

Modeling and Control of Mechanical Systems in Simulink of Matlab

Leghmizi Said and Boumediene Latifa
*College of Automation, Harbin Engineering University
China*

1. Introduction

Mechanical systems are types of physical systems. This is why it is important to study and control them using information about their structure to describe their particular nature. Dynamics of Multi-Body Systems (MBS) refers to properties of the mechanical systems. They are often described by the second-order nonlinear equations parameterized by a configuration-dependent inertia matrix and the nonlinear vector containing the Coriolis and centrifugal terms. These equations are the cornerstone for simulation and control of these systems, and then many researchers have attempted to develop efficient modeling techniques to derive the equations of motion of multi-body systems in novel forms. Furthermore, to prove the efficiency of these models and simulate them, efficient software for modeling is needed.

In the last few years, Simulink has become the most widely used software package in academia and industry for modeling and simulating mechanical systems. Used heavily in industry, it is credited with reducing the development of most control system projects. Simulink (**Simulation and Link**) is an extension of MATLAB by Mathworks Inc. It works with MATLAB to offer modeling, simulation, and analysis of mechanical systems under a graphical user interface (GUI) environment. It supports linear and nonlinear systems, modelled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates. It allows engineers to rapidly and accurately build computer models of mechanical systems using block diagram notation. It also includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors. Moreover it can allow the users to customize and create their own blocks.

Using Simulink we can easily build models from presentative schemes, or take an existing model and add to it. Simulations are interactive, so we can change parameters “on the fly” and immediately see the results. As Simulink is an integral part of MATLAB, it is easy to switch back and forth during the analysis process and thus, the user may take full advantage of features offered in both environments. So we can take the results from Simulink and analyze them in Matlab workspace.

In this chapter we present the basic features of Simulink focusing on modeling and control of mechanical systems. In the first part, we present the method for creating new Simulink models using different toolboxes to customize their appearance and use. Then in the second

part, we discuss Simulink and MATLAB features useful for viewing and analyzing simulation results. In the third part, we present different types of modeling of mechanical systems used in Simulink. Finally, we give two examples of modeling and control, illustrating the methods presented in the previous parts. The first example describes the Stewart platform and the second one describes a three Degree of Freedom (3-Dof) stabilized platform.

2. Getting started with Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamical systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, i.e., have different parts that are sampled or updated at different rates (Parlos, 2001).

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this interface, we can draw the models just as we would with pencil and paper (or depict them as it is done in most textbooks). Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components, and connectors. We can also customize and create our own blocks.

Models are hierarchical. This approach provides an insight how a model is organized and how its parts interact. After we define a model, we can simulate it, using a choice of different methods, either from the Simulink menus or by entering commands in MATLAB's command window. The menus are particularly convenient for interactive work, while the command-line approach is very useful for running a batch of simulations (for example, if we are doing Monte Carlo simulations or want to sweep a parameter across a range of values). Using scopes and other display blocks, we can see the simulation results while the simulation is running. In addition, we can change parameters and immediately see what happens, for "what if" exploration. The simulation results can be put in the MATLAB workspace for post processing and visualization. And because MATLAB and Simulink are integrated, we can simulate, analyze, and revise our models in either environment at any point (Parlos, 2001).

2.1 Starting Simulink

To start a Simulink session, we'd need to bring up Matlab program first (Nguyen, 1995).

From Matlab command window, enter:

```
>> simulink
```

Alternately, we may click on the Simulink icon located on the toolbar as shown:



Fig. 1. Simulink icon in Matlab window

Simulink's library browser window like one shown below will pop up presenting the block set for model construction.

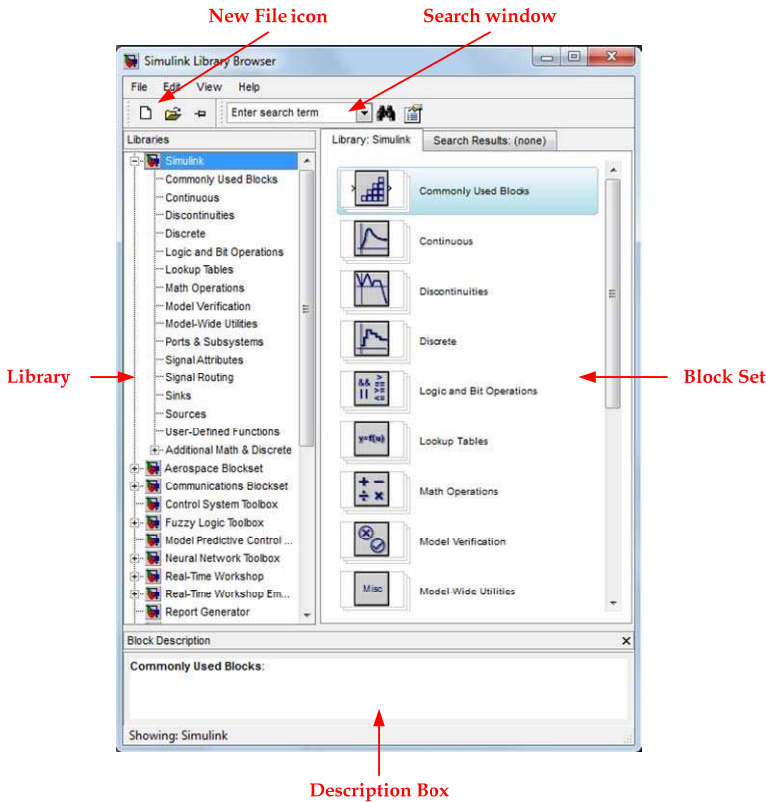


Fig. 2. Simulink’s library browser

To see the content of the blockset, click on the "+" sign at the beginning of each toolbox.

To start a model click on the NEW FILE ICON as shown in the screenshot above.

Alternately, we may use keystrokes CTRL+N. A new window will appear on the screen. We will be constructing our model in this window. Also in this window the constructed model is simulated. A screenshot of a typical working (model) window looks like one shown below:

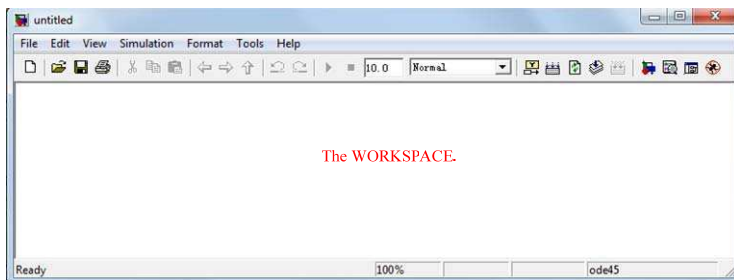


Fig. 3. Simulink workspace

To be more familiarized with the structure and the environment of Simulink, we are encouraged to explore the toolboxes and scan their contents. We may not know what they are all about but perhaps we could catch on the organization of these toolboxes according to the category. For an instant, we may see Control System Toolbox to consist of the Linear Time Invariant (LTI) system library and the MATLAB functions can be found under Function and Tables of the Simulink main toolbox. A good way to learn Simulink (or any computer program in general) is to practice and explore it. Making mistakes is a part of the learning curve. So, fear not, we should be (Nguyen, 1995).

A simple model is used here to introduce some basic features of Simulink. Please follow the steps below to construct a simple model.

Step 1. Creating Blocks.

From BLOCK SET CATEGORIES section of the SIMULINK LIBRARY BROWSER window, click on the "+" sign next to the Simulink group to expand the tree and select (click on) Sources.

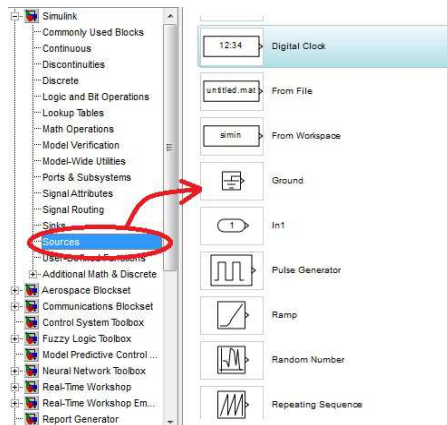


Fig. 4. Sources Block sets

A set of blocks will appear in the BLOCKSET group. Click on the Sine Wave block and drag it to the workspace window (also known as model window).

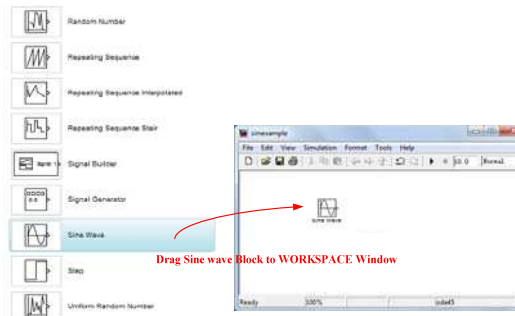



Fig. 5. Adding Blocks to Workspace

Now we have established a source of our model.

To save a model, click on the floppy diskette icon  or from FILE menu, select Save or CTRL+S. All Simulink model files will have an extension ".mdl". Simulink recognizes the file with .mdl extension as a simulation model (similar to how MATLAB recognizes files with the extension .m as an MFile).

Continue to build the model by adding more components (or blocks) to the model window. We will add the Scope block from Sinks library, an Integrator block from Continuous library, and a Mux block from Signal Routing library.

NOTE: If we wish to locate a block knowing its name, we may enter the name in the SEARCH WINDOW (at Find prompt) and Simulink will bring up the specified block.

To move the blocks around, click on them and drag to a desired location.

Once all the blocks are dragged over to the work space, we may remove (delete) a block, by clicking on it once to turn on the "select mode" (with four corner boxes) and use the DEL key or keys combination CTRL-X.

Step 2. Making connections.

To establish connections between the blocks, move the cursor to the output port represented by ">" sign on the block. Once placed at a port, the cursor will turn into a cross "+" enabling us to make connection between blocks.

To make a connection: left-click while holding down the control key (on the keyboard) and drag from source port to a destination port.

The connected model is shown below.

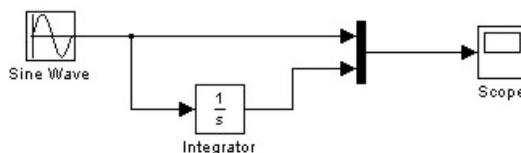


Fig. 6. Block diagram for Sine simulation

A sine signal is generated by the Sine Wave block (a source) and displayed on the scope (fig. 7). The integrated sine signal is sent towards the scope, to display it along with the original signal from the source via the Mux, whose function is to multiplex signals in form of scalar, vector, or matrix into a bus.

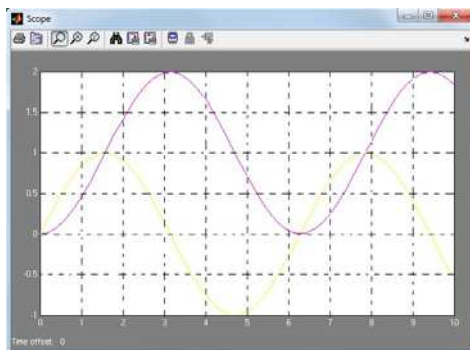


Fig. 7. Scope appearance

Step 3. Running simulation.

Now the simulation of the simple system above can be run by clicking on the play button (▶), alternatively, we may use key sequence CTRL+T, or choose Start submenu under Simulation menu).

Double click on the Scope block to display of the scope.

To view/edit the parameters, simply double click on the block of interest.

2.2 Handling of blocks and lines

The table below describes the actions and the corresponding keystrokes or mouse operations (Windows versions) (Nguyen, 1995).

Actions	Keystrokes or Mouse Actions
Copying a block from a library	Drag the block to the model window with the left mouse button on the OR use choose between select the COPY and PASTE from EDIT menu.
Duplicating blocks in a model	Hold down the CTRL key and select the block. Drag the block to a new location with the left mouse button.
Display block's parameters	Click doubly on the bloc.
Flip a block	CTRL-F
Rotate a block	CTRL-R
Changing blocks' names	Click on block's label and position the cursor to desired place.
Disconnecting a block	Hold down the SHIFT key and drag the block to a new location.
Drawing a diagonal line	Hold down the SHIFT key while dragging the mouse with the left button.
Dividing a line	Move the cursor to the line to where we want to create the vertex and use the left button on the mouse to drag the line while holding down the SHIFT key.

Table 1. The actions and the corresponding keystrokes or mouse operations.

2.3 Simulink block libraries

Simulink organizes its blocks into block libraries according to their behaviour.

The **Simulink** window displays the block library icons and names:

- The Sources library contains blocks that generate signals.
- The Sinks library contains blocks that display or write block output.
- The Discrete library contains blocks that describe discrete-time components.
- The Linear library contains blocks that describe linear functions.
- The Nonlinear library contains blocks that describe nonlinear functions.
- The Connections library contains blocks that allow multiplexing and demultiplexing, implement external Input/Output, pass data to other parts of the model, create subsystems, and perform other functions.
- The Blocksets and Toolbox library contains the Extras block library of specialized blocks.
- The Demos library contains useful MATLAB and Simulink demos.

3. Viewing and analyzing simulation results

Output trajectories from Simulink can be plotted using one of three methods (The MathWorks, 1999):

- Feeding a signal into either a Scope or an XY Graph block
- Writing output to return variables and using MATLAB plotting commands
- Writing output to the workspace using To Workspace blocks and plotting the results using MATLAB plotting commands

3.1 Using the scope block

We can use display output trajectories on a Scope block during a simulation.

This simple model shows an example of the use of the Scope block:

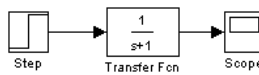


Fig. 8. Block diagram for Scope displaying

The display on the Scope shows the output trajectory. The Scope block enables to zoom in on an area of interest or save the data to the workspace.

The XY Graph block enables to plot one signal against another.

These blocks are described in Chapter 9.

3.2 Using return variables

By returning time and output histories, we can use MATLAB plotting commands to display and annotate the output trajectories.

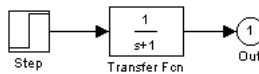


Fig. 9. Block diagram for output displaying

The block labelled **Out** is an Outport block from the Connections library. The output trajectory, *yout*, is returned by the integration solver. For more information, see Chapter 4.

This simulation can also be run from the Simulation menu by specifying variables for the time, output, and states on the Workspace I/O page of the Simulation Parameters dialog box. then these results can be plot using:

```
plot (tout,yout)
```

3.3 Using the To Workspace block

The **To Workspace** block can be used to return output trajectories to the MATLAB workspace. The model below illustrates this use:

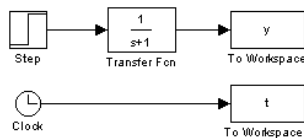


Fig. 10. Block diagram for Workspace displaying

The variables y and t appear in the workspace when the simulation is complete. The time vector is stored by feeding a Clock block into a To Workspace block. The time vector can also be acquired by entering a variable name for the time on the Workspace I/O page of the Simulation Parameters dialog box for menu-driven simulations, or by returning it using the `sim` command (see Chapter 4 for more information).

The **To Workspace** block can accept a vector input, with each input element's trajectory stored as a column vector in the resulting workspace variable.

4. Modeling mechanical systems with Simulink

Simulink's primary design goal is to enable the modeling, analysis, and implementation of dynamics systems so then mechanical systems. The mechanical systems consist of bodies, joints, and force elements like springs. Modeling a mechanical system need the equations of motion or the mechanical structure. Thus in general mechanical systems can be simulated by two ways:

- Using graphical representation of the mathematical model.
- Drawing directly the mechanical system using SimMechanics.

4.1 Modeling using graphical representation:

The equations of motion of mechanical systems have undergone historical development associated with such distinguished mathematicians as Newton, D'Alembert, Euler, Lagrange, Gauss, and Hamilton, among others (Wood & Kennedy, 2003). While all made significant contributions to the formulation's development of the underlying equations of motion, our interest here is on the computational aspects of mechanical simulation in an existing dynamic simulation package. Simulink is designed to model systems governed by these mathematical equations. The Simulink model is a graphical representation of mathematical operations and algorithm elements. Simulink solves the differential equation by evaluating the individual blocks according to the sorted order to compute derivatives for the states. The solver uses numeric integration to compute the evolution of states through time. Application of this method is illustrated in the first example of the section 5.

4.2 Modeling using SimMechanics

SimMechanics™ software is a block diagram modeling environment for the engineering design and simulation of rigid body machines and their motions, using the standard Newtonian dynamics of forces and torques. Instead of representing a mathematical model of the system, we develop a representation that describes the key components of the mechanical system. The base units in SimMechanics are physical elements instead of algorithm elements. To build a SimMechanics model, we must break down the mechanical system into the building blocks that describe it (Popinchalk, 2009).

After building the mechanical representation using SimMechanics, to study the system's response to and stability against external changes, we can apply small perturbations in the motion or the forces/torques to a known trajectory and force/torque set. SimMechanics software and Simulink® provide analysis modes and functions for analyzing the results of perturbing mechanical motion. To use these modes, we must first build a kinematic model of the system, one that specifies completely the positions, velocities, and accelerations of the system's bodies. We create a kinematic model by interconnecting blocks representing the

bodies and joints of the system and then connecting actuators to the joints to specify the motions of the bodies. Application of this method is illustrated in the second example of the section 5.

5. Examples of modeling and control of mechanical systems

5.1 Dynamics modeling for satellite antenna dish stabilized platform

The stabilized platform is the object which can isolate motion of the vehicle, and can measure the change of platform’s motion and position incessantly, exactly hold the motorial gesture benchmark, so that it can make the equipment which is fixed on the platform aim at and track object fastly and exactly. In the stabilized platform systems, the basic requirements are to maintain stable operation even when there are changes in the system dynamics and to have very good disturbance rejection capability.

The objective of this example is to develop the dynamics model simulation for satellite antenna dish stabilized platform. The dynamic model of the platform is a three degree of freedom system. It is composed of, the four bodies which are: case, outer gimbal, inner gimbal and platform as shown in fig. 11. Simulink is used to simulate the obtained dynamic model of the stabilized platform. The testing results can be used to analyze the dynamic structure of the considered system. In addition, these results can be applied to the stabilization controller design study (Leghmizi et al., 2011).

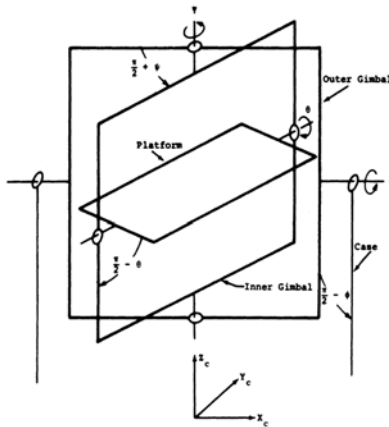


Fig. 11. The system structure

The mathematical modeling was established using Euler theory. The Euler’s moment equations are

$$\bar{M} = i\bar{H} \tag{1}$$

The net torque \bar{M} consists of driving torque applied by the adjacent outer member and reaction torque applied by the adjacent inner member.

$$i\bar{H} = \frac{dH}{dt} = m\bar{H} + \bar{\omega}_m \times \bar{H} \tag{2}$$

$i\bar{H}$: Inertial derivative of the vector \bar{H} ;

$m\bar{H}$: Derivative of H calculated in a rotating frame of reference;

$\bar{\omega}_m$: Absolute rotational rate of the moving reference frame;

\bar{H} : Inertial angular momentum;

\bar{M} : External torque applied to the body.

By applying equation (2) on the different parts of the platform system , the system may be expressed as a set of second-order differential equations in the state variables. Solving this system of equations we obtain:

$$\ddot{\phi} = \frac{C_i B_o - C_o B_i}{A_i B_o - A_o B_i} \quad (3)$$

$$\ddot{\psi} = \frac{C_o A_i - C_i A_o}{A_i B_o - A_o B_i} \quad (4)$$

$$\ddot{\theta} = \frac{C_p}{B_p} - \frac{A_p}{B_p} * \frac{C_i B_o - C_o B_i}{A_i B_o - A_o B_i} \quad (5)$$

Where

$$A_p = \sin \psi$$

$$B_p = 1$$

$$C_p = \frac{M_{ipy}^* - MPY}{I_{py}}$$

$$A_i = \cos \psi \cos \theta \sin \theta \left[\frac{I_{px} - I_{pz}}{I_{iz}} \right]$$

$$B_i = \left[1 + \sin^2 \theta \frac{I_{px}}{I_{iz}} + \cos^2 \theta \frac{I_{pz}}{I_{iz}} \right]$$

$$C_i = \frac{M_{oiz}^* - MIZ}{I_{iz}}$$

$$A_o = 1 + \cos^2 \psi \left[\frac{I_{ix} + I_{px} \cos^2 \theta + I_{pz} \sin^2 \theta}{I_{ox}} \right] + \sin^2 \psi \left[\frac{I_{iy}}{I_{ox}} \right]$$

$$B_o = \cos \theta \sin \theta \cos \psi \left[\frac{I_{px} - I_{pz}}{I_{ox}} \right]$$

$$C_o = \frac{M_{cox}^* - MCX}{I_{ox}}$$

Detailed equations computation is presented in the paper (Leghmizi, 2010, 2011).

Here, it suffices to note that designing a simulation for the system based on these complete nonlinear dynamics is extremely difficult. It is thus necessary to reduce the complexity of the problem by considering the linearized dynamics (Lee et al., 1996). This can be done by noting that the gimbal angles variations are effectively negligible and that the ship velocities

effect is insignificant. Applying the above assumptions to the nonlinear dynamics, the following equations are obtained.

$$\ddot{\phi} = \frac{D_{co}}{I_{px} + I_{ix} + I_{ox}} \dot{\phi} - \frac{1}{I_{px} + I_{ix} + I_{ox}} F_{co}(\text{sgn} \dot{\phi}) - \frac{I_{pz} - I_{py} + I_{px}}{I_{px} + I_{ix} + I_{ox}} \dot{\psi} \dot{\theta} - T_{oo} \tag{6}$$

$$\ddot{\psi} = \frac{D_{oi}}{I_{pz} + I_{iz}} \dot{\psi} - \frac{1}{I_{pz} + I_{iz}} F_{oi}(\text{sgn} \dot{\psi}) - \frac{I_{py} - I_{px} + I_{pz}}{I_{pz} + I_{iz}} \dot{\theta} \dot{\phi} - T_{im} \tag{7}$$

$$\ddot{\theta} = \frac{D_{ip}}{I_{py}} \dot{\theta} - \frac{1}{I_{py}} F_{ip}(\text{sgn} \dot{\theta}) - \frac{I_{px} - I_{pz} + I_{py}}{I_{py}} \dot{\psi} \dot{\phi} - T_{ll} \tag{8}$$

5.1.2 Modeling the equations of motion with Simulink

The model in fig. 12 is the graphical representation of equations (6), (7) and (8). It's obtained by using the **Simulink toolbox**.

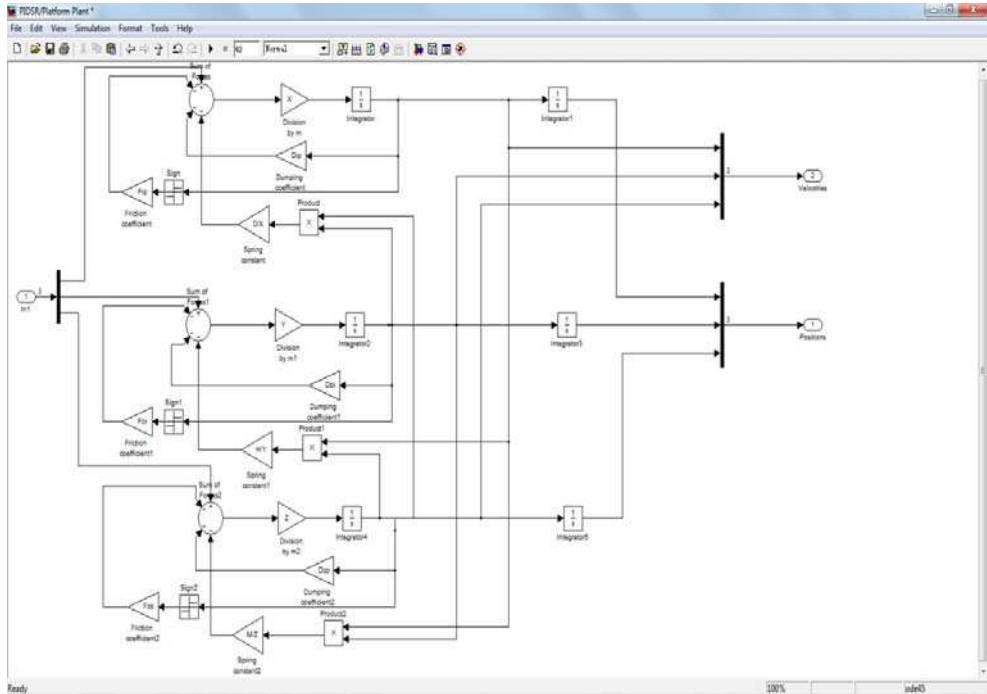


Fig. 12. The platform plant simulation

In order to enhance our understanding of the system, we performed a simulation in closed-loop mode. After that, a PID controller was applied to the closed-loop model. The PID controlled parameters was calculated using the Ziegler–Nichols method (Moradi, 2003). The obtained Simulink model is presented in the fig. 13.

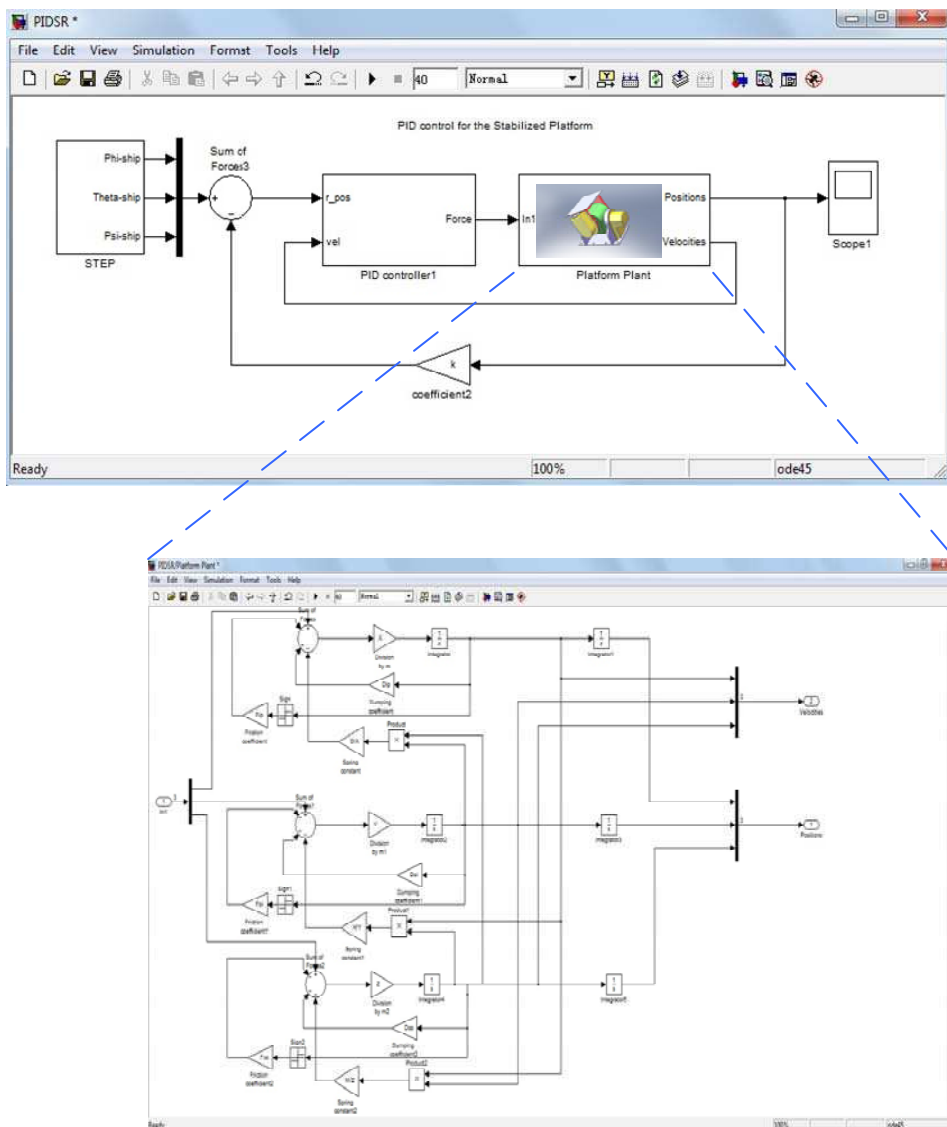


Fig. 13. Simulation model by Simulink

This simulation was particularly useful to recognize the contribution of each modelled effect to the dynamics of the system. Also, knowing the natural behavior of the system could be useful for establishing adapted control laws. Simulation results will be presented to illustrate the gimbals behaviour to different entries. They are presented in fig. 14, which contains the impulsion and step responses of the closed-loop system using the PID controller. Each graph superimposes the angular position on the X axes (blue), the Y axes (green) and the Z axes (red).

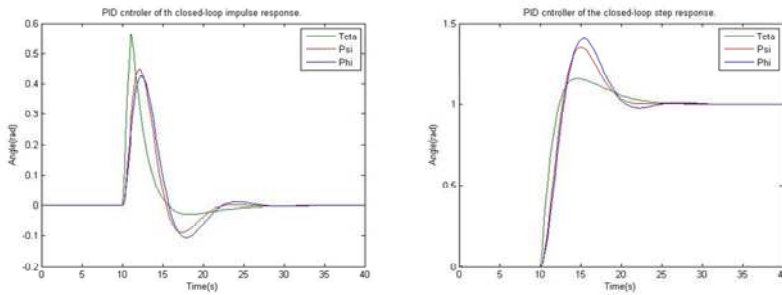


Fig. 14. The closed-loop system impulsion and step responses using the PID controller

5.2 Modeling a Stewart platform

The Stewart platform is a classic design for position and motion control, originally proposed in 1965 as a flight simulator, and still commonly used for that purpose (Stewart, 1965). Since then, a wide range of applications have benefited from the Stewart platform. A few of the industries using this design include aerospace, automotive, nautical, and machine tool technology. Among other tasks, the platform has been used, to simulate flight, model a lunar rover, build bridges, aid in vehicle maintenance, design crane and hoist mechanisms, and position satellite communication dishes and telescopes (Matlab Help).

The Stewart platform has an exceptional range of motion and can be accurately and easily positioned and oriented. The platform provides a large amount of rigidity, or stiffness, for a given structural mass, and thus provides significant positional certainty. The platform model is moderately complex, with a large number of mechanical constraints that require a robust simulation. Most Stewart platform variants have six linearly actuated legs with varying combinations of leg-platform connections. The full assembly is a parallel mechanism consisting of a rigid body top or mobile plate connected to an immobile base plate and defined by at least three stationary points on the grounded base connected to the legs.

The Stewart platform used here is connected to the base plate at six points by universal joints as shown in fig. 15. Each leg has two parts, an upper and a lower, connected by a cylindrical joint. Each upper leg is connected to the top plate by another universal joint. Thus the platform has $6 \times 2 + 1 = 13$ mobile parts and $6 \times 3 = 18$ joints connecting the parts.

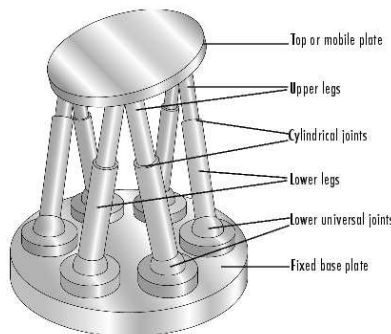


Fig. 15. Stewart platform

5.2.1 Modeling the physical Plant with SimMechanics

The Plant subsystem models the Stewart platform's moving parts, the legs and top plate. The model in the fig. 16 is obtained by using the SimMechanics toolbox. From the Matlab demos we can open this subsystem.

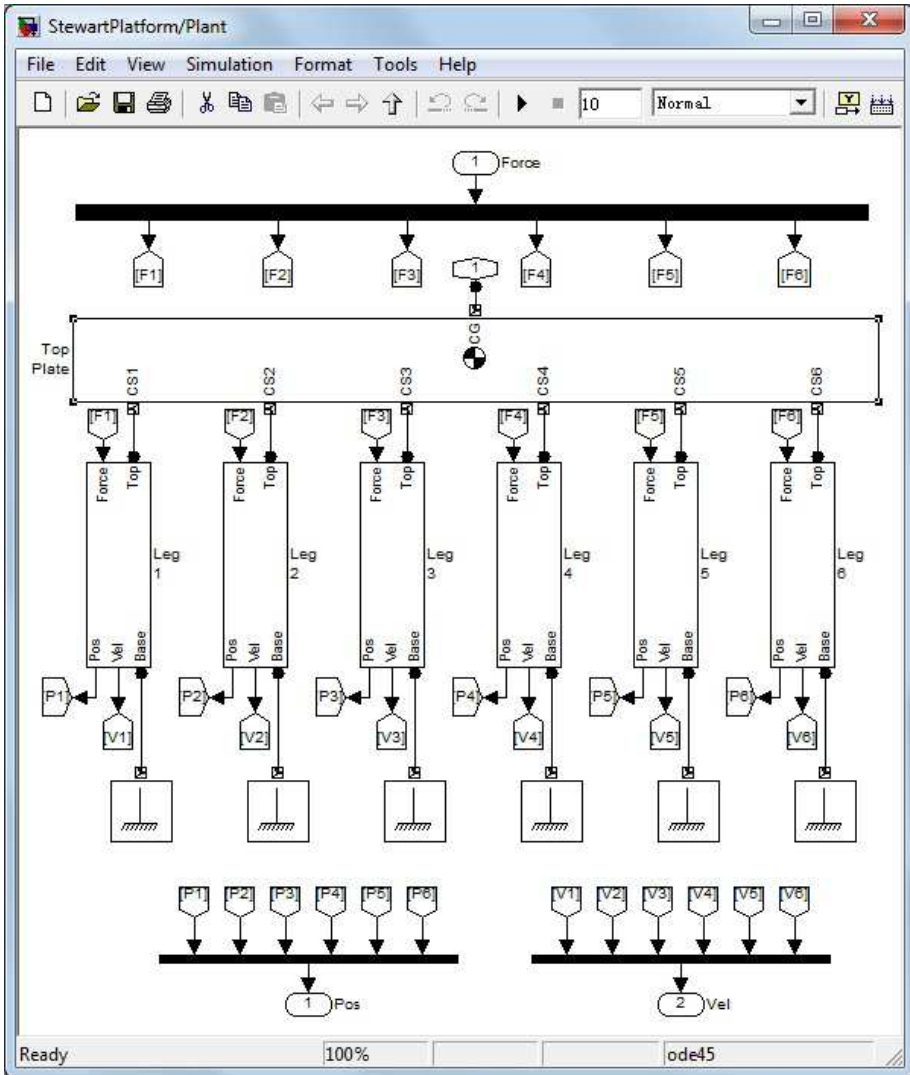


Fig. 16. Stewart platform plant representation with SimMechanics

The entire Stewart platform plant model is contained in a subsystem called **Plant**. This subsystem itself contains the base plate (the ground), the Top plate and the six platform legs. Each of the legs is a subsystem containing the individual Body and Joint blocks that make up the whole leg (see fig. 17).

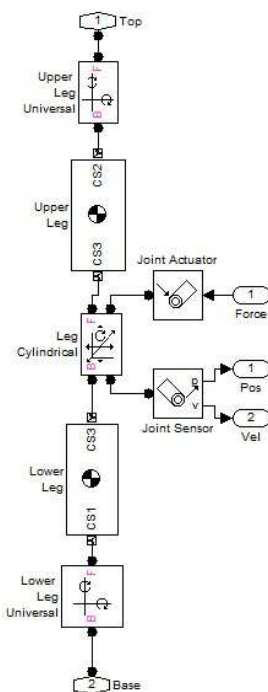


Fig. 17. Leg Subsystem content

To visualise the content of this subsystem, select one of the leg subsystems and right-click select **Look Under Mask**.

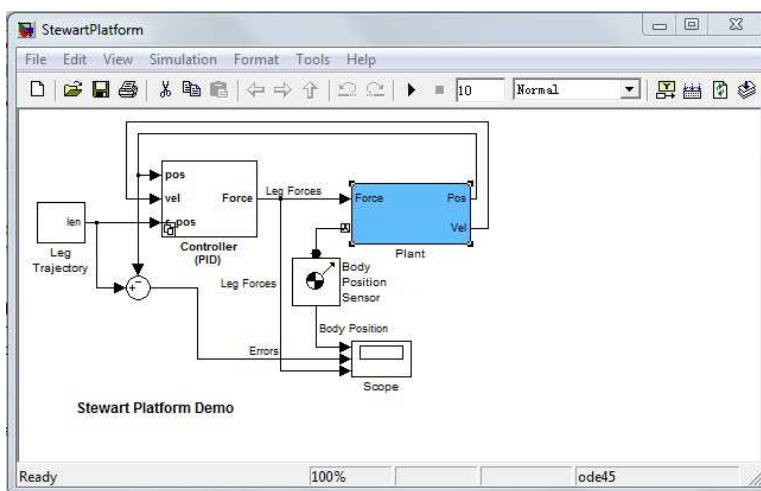


Fig. 18. Stewart Platform Control Design Model

The blue subsystem contains the Stewart platform plant presented in fig. 18. The simulation model in fig. 18 is the control of the Stewart platform's motion with the linear proportional-integral-derivative (PID) feedback system presented in fig. 19.

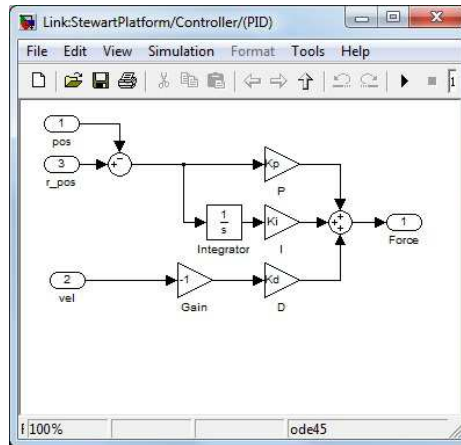


Fig. 19. Stewart Platform PID Controller Subsystem

The control transfer function of the PID linear feedback control system has the form $K_i/s + K_d \cdot s + K_p$. The control gains K_i , K_p , and K_d in their respective blocks refer to the variables K_i , K_p , K_d defined in the workspace. Check their initialized values:

$$K_i = 10000$$

$$K_p = 2000000$$

$$K_d = 45000$$

To simulate the Stewart platform with the PID controller:

- Open the Scope and start the simulation.
- Observe the controlled Stewart platform motion. The Scope results given in fig. 20 show how the platform initially does not follow the reference trajectory, which starts in a position different from the platform's home configuration. The motion errors and forces on the legs are significant. Observe also that the leg forces saturate during the initial transient.

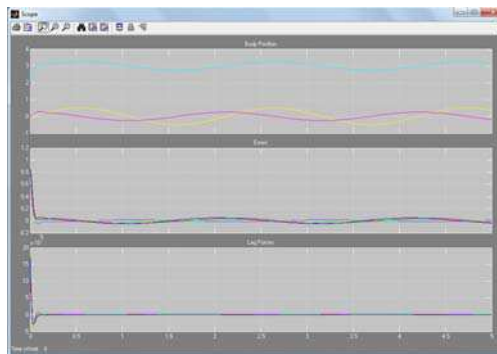


Fig. 20. Simulation results

The platform moves quickly to synchronize with the reference trajectory, and the leg forces and motion errors become much smaller.

6. Conclusion

The modeling of mechanical systems requires a language capable to describe physical phenomena in multiple energy domains, in continuous time or discrete time. Recent advances in modeling have resulted in several languages satisfying these requirements. Simulink of Matlab is one of such languages. Simulink is a software package that enables to model, simulate, and analyze dynamic systems, i.e., the systems with outputs and states changing with time. Simulating a mechanical system is a two-step process with Simulink involved. First, we create a graphical model of the system to be simulated, using the Simulink model editor. Then, we use Simulink to simulate the behavior of the system over a specified time span.

In this Chapter, using Simulink of Matlab, two examples of modeling and simulation were presented. We focused on the Simulation methods used to represent the dynamics of the mechanical systems. For this reason, in this chapter we explain the two methods used for modeling these systems. This chapter featured an explanation in what manner a mechanical system is simulated.

The models achieved in Matlab/Simulink and their simulations allow to study the mechanical system behavior, and to recognize the contribution of each modelled effect to the dynamics of the system. The results obtained could be useful for establishing adapted control laws.

7. References

- Lee, T H.; Koh, E K. & Loh M K. (1996). Stable adaptive Control of Multivariable Servomechanisms, with Application to passive line-of-Sight Stabilization System, *IEEE Transactions on Industrial Electronics*, Vol. 43, No.1, pp. 98-105, February 1996.
- Leghmizi, S. & Liu, S. (2010). Kinematics Modeling for Satellite Antenna Dish Stabilized Platform, *2010 International Conference on Measuring Technology and Mechatronics Automation*, pp. 558 - 563, Changsha, China, March 13 - 14, 2010
- Leghmizi, S.; Fraga, R.; Liu, S.; Later, K.; Ouanzar, A. & Boughelala, A. (2011). Dynamics Modeling for Satellite Antenna Dish Stabilized Platform, *2011 International Conference on Computer Control and Automation*, Jeju Island, South Korea, 1st-3rd, May 2011
- Moradi, M.H. (2003). New techniques for PID controller design, *Proceeding of IEEE International Conference on Control Applications*, Vol. 2, pp. 903 - 908, 2003
- Matlab Help documentation
- Nguyen, T. (1995) SIMULINK A Tutorial, available from:
<http://edu.levitas.net/Tutorials/Matlab/about.html>
- Parlos, AG. (2001). Introduction to Simulink, In: Department of Mechanical Engineering Student Information Retrieval System Texas A&M University, September 13th 2009, Available from:
<http://www1.mengr.tamu.edu/aparlos/MEEN651/SimulinkTutorial.pdf>

- Popinchalk, S. (2009). Modeling Mechanical Systems: The Double Pendulum, In: *Mathworks Blogs Seth on Simulink*, February 26th 2009, Available from: <http://blogs.mathworks.com/seth/2009/02/26/modeling-mechanical-systems-the-double-pendulum/>
- Stewart, D. (1965). A platform with six degrees of freedom, *Proceedings of the Institution of Mechanical Engineers*, Vol.180, pp. 371-386, ISSN 0020-3483
- The MathWorks, Inc. (1990- 1999). The Student Edition of Simulink Dynamic System Simulation for Matlab User's Guide
- Wood, GD.; Kennedy, DC. (2003). Simulating mechanical systems in Simulink with SimMechanics, in: *Technical report of The MathWorks, Inc.*, Available from: www.mathworks.com.



Applications of MATLAB in Science and Engineering

Edited by Prof. Tadeusz Michalowski

ISBN 978-953-307-708-6

Hard cover, 510 pages

Publisher InTech

Published online 09, September, 2011

Published in print edition September, 2011

The book consists of 24 chapters illustrating a wide range of areas where MATLAB tools are applied. These areas include mathematics, physics, chemistry and chemical engineering, mechanical engineering, biological (molecular biology) and medical sciences, communication and control systems, digital signal, image and video processing, system modeling and simulation. Many interesting problems have been included throughout the book, and its contents will be beneficial for students and professionals in wide areas of interest.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Leghmizi Said and Boumediene Latifa (2011). Modeling and Control of Mechanical Systems in Simulink of Matlab, Applications of MATLAB in Science and Engineering, Prof. Tadeusz Michalowski (Ed.), ISBN: 978-953-307-708-6, InTech, Available from: <http://www.intechopen.com/books/applications-of-matlab-in-science-and-engineering/modeling-and-control-of-mechanical-systems-in-simulink-of-matlab>

INTECH

open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](#), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.