot{\theta}^2 - mg l \cos \theta$$

(A.1)

From Lagrange’s equation and assuming that angle $\theta$ is small, we obtain the equations of motion as:

$$(M + m) \ddot{x} + ml \dddot{\theta} = \tau(t) \cdot \dot{r}^{-1}$$

(A.2)

$$ml \ddot{x} + ml^2 \dddot{\theta} - mg l \dot{\theta} = 0$$

(A.3)

The constant $\dot{r}$ can be considered as an effective moment arm. It is used to convert the torque $\tau(t)$ into horizontal force that is applied to the cart. Because the torque is created by a motor, it is proportional to the electric current that passes through the motor’s armature circuit, therefore we can control the system by using electric current $i(t)$ instead of torque $\tau(t)$. Form equation (A.2), $\tau(t) \cdot \dot{r}^{-1}$ can be replaced by $\varepsilon \cdot i(t)$, where $\varepsilon$ is an arbitrary constant.

Let $\mathbf{x} = [x \, \ddot{x} \, \theta \, \dot{\theta}]^T$, and $\mathbf{u} = [\varepsilon \cdot i(t)]$

Then: $\dot{\mathbf{x}} = [\dddot{x} \, \ddot{x} \, \ddot{\theta} \, \dot{\theta}]^T$

From state equation:

$$\dot{\mathbf{x}} = A \mathbf{x} + B \mathbf{u}$$

we get [4, 5]:

$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g(M + m)}{Ml} & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} \mathbf{u}$$

(A.4)

The state vector $\mathbf{x}$ can be obtained by using a numerical method, such as Taylor’s series and Runge-Kutta [6], to solve this equation with the different input vector $\mathbf{u}$. Therefore, this equation can be used to simulate the dynamics of an inverted pendulum system.
References


Services Contacts

Please contact us at services@nomagic.com for getting more detailed information about No Magic services, training courses, scheduling possibilities, or if you have any specific needs for your company.

No Magic, Inc.
Texas
7304 Alma Drive, Suite 600
Plano, TX 75025
Phone: +1 214 291 9100
Fax: +1 214 291 9099
www.nomagic.com
www.magicdraw.com

No Magic European Sales Office
UAB “Baltijos programine iranga”
Savanoriu pr. 363
LT-49425, Kaunas, Lithuania
Phone: +370 37 324032
Fax: +370 37 320670
http://www.bpi.lt/en

No Magic Asia, Inc.
719 KPN Tower, 22nd Floor,
Rama IX Road, Bangkapi, Huaykwang
Bangkok 10310, Thailand
Phone: +66-2-7170250
Fax: +66-2-7170251
http://www.nomagicasia.com