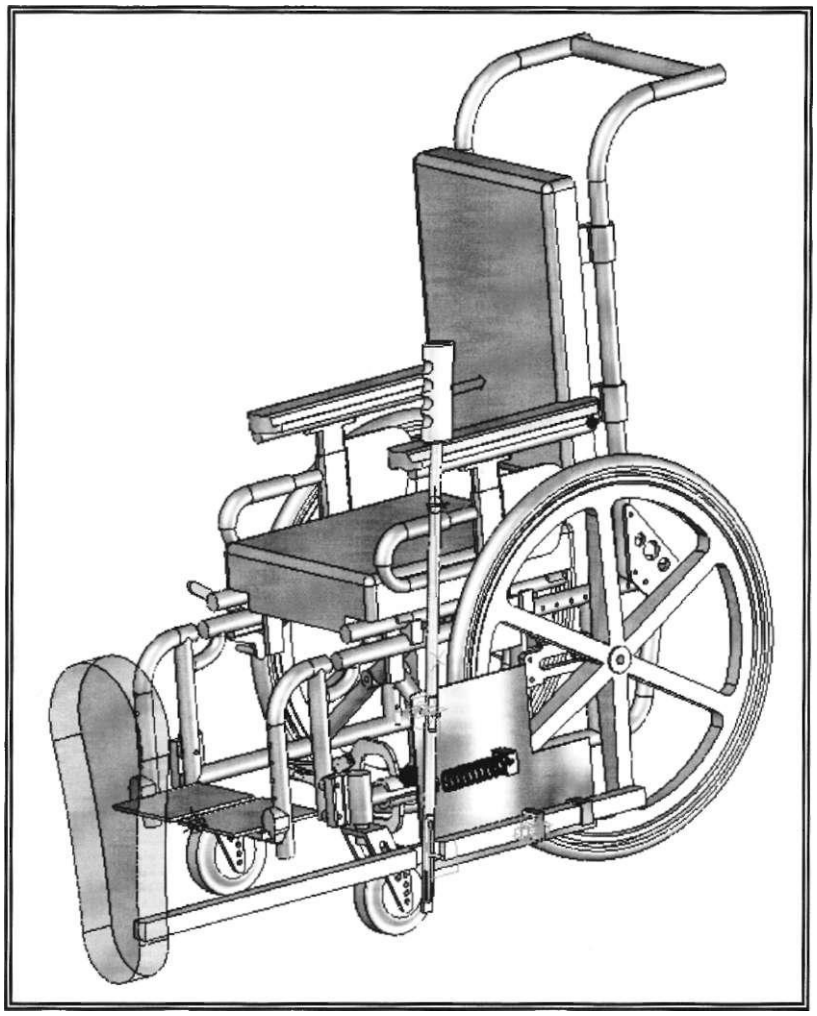
Motion Simulation and Mechanism (Design

with COSMOSMotion 2007



Kuang-Hua Chang, Ph.D.
School of Aerospace and Mechanical Engineering
The University of Oklahoma

SDC
PUBLICATIONS
Schroff Development Corporation
www.schroff.com

Better Textbooks. Lower Prices.

Motion Simulation m£ Mechanism (Design

with COSMOSMotion 2007

Kuang-Hua Chang, Ph.D.
School of Aerospace and Mechanical Engineering
The University of Oklahoma

ISBN: 978-1-58503-482-6

SDC
PUBLICATIONS

Preface

This book is written to help you become familiar with *COSMOSMotion,* an add-on module of the *SolidWorks* software family, which supports modeling and analysis (or simulation) of mechanisms in a virtual (computer) environment. Capabilities in *COSMOSMotion* support you to use solid models created in *SolidWorks* to simulate and visualize mechanism motion and performance. Using *COSMOSMotion* early in the product development stage could prevent costly (and sometimes painful) redesign due to design defects found in the physical testing phase. Therefore, using *COSMOSMotion* for support of design decision making contributes to a more cost effective, reliable, and efficient product design process.

This book covers the basic concepts and frequently used commands required to advance readers from a novice to an intermediate level in using *COSMOSMotion.* Basic concepts discussed in this book include model generation, such as assigning moving parts and creating joints and constraints; carrying out simulation and animation; and visualizing simulation results, such as graphs and spreadsheet data. These concepts are introduced using simple, yet realistic examples.

Verifying the results obtained from the computer simulation is extremely important. One of the unique features of this book is the incorporation of theoretical discussions for kinematic and dynamic analyses in conjunction with the simulation results obtained using *COSMOSMotion.* The purpose of the theoretical discussions lies in solely supporting the verification of simulation results, rather than providing an in-depth discussion on the subject of mechanism design. *COSMOSMotion* is not foolproof. It requires a certain level of experience and expertise to master the software. Before arriving at that level, it is critical for you to verify the simulation results whenever possible. Verifying the simulation results will increase your confidence in using the software and prevent you from being fooled (hopefully, only occasionally) by any erroneous simulations produced by the software.

Example files have been prepared for you to go through the lessons, including *SolidWorks* parts and assemblies, as well as completed *COSMOSMotion* models. You may want to start each lesson by reviewing the introduction and model sections and opening the assembly in *COSMOSMotion* to see the motion simulation, in hope of gaining more understanding about the example problems. In addition, *Excel* spreadsheets that support the theoretical verifications of selected examples are also available. You may download all model files and *Excel* spreadsheets from the web site of *Schroff Development Corporation* at:

http://www.schroff.com/resources

This book is written following a project-based learning approach and is intentionally kept simple to help you learn *COSMOSMotion.* Therefore, this book may not contain every single detail about *COSMOSMotion.* For a complete reference of *COSMOSMotion,* you may use on-line help in *COSMOSMotion,* or visit the web site *of SolidWorks Corporation* at:

http://www.solidworks.com/

This book should serve self-learners well. If such describes you, you are expected to have basic *Physics* and *Mathematics* background, preferably a Bachelor's degree in science or engineering. In addition, this book assumes that you are familiar with the basic concept and operation of *SolidWorks* part and assembly modes. A self-learner should be able to complete all lessons in this book in about fifty hours. An investment of fifty hours should advance you from a novice to an intermediate level user.

This book also serves class instructions well. The book will most likely be used as a supplemental textbook for courses like *Mechanism Design, Rigid Body Dynamics, Computer-Aided Design,* or

Computer-Aided Engineering. This book should cover four to six weeks of class instruction, depending on how the courses are taught and the technical background of the students. Some of the exercise problems given at the end of the lessons may require significant effort for students to complete. The author strongly encourages instructors and/or teaching assistants to go through those exercises before assigning them to students.

KHC
Norman, Oklahoma
May 15,2008

Acknowledgements

I would like to thank my family for the patience and support they have given to me in completing this book, especially, my wife Sheng-Mei for her unconditional giving and encouragement. Thanks are due to my children, Charles and Annie, for their understanding, caring, and appreciation. Especially, I appreciate their patience in reviewing the whole book and correcting a few sentences for me.

Acknowledgment is due to Mr. Stephen Schroff at Schroff Development Corporation for his encouragement and help. Without his encouragement, this book would still be in its primitive stage.

Thanks are also due to undergraduate students at the University of Oklahoma (OU) for their help in testing examples included in this book. They made numerous suggestions that improved clarity of presentation and found numerous errors that would have otherwise crept into the book. Their contributions to this book are greatly appreciated.

I am grateful to my current and former students, Thomas Cates, Petr Sramek, and Tyler Bunting, for their excellent contribution in creating examples for the application lesson; i.e., *Lesson 8*. The assistive device project employed as the example in *Lesson 8* was successful and well recognized.

Finally, I would like to thank our Creator, who has given me the strength and intelligence to complete this book.

About the Author

Dr. Kuang-Hua Chang is a *Williams Companies Foundation Presidential Professor* at the University of Oklahoma (OU), Norman, OK. He received his diploma in Mechanical Engineering from the National Taipei Institute of Technology, Taiwan, in 1980; and a M.S. and Ph.D. in Mechanical Engineering from the University of Iowa in 1987 and 1990, respectively. Since then, he has joined the Center for Computer-Aided Design (CCAD) at Iowa as a Research Scientist and CAE Technical Manager. In 1996, he joined Northern Illinois University as an Assistant Professor. In 1997, he joined OU.

Dr. Chang teaches mechanical design and manufacturing, in addition to conducting research in computer-aided modeling and simulation for design and manufacturing of mechanical systems as well as bioengineering applications. His research work has been published in more than 100 articles in international journals and conference proceedings. He has been invited to deliver talks and offer short courses for US and foreign companies and universities. He has also served as a technical consultant to US industry and foreign companies.

Dr. Chang won awards in both teaching and research in the past few years. He is a recipient of the SAE Ralph R. Teetor Award (2006), Outstanding Asian American Award sponsored by Oklahoma Asian American Association (2003), and Public Employee Award of OKC Mayor's Committee Award on Disability Concerns (2002). In addition, he received several awards from OU, including the OU Alumni Teaching Award (Spring 2007 and Fall 2007), Regents' Award on Superior Research and Creative Activities (2004), BP AMOCO Good Teaching Award (2002), and Junior Faculty Award (1999). Dr. Chang was named *Williams Companies Foundation Presidential Professor* in 2005 by OU President David L. Boren *for meeting the highest standards of excellence in scholarship and teaching.*

About the Cover Page

The picture displayed on the cover page is the motion model of an assistive device designed and built by engineering students at the University of Oklahoma (OU) during 2007-2008. Go Sooners! This device was created for the purpose of enhancing experience and encouraging children with physical disabilities to participate in a soccer game. This is a special mechanism that can be mounted on a wheelchair to mimic soccer ball-kicking action while being operated by a child sitting on the wheelchair with limited mobility and hand strength. This example was extracted from an undergraduate student design project that was carried out in conjunction with a local children hospital. This device was intended primarily to be used in the summer camp sponsored by the children hospital. This device has been employed as the application example to be discussed in *Lesson 8* of this book.

In addition to the soccer ball kicking device, students at OU have been involved in developing many other assistive devices. Currently, an Undergraduate student team is developing a transporting device that will help a local resident with only functional right hand move from her wheelchair to her bed and vise versa independently. Students were also involved in developing special baby crib, pediatric assistive walking device, and modification of a child walker, etc., in the past. All these projects require customized features to meet special needs. These projects have been supported by Honors College of OU, Schlumberger, and private donations. All supports are sincerely appreciated.

Table of Contents

Lesson 5: A Slider-Crank Mechanism

Lesson 6: A Compound Spur Gear Train

Lesson 7: Cam and Follower

Lesson 8: Assistive Device for Wheelchair Soccer Games

Notes:

Lesson 1: Introduction to *COSMOSMotion*

1.1 Overview of the Lesson

The purpose of this lesson is to provide you with a brief overview on *COSMOSMotion. COSMOSMotion* is a virtual prototyping tool that supports mechanism analysis and design. Instead of building and testing physical prototypes of the mechanism, you may use *COSMOSMotion* to evaluate and refine the mechanism before finalizing the design and entering the functional prototyping stage. *COSMOSMotion* will help you analyze and eventually design better engineering products. More specifically, the software enables you to size motors and actuators, determine power consumption, layout linkages, develop cams, understand gear drives, size springs and dampers, and determine how contacting parts behave, which would usually require tests of physical prototypes. With such information, you will gain insight on how the mechanism works and why it behaves in certain ways. You will be able to modify the design and often achieve better design alternatives using the more convenient and less expensive virtual prototypes. In the long run, using virtual prototyping tools, such as *COSMOSMotion,* will help you become a more experienced and competent design engineer.

In this lesson, we will start with a brief introduction to *COSMOSMotion* and the various types of physical problems that *COSMOSMotion* is capable of solving. We will then discuss capabilities offered by *COSMOSMotion* for creating motion models, conducting motion analyses, and viewing motion analysis results. In the final section, we will mention examples employed in this book and topics to learn from these examples.

Note that materials presented in this lesson will be kept brief. More details on various aspects of mechanism design and analysis using *COSMOSMotion* will be given in later lessons.

1.2 What is *COSMOSMotion?*

COSMOSMotion is a computer software tool that supports engineers to analyze and design mechanisms. *COSMOSMotion* is a module of the *SolidWorks* product family developed by *SolidWorks Corporation.* This software supports users to create virtual mechanisms that answer general questions in product design as described next. An internal combustion engine shown in Figures 1-1 and 1-2 will be used to illustrate some typical questions.

1. Will the components of the mechanism collide in operation? For example, will the connecting rod collide with the inner surface of the piston or the inner surface of the engine case during operation?

2. Will the components in the mechanism you design move according to your intent? For example, will the piston stay entirely in the piston sleeve? Will the system lock up when the firing force aligns vertically with the connecting rod?

3. How much torque or force does it take to drive the mechanism? For example, what will be the minimum firing load to move the piston? Note that in this case, proper friction forces must be added to simulate the resistance of the mechanism before a realistic firing force can be calculated.

4. How fast will the components move; e.g., the longitudinal motion of the piston?

5. What is the reaction force or torque generated at a connection (also called *joint* or *constraint)* between components (or bodies) during motion? For example, what is the reaction force at the joint between the connecting rod and the piston pin? This reaction force is critical since the structural integrity of the piston pin and the connecting rod must be ensured; i.e., they must be strong and durable enough to sustain the load in operation.

The modeling and analysis capabilities in *COSMOSMotion* will help you answer these common questions accurately and realistically, as long as the motion model is properly defined.

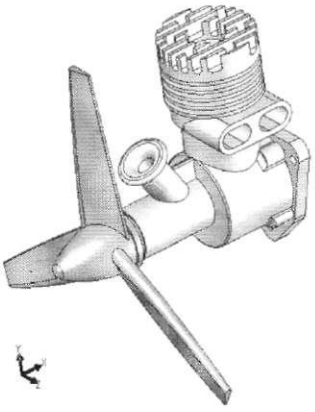


Figure 1-1 An Internal Combustion Engine (Unexploded View)

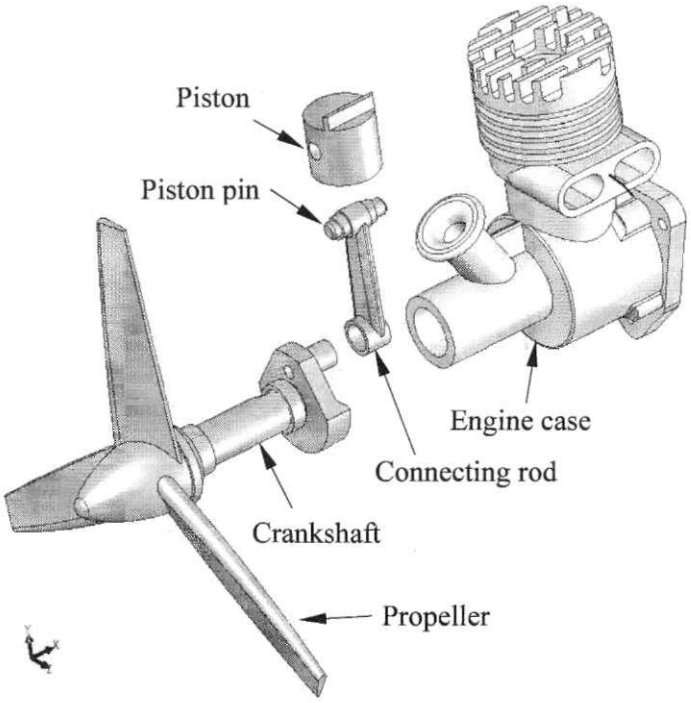


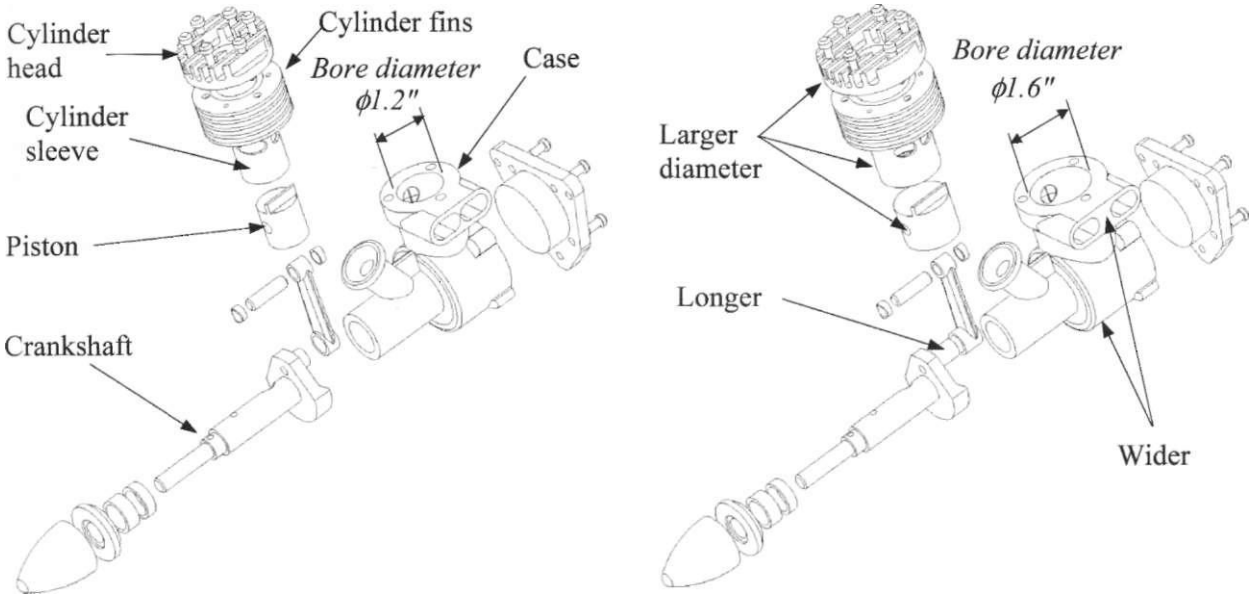
Figure 1-2 Internal Combustion Engine (Exploded View)

The capabilities available in *COSMOSMotion* also help you search for better design alternatives. A better design alternative is very much problem dependent. It is critical that a design problem be clearly defined by the designer up front before searching for better design alternatives. For the engine example, a better design alternative can be a design that reveals:

1. A smaller reaction force applied to the connecting rod, and
2. No collisions or interference between components.

In order to vary component sizes for exploring better design alternatives, the parts and assembly must be adequately parameterized to capture design intents. At the parts level, design parameterization implies creating solid features and relating dimensions properly. At the assembly level, design parameterization involves defining assembly mates and relating dimensions across parts. When a solid model is fully parameterized, a change in dimension value can be propagated to all parts affected automatically. Parts affected must be rebuilt successfully, and at the same time, they will have to maintain proper position and orientation with respect to one another without violating any assembly mates or revealing part penetration or excessive gaps. For example, in this engine example, a change in the bore

diameter of the engine case will alter not only the geometry of the case itself, but all other parts affected, such as the piston, piston sleeve, and even the crankshaft, as illustrated in Figure 1-3. Moreover, they all have to be rebuilt properly and the entire assembly must stay intact through assembly mates.

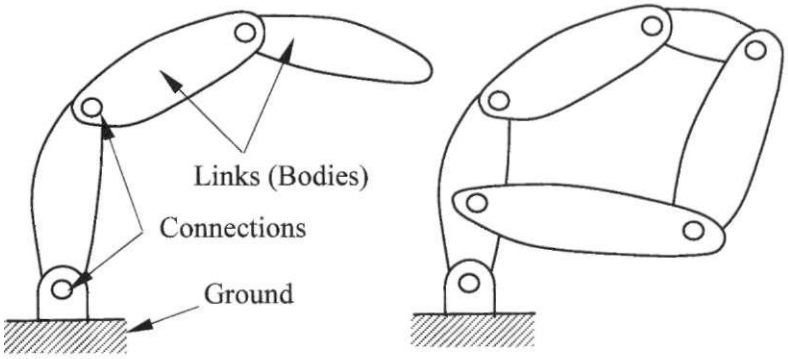


(a) Bore Diameter 1.2" (b) Bore Diameter 1.6"

Figure 1-3 A Single-Piston Engine—Exploded View

1.3 Mechanism Design and Motion Analysis

A mechanism is a mechanical device that transfers motion and/or force from a source to an output. It can be an abstraction (simplified model) of a mechanical system. A linkage consists of links (or bodies), which are connected by joints (or connections), such as a revolute joint, to form open or closed chains (or loops, see Figure 1-4). Such kinematic chains, with at least one link fixed, become mechanisms. In this book, all links are assumed rigid. In general, a mechanism can be represented by its corresponding schematic drawing for analysis purpose. For example, a slider-crank mechanism represents the engine motion, as shown in Figure 1-5, which is a closed loop mechanism.



(a) Open Loop Mechanism (b) Closed Loop Mechanism

Figure 1-4 General Mechanisms

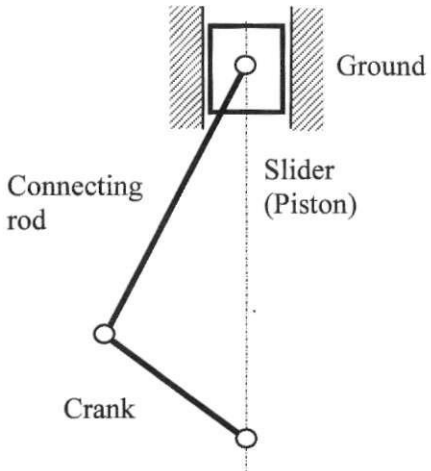


Figure 1-5 Schematic View of the Engine Motion Model

In general, there are two types of motion problems that you will have to solve in order to answer questions regarding mechanism analysis and design: kinematic and dynamic.

Kinematics is the study of motion without regard for the forces that cause the motion. A kinematic mechanism must be driven by a servomotor (or motion driver) so that the position, velocity, and acceleration of each link of the mechanism can be analyzed at any given time. Typically, a kinematic analysis must be conducted before dynamic behavior of the mechanism can be simulated properly.

Dynamic analysis is the study of motion in response to externally applied loads. The dynamic behavior of a mechanism is governed by Newton's laws of motion. The simplest dynamic problem is the particle dynamics introduced in Sophomore Dynamics—for example, a spring-mass-damper system shown in Figure 1-6. In this case, motion of the mass is governed by the following equation derived from Newton's second law,

$$\sum F = p(t) - kx - c\dot{x} = m\ddot{x} \tag{1.1}$$

where (•) appearing on top of the physical quantities represents time derivative of the quantities, m is the total mass of the block, k is the spring constant, and c is the damping coefficient.

For a rigid body, mass properties (such as the total mass, center of mass, moment of inertia, etc.) are taken into account for dynamic analysis. For example, motion of a pendulum shown in Figure 1-7 is governed by the following equation of motion,

$$Y_o M = -mgl \sin\theta = I\ddot{\theta} = m\ddot{\ell}\theta \tag{1.2}$$

where M is the external moment (or torque), / is the polar moment of inertia of the pendulum, m is the pendulum mass, g is the gravitational acceleration, and θ is the angular acceleration of the pendulum.

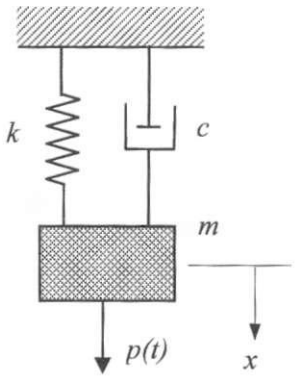


Figure 1-6 The Spring-Mass-Damper System

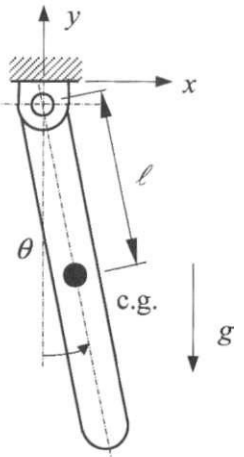


Figure 1-7 A Simple Pendulum

Dynamic analysis of a rigid body system, such as the single piston engine shown in Figure 1-3, is a lot more complicated than the single body problems. Usually, a system of differential and algebraic equations governs the motion and the dynamic behavior of the system. Newton's law must be obeyed by every single body in the system at all time. The motion of the system will be determined by the loads acting on the bodies or joint axes (e.g., a torque driving the system). Reaction loads at the joint connections hold the bodies together.

Note that in *COSMOSMotion,* you may create a kinematic analysis model; e.g., using a motion driver to drive the mechanism; and then carry out dynamic analyses. In dynamic analysis, position, velocity, and acceleration results are identical to those of kinematic analysis. However, the inertia of the bodies will be taken into account for analysis; therefore, reaction forces will be calculated between bodies.

1.4 *COSMOSMotion* Capabilities

Overall Process

The overall process of using *COSMOSMotion* for analyzing a mechanism consists of three main steps: model generation, analysis (or simulation), and result visualization (or post-processing), as illustrated in Figure 1-8. Key entities that constitute a motion model include ground parts that are always fixed, moving parts that are movable, joints and constraints that connect and restrict relative motion between parts, servo motors (or motion drivers) that drive the mechanism for kinematic analysis, external loads (force and torque), and the initial conditions of the mechanism. More details about these entities will be discussed later in this lesson.

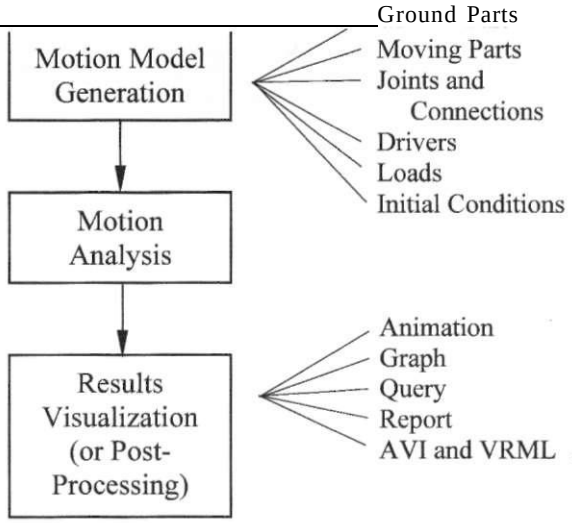


Figure 1-8 General Process of Using *COSMOSMotion*

The analysis or simulation capabilities in *COSMOSMotion* employ simulation engine, *ADAMS/Solver,* which solves the equations of motion for your mechanism. *ADAMS/Solver* calculates the position, velocity, acceleration, and reaction forces acting on each moving part in the mechanism. Typical simulation problems, including static (equilibrium configuration) and motion (kinematic and dynamic), are supported. More details about the analysis capabilities in *COSMOSMotion* will be discussed later in this lesson.

The analysis results can be visualized in various forms. You may animate motion of the mechanism, or generate graphs for more specific information, such as the reaction force of a joint in time domain. You may also query results at specific locations for a given time. Furthermore, you may ask for a report on results that you specified, such as the acceleration of a moving part in the time domain. You may also convert the motion animation to an AVI for faster viewing and file portability. In addition to AVI, you can export animations to VRML format for distribution on the Internet. You can then use *Cosmo Player,* a plug-in to your Web browser, to view VRML files. To download *Cosmo Player* for *Windows,* go to http://ovrt.nist.gov/cosmo/.

Operation Mode

COSMOSMotion is embedded in *SolidWorks.* It is indeed an add-on module of *SolidWorks,* and transition from *SolidWorks* to *COSMOSMotion* is seamless. All the solid models, materials, assembly mates, etc. defined in *SolidWorks* are automatically carried over into *COSMOSMotion. COSMOSMotion* can be accessed through menus and windows inside *SolidWorks.* The same assembly created in *SolidWorks* can be directly employed for creating motion models in *COSMOSMotion.* In addition, part geometry is essential for mass property computations in motion analysis. In *COSMOSMotion,* all mass properties calculated in *SolidWorks* are ready for use. In addition, the detailed part geometry supports interference checking for the mechanism during motion simulation in *COSMOSMotion.*

User Interfaces

User interface of the *COSMOSMotion* is identical to that of *SolidWorks,* as shown in Figure 1-9. *SolidWorks* users should find it is straightforward to maneuver in *COSMOSMotion.*

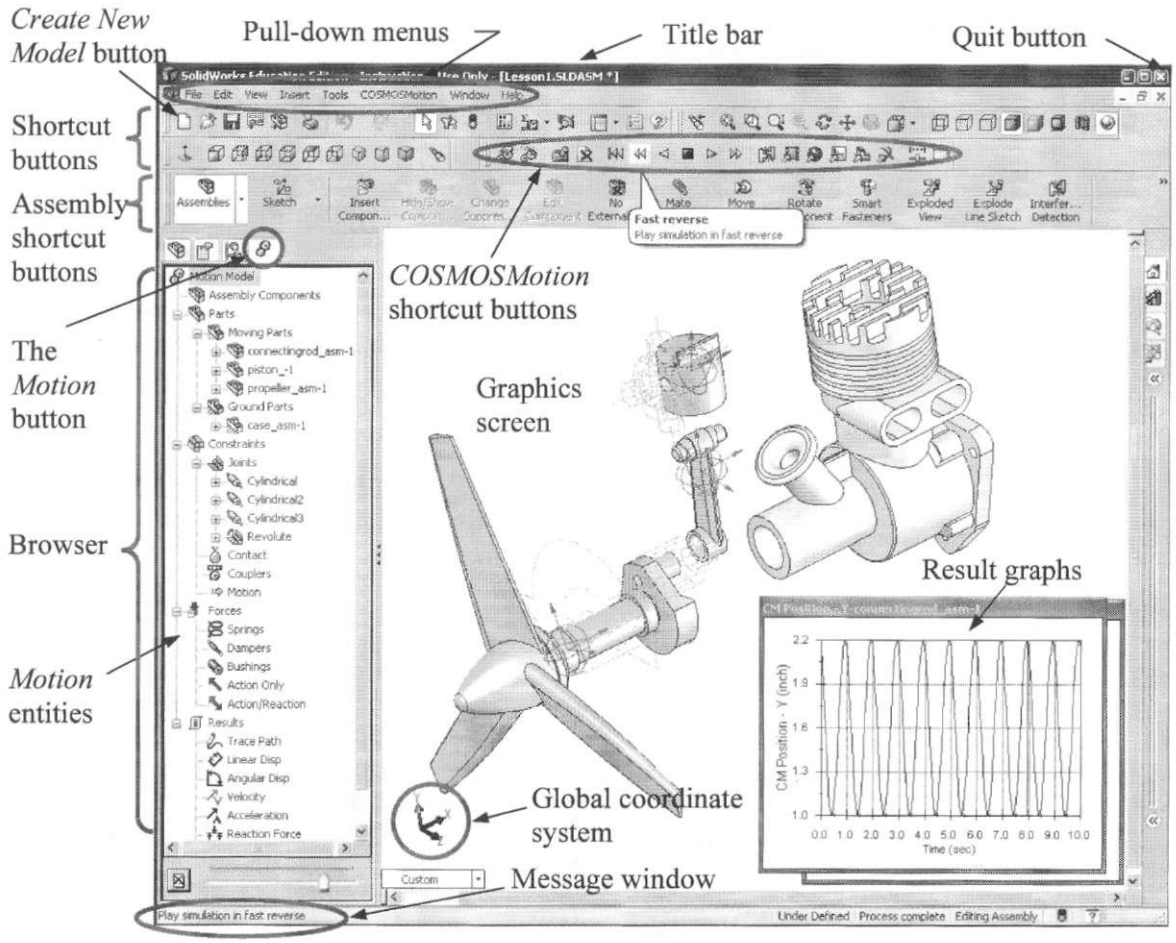


Figure 1-9 User Interface of *COSMOSMotion*

As shown in Figure 1-9, the user interface window of *COSMOSMotion* consists of pull-down menus, shortcut buttons, the browser, the graphics screen, the message window, etc. When *COSMOSMotion* is active, an extra tab $ (the *Motion* button) is available on top of the browser. This *Motion* button will allow you to access *COSMOSMotion.*

The graphics screen displays the motion model with which you are working. The global coordinate system at the lower left corner of the graphics screen is fixed and serves as the reference for all the physical parameters defined in the motion model. The pull-down menus and the shortcut buttons at the top of the screen provide typical *SolidWorks* functions. The assembly shortcut buttons allow you to assemble your *SolidWorks* model. The *COSMOSMotion* shortcut buttons on top of the graphics screen shown in Figure 1-9 provide all the functions required to create and modify the motion models, create and run analyses, and visualize results.

As you move the mouse over a button, a brief description about the functionality of the button, such as the *Fast Reverse* button shown in Figure 1-9, will appear.

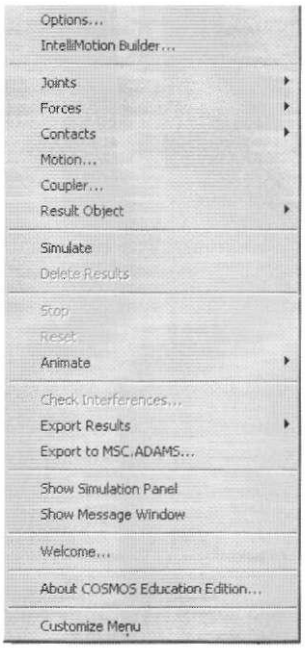


Figure 1-10

When you choose a menu option, the *Message* window, at the lower left corner shown in Figure 1-9, shows a brief description about the option. In addition to these buttons, a *COSMOSMotion* pull-down menu also provides similar options, as shown in Figure 1-10.

When you click the *Motion* button, the browser will provide you with a graphical, hierarchical view of motion model and allow you to access all *COSMOSMotion* functionalities through a combination of drag-and-drop and right-click activated menus.

For example, you may drag and drop *connectingrod_asm-l* under the *Assembly Components,* as shown in Figure 1-11 to *Moving Parts* under the *Parts* branch to define it as a moving part. You may also right click an entity and choose to define or edit its property. For example, you may right click the *Springs* node under *Forces* branch in Figure 1-12, and choose *Add Translational Spring* to add a spring.

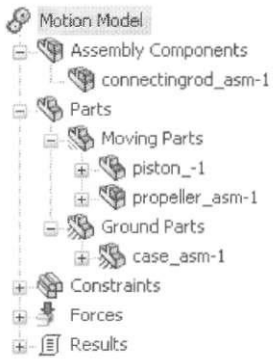


Figure 1-11

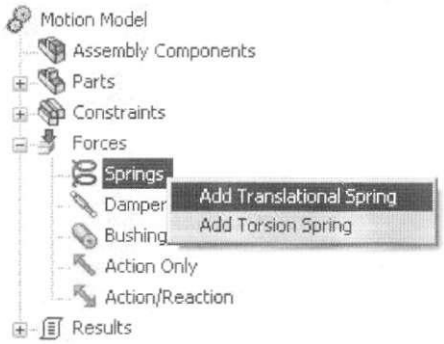


Figure 1-12

Switching back and forth between *COSMOSMotion* and *SolidWorks* assembly mode is straightforward. All you have to do is to click the *Motion* J? or *Assembly* buttons ^ (on top of the browser) when needed. When you click the *Motion* button <f? to enter *COSMOSMotion,* a different set of entities will be listed in the browser. In addition, an additional toolbar is added to *SolidWorks,* located at the top of the graphics screen, as depicted in Figure 1-13. They are the *COSMOSMotion* shortcut buttons shown in Figure 1-9. This toolbar provides settings, simulation and post processing features. Especially, the *Play Simulation* button is handy when you finish running a simulation and are ready to animate the motion. Click some of the buttons and try to get familiar with their functions. The shortcut buttons in *COSMOSMotion* and their functions are also summarized in Table 1-1 with a few more details.

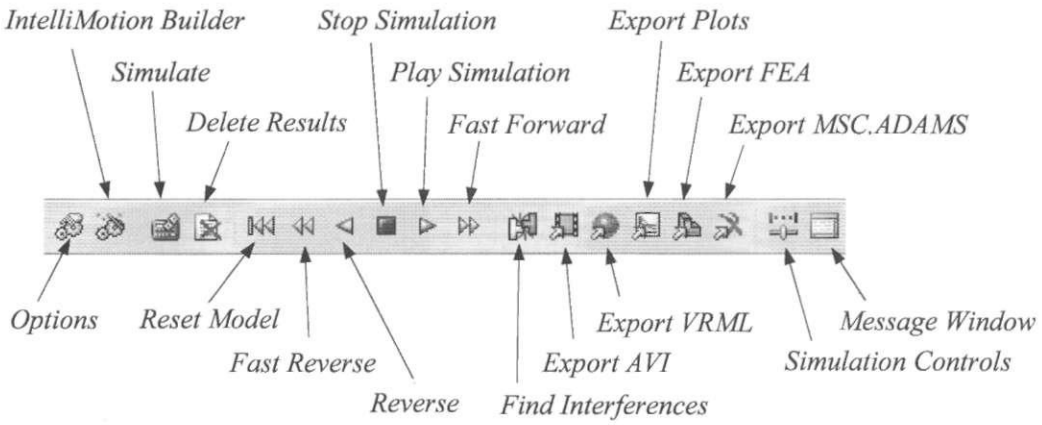


Figure 1-13 The Shortcut Buttons in *COSMOSMotion*

In addition to using the browser, another convenient way of using *COSMOSMotion* is the *IntelliMotion Builder*. The *IntelliMotion Builder* is the primary interface in *COSMOSMotion*. It is a

tabbed dialog box and a wizard that leads you through the process of converting an assembly model into a motion model, performing motion simulations, and viewing simulation results.

Table 1-1 The Shortcut Buttons in *COSMOSMotion*

Symbol	Name	Function
	Options	Allow you to control the default values and current settings for the mechanism in the *Options* dialog box. This options button allows you to access five major functions, including display, simulation, animation, results, and plot settings.
	IntelliMotion Builder	The *IntelliMotion Builder* is the primary interface in *COSMOSMotion*.
	Simulate	Simulate the motion of the mechanical system you created.
	Delete Results	Delete the simulation results.
	Reset Model	Reset the mechanism to its original configuration after a simulation or animation.
	Fast Reverse	Play the mechanism in the reverse direction but skips frames so the animation is faster than regular play.
	Reverse	Play every frame of the animation in the reverse direction.
	Stop Simulation	Stop a simulation or animation.
	Play Simulation	Play every frame of the animation in the forward direction (*Forward Play*).
	Fast Forward	Play the animation in the forward direction, skipping frames so that the animation is faster than *Forward Play*.
	Find Interferences	All the interferences that occur between the selected components as the assembly move through a specified range of motion.
	Export AVI	Create an AVI movie from your animation.
	Export VRML	Export your animations to VRML format for distribution on the Internet.
	Export Plots	Create a text file that contains results of your simulation.
	Export FEA	Export reaction force results in a form suitable for import by many Finite Element Analysis (FEA) programs
	Export *MSC.ADAMS*	Export your mechanism to *ADAMS/View*.*
	Simulation Controls	Show simulation control. You may also ask *COSMOSMotion* to calculate the total degrees-of-freedom of the mechanism using the *Calculate* button.
	Message Window	Show simulation message in the *Messages* window. You may want to check the messages for possible error or warnings during the simulation.

* ADAMS/View is a motion simulation solution for analyzing the complex behavior of mechanical assemblies. More can be found at http://www.mscsoftware.com/products/adams.cfm.

To use the *IntelliMotion Builder*, click the *IntelliMotion Builder* button or right-click the *Motion Model* node (the root entity of the motion model) from the browser, and then select *IntelliMotion Builder* (see Figure 1-14). The first tab is *Units*, which brings up the *Units* page.

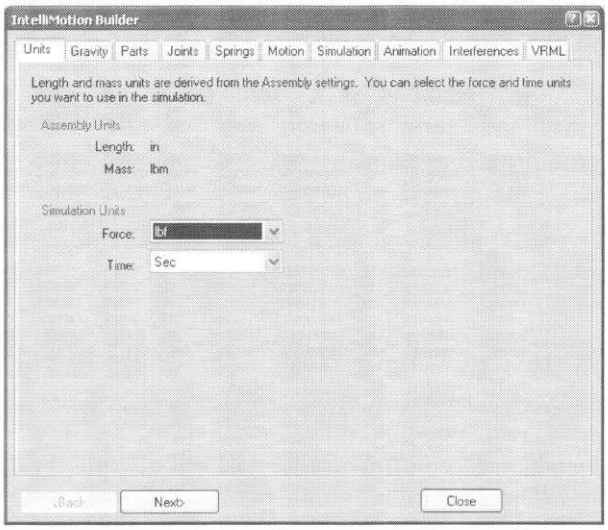


Figure 1-14 The *IntelliMotion Builder*

At the lower-left corner of each page in the *IntelliMotion Builder* are the *Back* and *Next* buttons, which help you move sequentially through the motion model creation, simulation, and animation process. You may also click a tab on top to jump to that page directly, for example *Parts*, to define ground and moving parts. For a new *COSMOSMotion* user, the *IntelliMotion Builder* is very helpful in terms of leading you through the steps of creating simulations models, running simulations, and visualizing the simulation results. For a more experienced user, the drag-and-drop and right-click activated menus may be more convenient.

Table 1-2 gives a brief explanation of each tab available in the *IntelliMotion Builder*.

Table 1-2 The Brief Summary of Capabilities Provided in *IntelliMotion Builder*

Tab Name	Function
Units	Allow you to choose or change units system.
Gravity	Allow you to specify or modify gravity.
Parts	Allows you to specify the assembly components that participate in the motion simulation as both moving and non-moving (ground) parts.
Joints	Allows you to modify joints that were automatically created from assembly constraints and add additional joints to the motion model if necessary.
Springs	Allows you to add springs to the *COSMOSMotion* model.
Motion	Allow you to add motion driver to any joint with at least one open degree of freedom.
Simulation	Allows you to specify basic parameters about the simulation and start *ADAMS/Solver*.
Animation	Allow you to control playback of an animation.
Interference	Allows you to check the *SolidWorks* components for interference as they move.
VRML	Allow you to export your animations to VRML format for distribution on the Internet.

Defining COSMOSMotion Entities

The basic entities of a valid *COSMOSMotion* simulation model consist of ground parts, moving parts, constraints (including joints), initial conditions, and forces and/or drivers. Each of the basic entities will be briefly discussed next. More details can be found in later lessons.

Ground Parts (or Ground Body)

A ground part, or a ground body, represents a fixed reference in space. The first component brought into the assembly is usually stationary; therefore, often becoming a ground part. You will have to identify moving and non-moving parts in your assembly, and assign the non-moving parts as ground parts using either the *IntelliMotion Builder* or the drag-and-drop in the browser.

Moving Parts (or Bodies)

A moving part or body is an entity represents a single rigid component (or link) that moves relatively to other parts (or bodies). A moving part may consist of a single *SolidWorks* part or an assembly composed of multiple parts. When an assembly is assigned as a moving part, none of its composing parts is allowed to move relative to one another within the assembly. A moving part has six degrees of freedom, three translational and three rotational, while a ground part has none. That is, a rigid body can translate and rotate along the Y-, and Z-axes of a coordinate system. Rotation of a rigid body is measured by referring the orientation of its local coordinate system to the global coordinate system, which is shown at the lower left corner on the graphics screen. In *COSMOSMotion*, the local coordinate system is assigned automatically, usually, at the mass center of the part. Mass properties, including total mass, inertia, etc., are calculated using part geometry and material properties referring to the local coordinate system. A moving part has a symbol (see Figure l-16a) attached, usually located at its mass center, as shown in Figure 1-15.

Constraints

A constraint (or connection) in *COSMOSMotion* can be a joint, contact, or coupler that connects two parts and constrains the relative motion between them. Typical joints include a revolute, cylindrical, spherical, etc. Each independent movement permitted by a constraint is a free degree of freedom (dof). The degrees of freedom that a constraint allows can be translational or rotational along the three perpendicular axes. The free dof is revealed by the symbol of the constraint. For example, the symbol of cylindrical joints, such as those defined in the engine example shown in Figure 1-15, show two concentric cylinders implying two free dofs, a translational and a rotational, both are along the common axis, as illustrated in Figure l-16b. Also, a revolute joint, for example the one between the propeller and the case shown in Figure 1-15, allows only one rotational dof, as depicted in a hinge symbol shown in Figure 1-16c. Understanding the joint symbols will enable you to read existing motion models. Also, a joint produces equal and opposite reactions (forces and/or torques) on the bodies connected due to Newton's 3rd Law. More about joints will be discussed in later lessons and for a list of commonly employed joints, please refer to Appendix A.

COSMOSMotion automatically converts assembly mates to joints. For example, a concentric mate together with a coincident mate will be converted to a revolute joint. Sometimes, *COSMOSMotion* will simply carry over the assembly mates to motion if there is no adequate joint to convert to, following the mapped mates established internally. For a list of common mapped mates, please refer to Appendix A. You may either stay with the joint set converted by *COSMOSMotion* or delete some of them to create your own. However, it is strongly recommended that you stay with the converted joint set before completing all the examples provided in this book. In all the examples presented in this book, mapped joints or mates are employed without any modification.

Note that instead of completely fixing all the movements, certain dofs (translational and/or rotational) are left to allow designated movement. For example, a motion driver is defined at the rotation dof of the revolute joint in the engine example, as shown in Figure 1-15. This motion driver will rotate the

propeller at a prescribed angular velocity. In addition to prescribed velocity, you may use the motion driver to drive the dof at a prescribed displacement or acceleration, both translation and rotational.

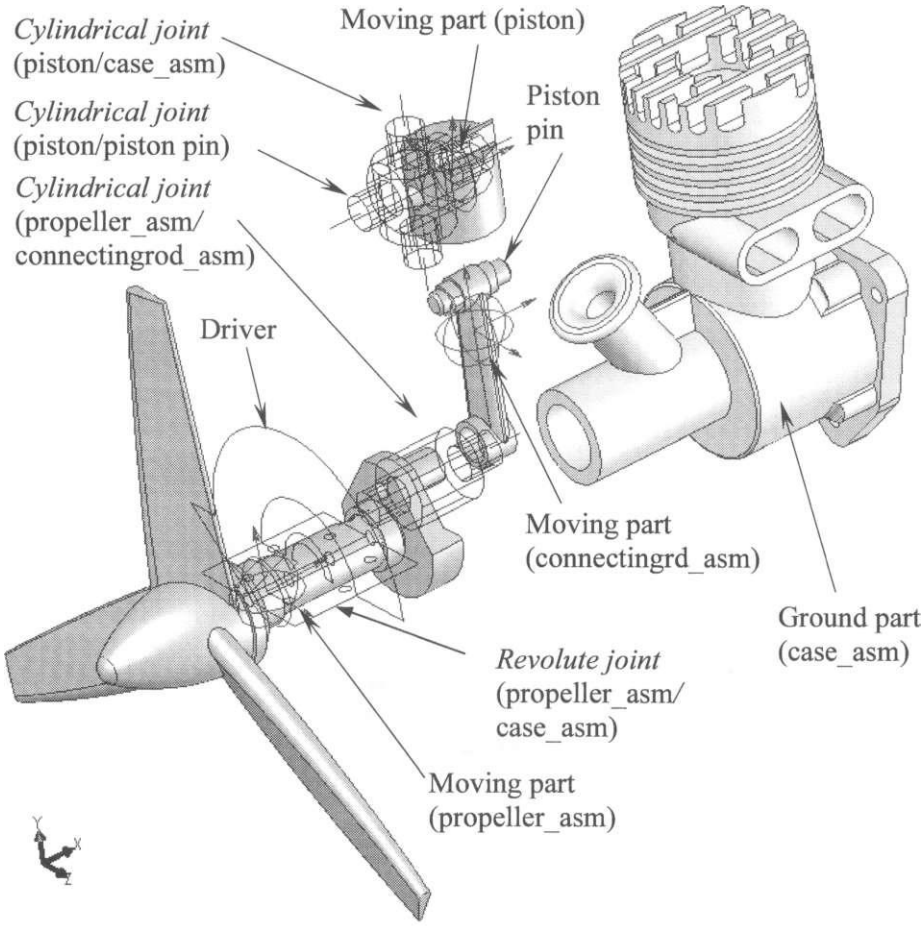
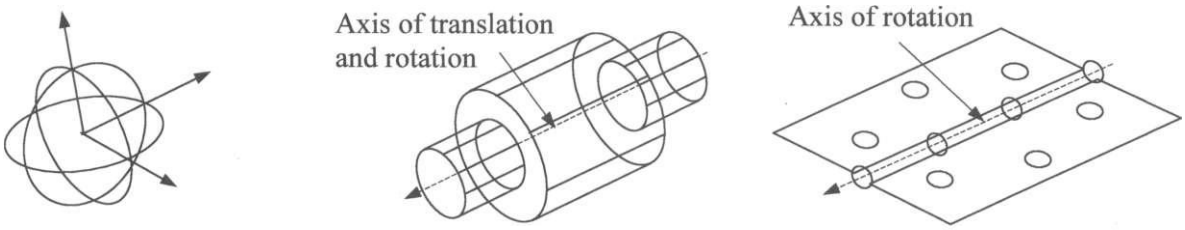


Figure 1-15 A Typical Motion Model in an Exploded View



(a) Moving Part Symbol (b) Cylindrical Joint Symbol (c) Revolute Joint Symbol

Figure 1-16 Symbols of Common Motion Entities

In addition to joints, *COSMOSMotion* provides contact and coupling constraints. The contact constraints help to simulate physical problems more realistically. *COSMOSMotion* supports four types of contact, point-curve, curve-curve, intermittent curve-curve, and 3D contact. Only the first two types of contact impose degree-of-freedom restrictions on the connected parts and are true constraints. The 3D contact is employed most frequently, which applies a force to separate the parts when they are in contact

and prevent them from penetrating each other. The 3D contact constraint will become active as soon as the parts are touching.

Joint couplers allow the motion of a revolute, cylindrical, or translational joint to be coupled to the motion of another revolute, cylindrical or translational joint. The two coupled joints may be of the same or different types. For example, a revolute joint may be coupled to a translational joint. The coupled motion may also be of the same or different type. For example, the rotary motion of a revolute joint may be coupled to the rotary motion of a cylindrical joint, or the translational motion of a translational joint may be coupled to the rotary motion of a cylindrical joint. A coupler removes one additional degree of freedom from the motion model.

Degrees of Freedom

As mentioned earlier, an unconstrained body in space has six degrees of freedom; i.e., three translational and three rotational. When joints are added to connect bodies, constraints are imposed to restrict the relative motion between them.

For example, the revolute joint defined in the engine example restricts movement on five dofs so that only one rotational motion is allowed between the propeller assembly and the engine case. Since the engine case is a ground body, the propeller assembly will rotate along the axis of the revolute joint, as illustrated in the symbol shown in Figure 1-16b. Therefore, there is only one degree of freedom left for the propeller assembly. For a given motion model, you can determine its number of degrees of freedom using the Gruebler's count.

COSMOSMotion uses the following equation to calculate the Gruebler's count:

$$D = 6M-N-0 \tag{1.3}$$

where D is the Gruebler's count representing the total degrees of freedom of the mechanism, M is the number of bodies excluding the ground body, TV is the number of dofs restricted by all joints, and O is the number of motion drivers defined in the system.

In general, a valid motion model should have a Gruebler's count *0*. However, in creating motion models, some joints remove redundant dofs. For example, two hinges, modeled using two revolute joints, support a door. The second revolute joint adds five redundant dofs. The Gruebler's count becomes:

$$D = 6x1 - 2x5 = -4$$

For kinematic analysis, the Gruebler's count must be equal to or less than *0*. The *ADAMS/Solver* recognizes and deactivates redundant constraints during analysis. For a kinematic analysis, if you create a model and try to animate it with a Gruebler's count greater than 0, the animation will not run and an error message will appear.

The single-piston engine shown in Figure 1-15 consists of three bodies (excluding the ground body), one revolute joint and three cylindrical joints. A revolute joint removes five degrees of freedom, and a cylindrical joint removes four dofs. In addition, a motion driver is added to the rotational dof of the revolute joint. Therefore, according to Eq. 1.3, the Gruebler's count for the engine example is

$$D = 6\times3 - (1\times5 - 3\times4) - 1\times1 = 0$$

If the Gruebler's count is less than zero, the solver will automatically remove redundancies. In this engine example, if the two of the cylindrical joints; between piston and the piston pin, and between the connecting rod and the crank shaft, are replaced by revolute joints, the Gruebler's count becomes

D = 6x(4-l)-(3x5-1x4) - lxl = -2

To get the Gruebler's count to zero, it is often possible to replace joints that remove a large number of constraints with joints that remove a smaller number of constraints and still restrict the mechanism motion in the same way. *COSMOSMotion* detects the redundancies and ignores redundant dofs in all analyses, except for dynamic analysis. In dynamic analysis, the redundancies lead to an outcome with a possibility of incorrect reaction results, yet the motion is correct. For complete and accurate reaction forces, it is critical that you eliminate redundancies from your mechanism. The challenge is to find the joints that will impose non-redundant constraints and still allow for the intended motion. Examples included in this book should give you some ideas in choosing proper joints.

Forces

Forces are used to operate a mechanism. Physically, forces are produced by motors, springs, dampers, gravity, tires, etc. A force entity in *COSMOSMotion* can be a force or torque. *COSMOSMotion* provides three types of forces: applied forces, flexible connectors, and gravity.

Applied forces are forces that cause the mechanism to move in certain ways. Applied forces are very general, but you must supply your own description of the force by specifying a constant force value or expression function, such as a harmonic function. The applied forces in *COSMOSMotion* include action-only force or moment (where force or moment is applied at a point on a single rigid body, and no reaction forces are calculated), action and reaction force and moment, and impact force. The force and moment symbols in *COSMOSMotion* are shown in Figure 1-17 and 1-18, respectively.

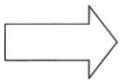


Figure 1-17 The Force (or Translational Driver) Symbol

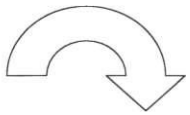
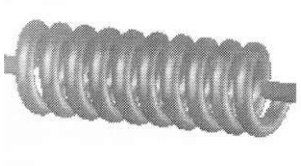
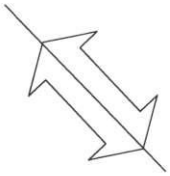


Figure 1-18 The Moment (or Rotational Driver) Symbol

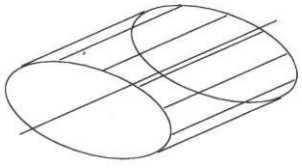
Flexible connectors resist motion and are simpler and easier to use than applied forces because you only supply constant coefficients for the forces, for instance a spring constant. The flexible connectors include translational springs, torsional springs, translational dampers, torsional dampers, and bushings, which symbols are shown in Figure 1-19.



(a) Spring



(b) Damper



(c) Bushing

Figure 1-19 Symbols of the Flexible Connectors

A magnitude and a direction must be included for a force definition. You may select a predefined function, such as a harmonic function, to define the magnitude of the force or moment. For spring and damper, *COSMOSMotion* automatically makes the force magnitude proportional to the distance or velocity between two points, based on the spring constant and damping coefficient entered, respectively. The direction of a force (or moment) can be defined by either along an axis defined by an edge or along the line between two points, where a spring or a damper is defined.

Initial Conditions

In motion simulations, initial conditions consist of initial configuration of the mechanism and initial velocity of one or more components of the mechanism. Motion simulation must start with a properly assembled solid model that determines an initial configuration of the mechanism, composed by position and orientation of individual components. The initial configuration can be completely defined by assembly mates. However, one or more assembly mates will have to be suppressed, if the assembly is fully constrained, to provide adequate movement.

In *COSMOSMotion*, initial velocity is defined as part of definition of a moving part. The initial velocity can be translational or rotational along one of the three axes.

Motion Drivers

Motion drivers are used to impose a particular movement of a joint or part over time. A motion driver specifies position, velocity, or acceleration as a function of time, and can control either translational or rotational motion. The driver symbol is identical to those of Figures 1-17 and 1-18, for translational and rotational, respectively. When properly defined, motion drivers will account for the remaining dof s of the mechanism that brings the Gruebler's count to zero or less.

In the engine example shown in Figure 1-15, a motion driver is defined at the revolute joint to rotate the propeller at a constant angular velocity.

Motion Simulation

The *ADAMS/Solver* employed by *COSMOSMotion* is capable of solving typical engineering problems, such as static (equilibrium configuration), kinematic, and dynamic, etc.

Static analysis is used to find the rest position (equilibrium condition) of a mechanism, in which none of the bodies are moving. A simple example of the static analysis is illustrated in Figure 1-20, in which an equilibrium position of the block is to be determined according to its own mass *m*, the two spring constants *ki* and k_2, and the gravity *g*.

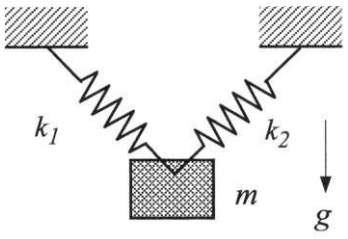


Figure 1-20 Static Analysis

As discussed earlier, kinematics is the study of motion without regard for the forces that cause the motion. A mechanism can be driven by a motion driver (e.g., a servomotor) for a kinematic analysis, where the position, velocity, and acceleration of each link of the mechanism can be analyzed at any given time. Figure 1-21 shows a servomotor drives a mechanism at a constant angular velocity.

Dynamic analysis is used to study the mechanism motion in response to loads, as illustrated in Figure 1-22. This is the most complicated and common, and usually a more time-consuming analysis.

Viewing Results

In *COSMOSMotion*, results of the motion analysis can be realized using animations, graphs, reports, and queries. Animations show the configuration of the mechanism in consecutive time frames. Animations will give you a global view on how the mechanism behaves, for example, the single-piston engine shown in Figure 1-23. You may also export the animation to AVI or VRML for various purposes.

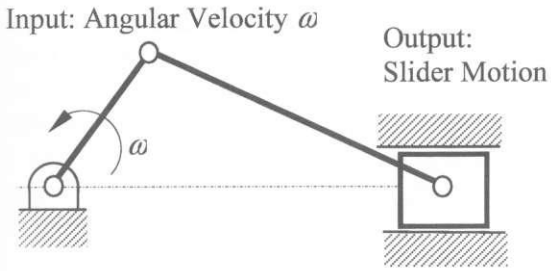


Figure 1-21 Kinematic Analysis

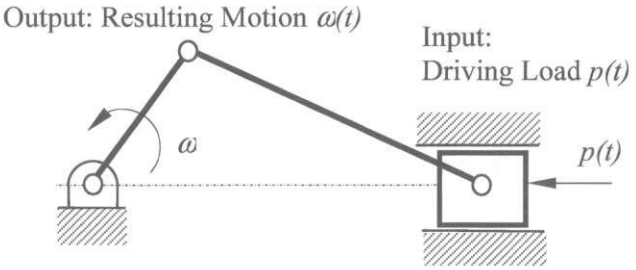


Figure 1-22 Dynamic Analysis

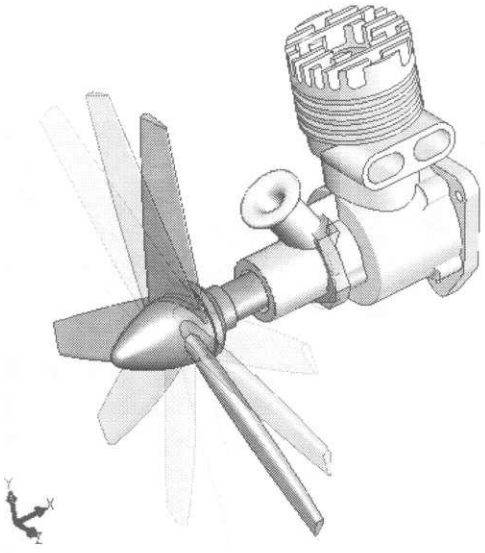


Figure 1-23 Motion Animation

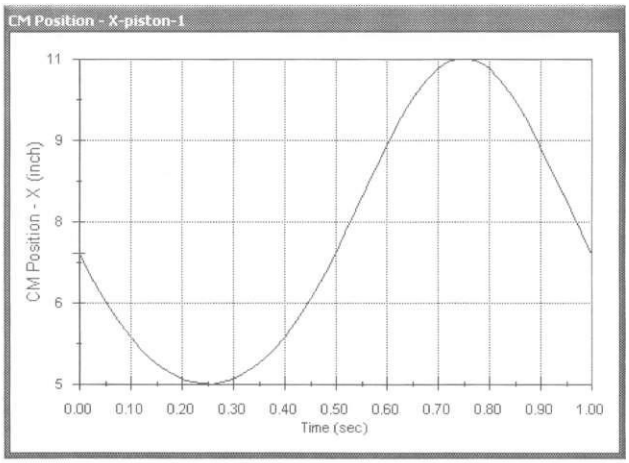


Figure 1-24 Graph of Position of the Piston vs. Time

In addition, you may choose a joint or a part to generate result graphs, for example, the position vs. time of the piston in the engine example shown in Figure 1-24. These graphs give you a quantitative understanding on the characteristics of the mechanism.

You may also query the results by moving the cursor closer to the curve and leave the cursor for a short period. The result data will appear next to the cursor. In addition, you may ask *COSMOSMotion* for a report that includes a complete set of results output in the form of textual data or a Microsoft® Excel spreadsheet.

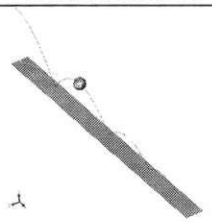
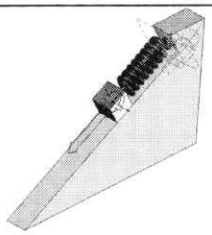
In addition to the capabilities discussed above, *COSMOSMotion* allows you to check interference between bodies during motion (please see *Lesson 5* for more details). Furthermore, the reaction forces calculated can be used to support structural analysis using, for example, *COSMOSWorks.*

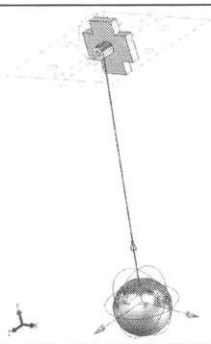
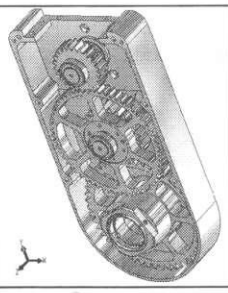
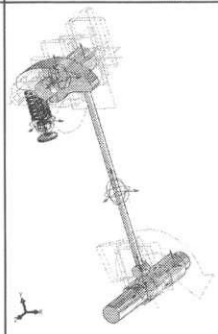
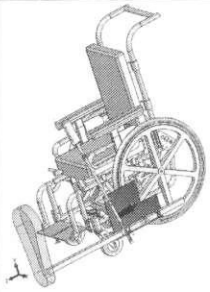
1.5 Motion Examples

Numerous motion examples will be introduced in this book to illustrate the step-by-step details of modeling, simulation, and result visualization capabilities in *COSMOSMotion.* In addition, an application example will be introduced to illustrate the steps and principles of using *COSMOSMotion* for support of mechanism design.

We will start with a simple ball-throwing example in *Lesson 2*. This example will give you a quick run-through on using *COSMOSMotion. Lessons 3* through 7 focus on modeling and analysis of basic mechanisms and dynamic systems. In these lessons, you will learn various joint types, including revolute, planar, cylindrical, etc.; forces and connections, including springs, gears, cam-followers; drivers and forces; various analyses; and graphs and results. *Lesson 8* is an application lesson, in which an assistive soccer ball kicking device that can be mounted on a wheelchair will be introduced to show you how to apply what you learn to real-world applications. All examples and main topics to be discussed in each lesson are summarized in the following table.

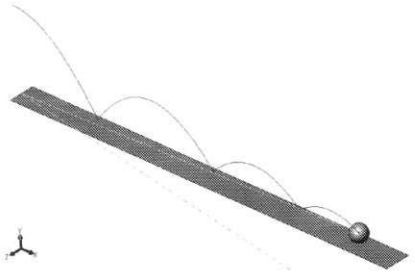
Table 1-3 Examples Employed for Discussion

Lesson	Title	Motion Model	Problem Type	Things to Learn
2	Ball Throwing Example		Particle Dynamics	1. This lesson offers a quick run-through of general modeling and simulation capabilities in *COSMOSMotion* using *IntelliMotion Builder*. 2. You will learn the general process of using *COSMOSMotion* to construct a motion model, run simulation, and visualize the motion simulation results. 3. Gravity will be turned on and a 3D contact joint will be defined between the ball and the ground to simulate the ball bouncing scenario. 4. Simulation results are verified using analytical equations of motion.
3	Spring-Mass System		Particle Dynamics	1. This is a classical spring-mass system example you learned in Sophomore *Dynamics*. 2. You will learn how to create a mechanical spring, align the block with the slope surface, and add an external force to pull the block. 3. Friction will be added between the block and the slope face. 4. Simulation results are verified using analytical equations of motion.

4	A Simple Pendulum		Particle Dynamics	1. This lesson provides more in depth about creating a revolute joint in *COSMOSMotion*. 2. Simulation results are verified using analytical equations of motion.
5	A Slider Crank Mechanism		Multibody Dynamic Analyses	1. This lesson uses a slider-crank mechanism to discuss more joint types, as well as conduct kinematic and dynamic simulations. 2. In addition to joints, you will learn to create motion driver as well as add a firing force for simulation. 3. The interference checking capability will be discussed. 4. Kinematic analysis results are verified using analytical equations of motion.
6	A Compound Spur Gear Train		Gear Train Analysis	1. This lesson focuses on simulating motion of a spur gear train system. 2. You will learn how to use *SolidWorks* and *COSMOSMotion* to create a gear connection, and simulate the gear train motion. 3. Simulation results are verified using analytical equations.
7	Cam and Follower		Multibody Dynamic Analysis	1. This lesson discusses cam and follower connection. 2. An inlet or outlet valve system of an internal combustion engine will be created and simulated. 3. Position and velocity of the valve will be graphed to monitor the motion of the system as well as assess the engineering design of the system.
8	Assistive Device for Wheelchair Soccer Game		Multibody Dynamic Analysis	1. This is an application lesson. This lesson shows you how to design an assistive device for playing wheelchair soccer game using motion simulations. 2. Numerous joints, spring, and force will be created for the system. 3. Results will be carefully reviewed to assess and modify the design of the system.

Notes:

Lesson 2: A Ball Throwing Example



2.1 Overview of the Lesson

The purpose of this lesson is to provide you a quick run-through on using *COSMOSMotion.* This example simulates a ball thrown with an initial velocity at an elevation. Due to gravity, the ball will travel following a parabolic trajectory and bounce back a few times when it hits the ground, as depicted in Figure 2-1. In this lesson, you will learn how to create a motion model to simulate the ball motion, run a simulation, and animate the ball motion. Simulation results obtained from *COSMOSMotion* can be verified using particle dynamics theory that was learned in high school *Physics.* We will review the equations of motion, calculate the position and velocity of the ball, and compare our calculations with results obtained from *COSMOSMotion.* Validating results obtained from computer simulations is extremely important. *COSMOSMotion* is not foolproof. It requires a certain level of experience and expertise to master the software. Before you arrive at that level, it will be indispensable to verify the simulation results, whenever possible. Verifying the simulation results will increase your confidence in using the software and prevent you from being occasionally fooled by the erroneous simulations produced by the software. Note that very often the erroneous results are due to modeling errors.

2.2 The Ball Throwing Example

Physical Model

The physical model of the ball example is very simple. The ball is made of *Cast Alloy Steel* with a radius of *10* in. The units system employed for this example is *IPS* (inch, pound, second). The gravitational acceleration is *386* in/sec^2. Note that you may check or change the units system by choosing from the pull-down menu

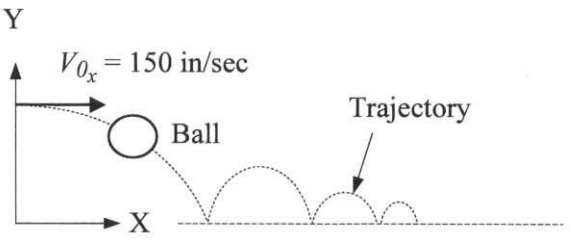


Figure 2-1 The Ball Throwing Example

Tools > Options

and choose the *Document Properties* tab in the *System Options - General* dialog box, click the *Units* node, and then pick the units system you prefer, as shown in Figure 2-2.

The ball and ground are assumed rigid. The ball will bounce back when it hits the ground. A coefficient of restitution C_R = *0.75* is specified to determine the bounce velocity (therefore, the force) when the impact occurs. For this example, C_R = *V/V*„ where *V%* and are the velocities of the ball before and after the impact. That is, the bounce velocity will be 75% of the incoming velocity, and certainly, in the opposite direction. Note that in order for *COSMOSMotion* to capture the moment when the ball hits the ground, we will define a 3D contact constraint between the ball and the ground, and use the true geometry of the parts for a finer interference calculation during the simulation.

SolidWorks Parts and Assembly

For this lesson, the parts and assembly have been created for you in *SolidWorks.* There are four files created, *ball.SLDPRT, ground. SLDPRT, Lesson2. SLDASM,* and *Lesson2withresults.SLDASM.* You can find these files at the publisher's web site (ht1p://www.schrofflxom/). We will start with *Lesson2.SLDASM,* in which the ball is fully assembled to the ground; i.e., no movement is allowed. In addition, the assembly file *Lesson2withresults.SLDASM* consists of a complete simulation model with simulation results, in which some of the assembly mates were suppressed in order to provide adequate degrees of freedom for the ball to move. You may want to open this file to see how the ball is supposed to move. Since the gravity is defined in the negative 7-direction of the global coordinate system as default, all parts and assembly are created for a motion simulation that complies with the default setting.

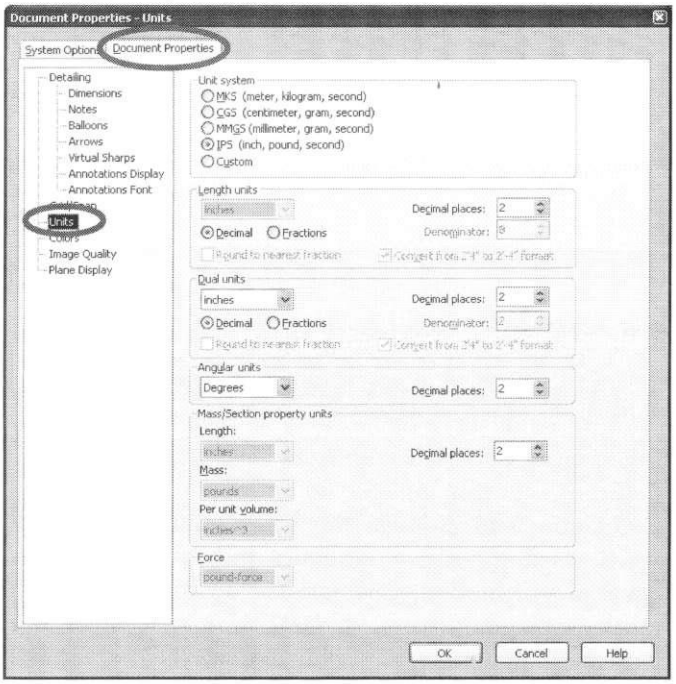


Figure 2-2 Checking Units System

The assembly *Lesson2.SLDASM* consists of two parts: the ball *(ball.SLDPRT)* and the ground *(ground.SLDPRT).* The ball is fully assembled with the ground by three assembly mates of three pairs of reference planes. They are *Front* (ball)/*Front* (ground), *Right* (ball)/'*Right* (ground), and *Top* (ball)/7b/? (ground), as shown in Figure 2-3. The distance between the reference planes *Top* (ball) and *Top* (ground) is *100* in., which defines the initial position of the ball, as shown in Figure 2-4. The radius of the ball is *10* in., and the ground is modeled as a *30*500*0.04* in.^{3} rectangular block.

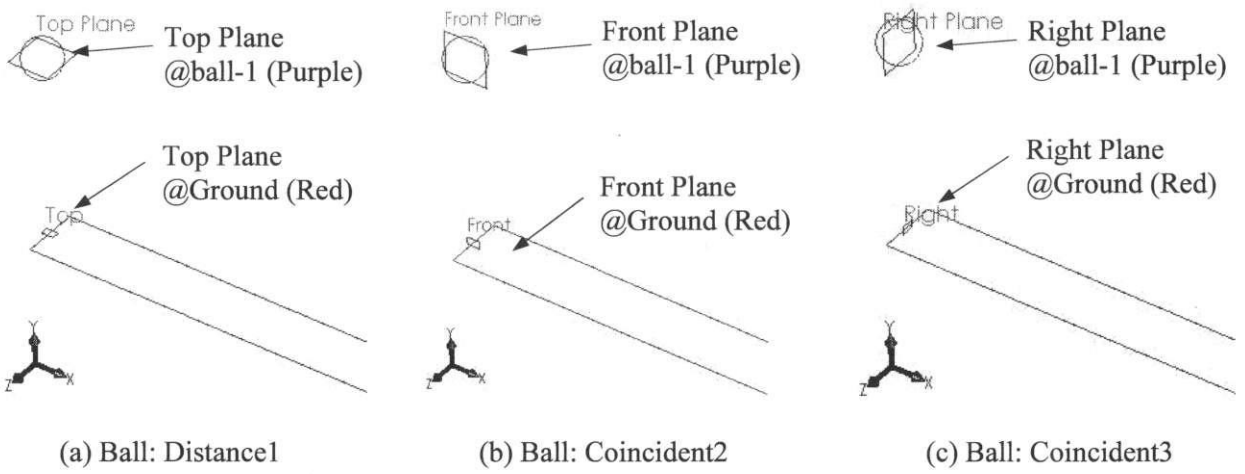
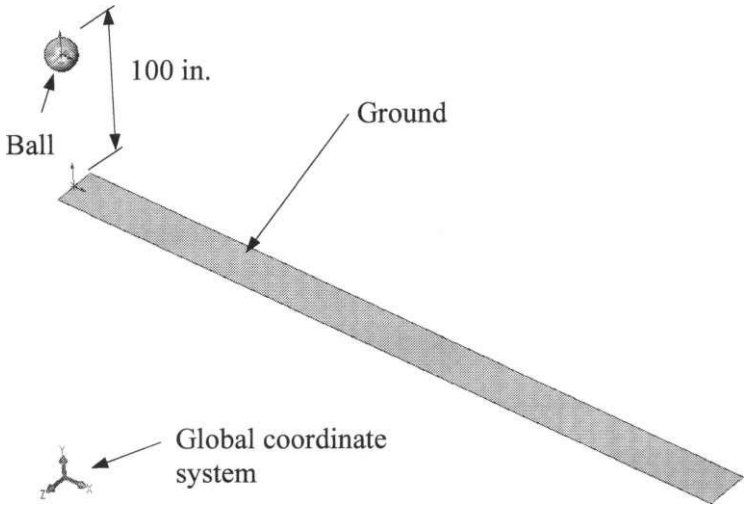


Figure 2-3 Assembly Mates Defined for the Example

Note that the 7-axis of the global coordinate system (located at the lower left corner of the *SolidWorks* graphics screen, as shown in Figure 2-4) is pointing upward, which is consistent with the default direction of the gravity, but in the opposite direction.



R 2-4 The SolidWorks Assembly

Motion Model

In this example, the ball will be the only movable body. Two assembly mates, *Distancel* and *Coincident^*, as shown in Figures 2-3a and 2-3c, will be suppressed to allow the ball to move on the *X-Y* plane As mentioned earlier, the ball will be thrown with an initial velocity of *VQ = 150* in/sec.

A gravitational acceleration *-386* in/sec^2 is defined in the 7-direction of the global coordinate system. The ball will reveal a parabolic trajectory due to gravity. A *3D contact* constraint will be added to characterize the impact between the ball and the ground. As discussed earlier, a coefficient of restitution C_R *= 0.75* will be specified to determine the force that acts on the ball when the impact occurs. In this example, no friction is assumed.

2.3 Using *COSMOSMotion*

Start *SolidWorks* and open assembly file ***Lesson2****.SLDASM.*

When *COSMOSMotion* is active, the browser has an extra tab (the *Motion* button) for the *Motion* (see the buttons on top of the browser shown in Figure 2-5). This browser provides you with a graphical, hierarchical view of the motion model and allows you to access all *COSMOSMotion* functionalities through a combination of drag-and-drop and right click menus.

Figure 2-5

Switch back and forth between *COSMOSMotion* and *SolidWorks* assembly mode is straightforward. All you have to do is to click the *Motion* and *Assembly* buttons when needed. When you click the *Motion* button to enter *COSMOSMotion,* a different set of entities will be listed in the browser, in addition, an additional toolbar is added to *SolidWorks,* located at the top of the graphics screen, as shown in Figure 2-6. This toolbar provides settings, simulation and post processing features. Especially, the *Play Simulation* button is handy when you finish running a simulation and ready to animate the motion. Click some of the buttons and try to get familiar with their functions.

In this lesson, we will use the *IntelliMotion Builder* for most of the steps. The *IntelliMotion Builder* is the primary interface in *COSMOSMotion.* It is a tabbed dialog box and a wizard that leads you through

the process of changing an assembly model into a motion model, performing the motion simulation, anc viewing the simulation results.

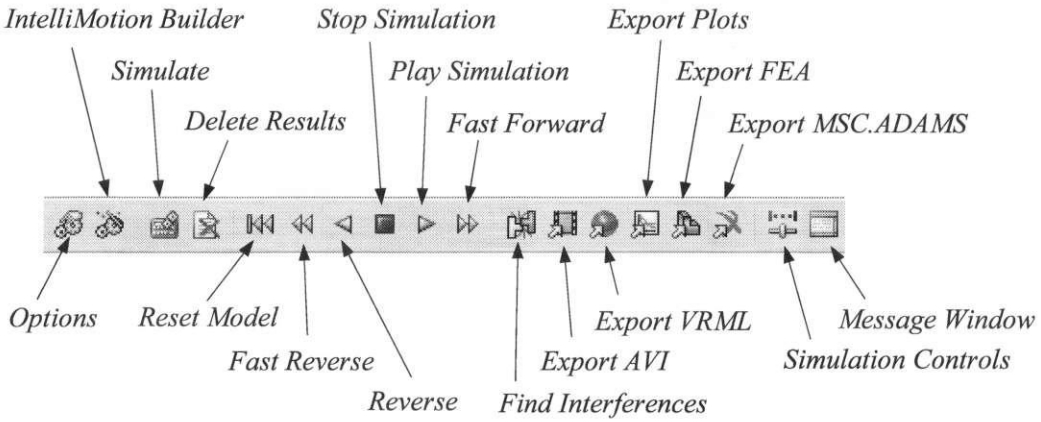


Figure 2-6

Before we start, we will suppress two assembly mates to allow the ball to move on the *X-Y* plane. Choose the *Assembly* button on top of the browser, and expand the *Mates* branch by clicking the small ⊞ button in front of it.

Distance1(Ground<1>,ball<1>), *Coincident2(Ground<1>,ball<1>)*, and *Coincident3(Ground<1>,ball<1>)* listed in the browser, as shown in Figure 2-7.

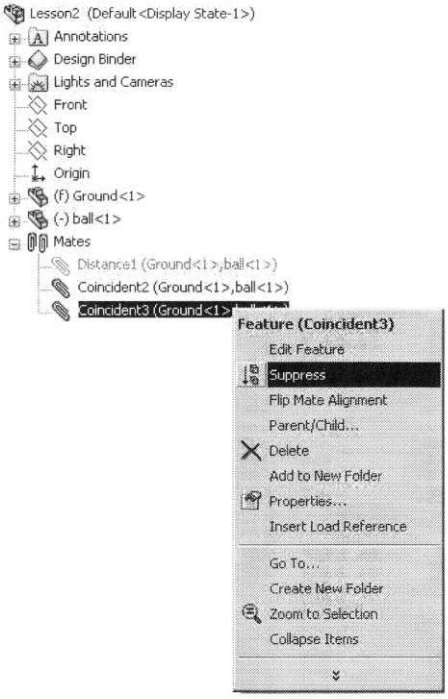


Figure 2-7

Choose *Distance 1 (Ground< 1 >, ball<1>),* press the right mouse button, and choose *Suppress.* Repeat the same for *Coincident3(Ground<l>,ball<l>).* Both mates will become inactive.

If you move the cursor to the root node, *Lesson2 (Default<Display State-1>),* in the browser, you should see a rectangular box appears in the graphics screen, which is simply the bounding box for the assembly (see Figure 2-8).

Click the *Motion* button on ton of the browser to enter *COSMOSMotion.*

Start the *IntelliMotion Builder*

To use the *IntelliMotion Builder,* right click the *Motion Model* node from the browser, and then select *IntelliMotion Builder* (see Figure 2-5). The first tab of the *IntelliMotion Builder* is *Units,* which brings up the *Units* page. As shown in Figure 2-9, the *IPS* units system has been chosen. No action is needed.

At the lower-left corner of each page in the *IntelliMotion Builder* are the *Back* and *Next* buttons, which help you move sequentially through the motion model creation, simulation, and animation process. Choose *Next* or click the *Gravity* tab.

The *Gravity* page (Figure 2-10) shows that the default acceleration is *386.22* in/sec^2 and is acting in the negative 7-direction. This is what we want, and no action is needed. Choose *Next* or click the *Parts* tab.

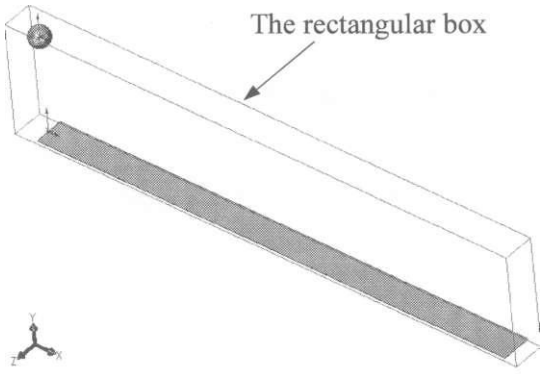


Figure 2-8

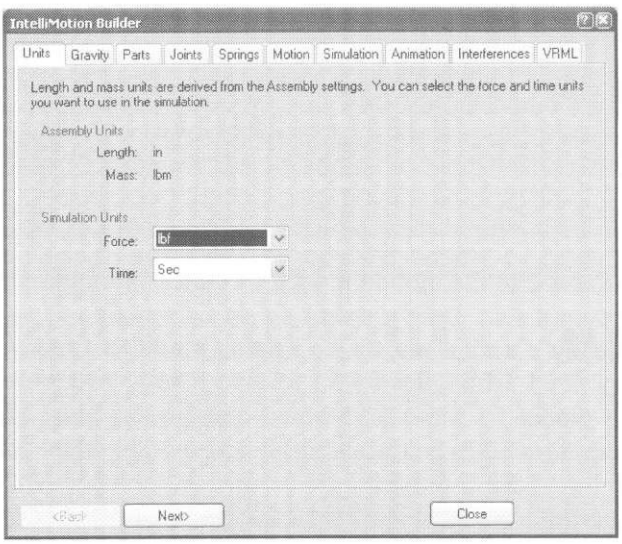


Figure 2-9 The *IntelliMotion Builder* and the *Units* Page

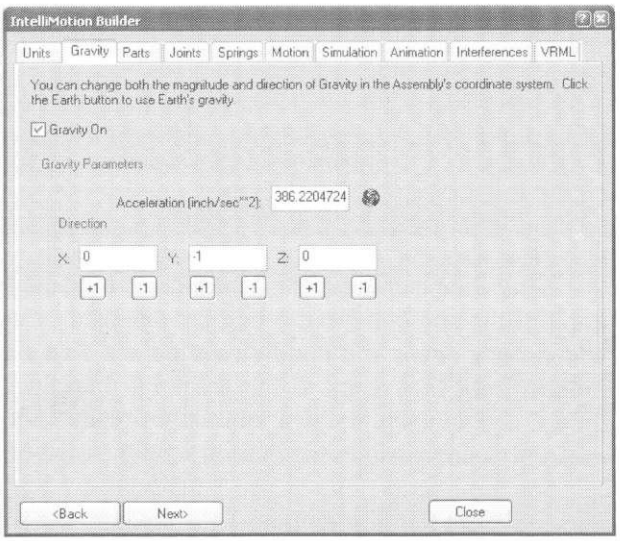


Figure 2-10 The *Gravity* Page

Defining Bodies

The first step in creating a motion model is to indicate which components from your *SolidWorks* assembly model participating in the motion model. In an assembly that does not yet have any motion parts defined, all of the assembly components are listed under the *Assembly Components* branch (right column) of the *Parts* page, as shown in Figure 2-11. We will move *ball-1* to *Moving Parts* (left column) and *Ground-1* to *Ground Parts* by using the drag-and-drop method.

Click *ball-1* and drag it by holding down the left-mouse button, moving the mouse until the cursor is over the *Moving Parts* node, and then releasing the mouse button. Part *ball-1* is now added to the motion model as moving parts (can move). Repeat the same steps to move *ground-1* to the *Ground Parts* node. Now, the part *Ground-1* is added to the motion model as a ground part (cannot move).

Note that you may select multiple components by holding *Ctrl* key and selecting each component. You may select multiple components by selecting the first component, pressing and holding *Shift* key, and then selecting the last component. All of the components between the first and second selected components will be selected. Or you can drag-select by pressing the left-mouse button and moving the

mouse so that the selection rectangle intersects the components. In this case, any components within the selection rectangle will be selected.

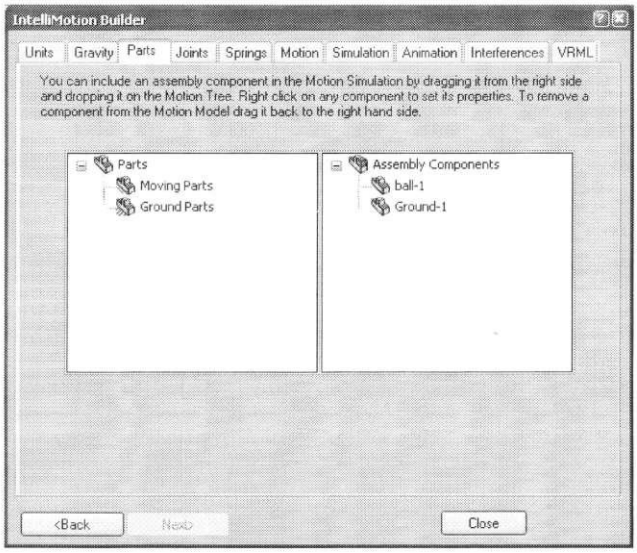


Figure 2-11 The *Parts* Page

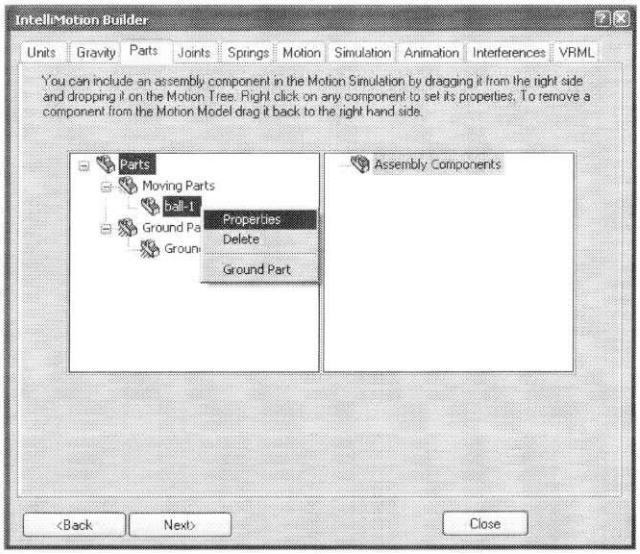


Figure 2-12

Any time a component (a part or an assembly) is added to the motion model, *COSMOSMotion* looks at all of the assembly mates that are attached to that component. If an assembly mate between the newly added component and another component that is already participating in the motion model is found, a motion joint that maps to the assembly mate is generated. This allows you to take a fully assembled model and quickly build a simulation-ready motion model by indicating which components from the assembly participate in the motion model. A list of mappings between the assembly and motion joints that are frequently encountered can be found in Appendix A. Take a few minutes to review Appendix A to become more familiar with the mapping and the joint types supported in *COSMOSMotion.* Understanding joints and the mapping will help you assemble parts adequately for motion models, avoiding unnecessary model editing and confusion.

Defining Initial Velocity

We will add an initial velocity $Vo_x = 150$ in/sec to the ball. Click *ball-1* from the *Parts* page, press the right mouse button, and choose *Properties* (see Figure 2-12), the *Edit Part* dialog box will appear. Click the *IC's* tab and enter *150* for *X Velocity,* as shown in Figure 2-13. Click *Apply.* The initial velocity has been defined.

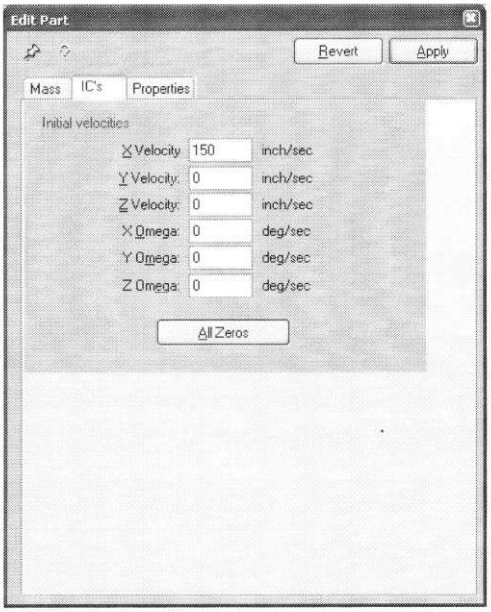


Figure 2-13 Defining Initial Velocity

Defining Joints

Choose *Next* or click the *Joints* tab. The *Joints* page allows you to modify joints that were automatically created from assembly mates. You may add additional joints to the motion model if adequate. This page contains a single tree that lists all of the joints in the motion model. As shown in Figure 2-14, there is only one joint listed, *Coincident!,* which mates *Front Plane* of ball to the *Front Plane* of the ground to allow a planar motion for the ball.

On the right, you will see a list of joint types you can choose to add to your motion model. The first joint *Revolute* is selected by default. As indicated in the message, a revolute joint removes five degrees of freedom, three translational and two rotational. However, for this ball-throwing example, the joint carried over from *SolidWorks;* i.e., *Coincident2,* is exactly what we want; therefore, no action is needed for the time being.

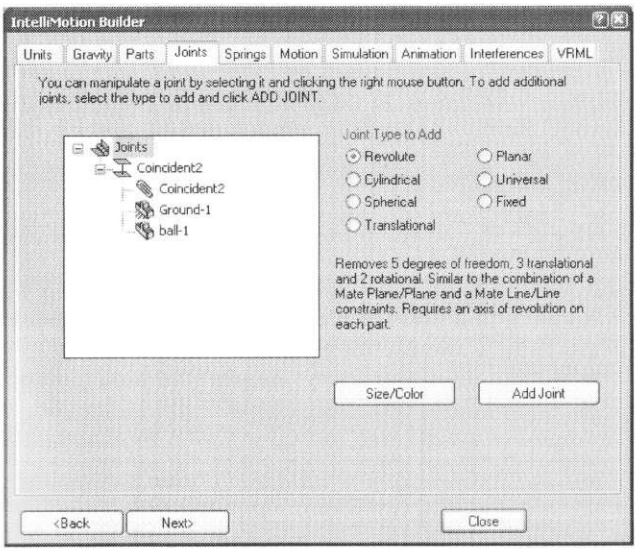


Figure 2-14 The *Joints* Tab

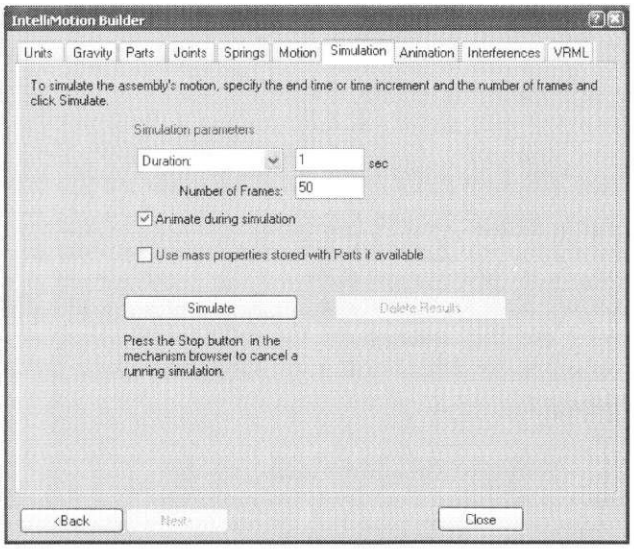


Figure 2-15 The *Simulation* Tab

Running Simulation

Click the *Next* button three times or click the *Simulation* tab directly (no spring or driver is needed for this example; therefore, we are skipping the *Spring* and *Motion* tabs). As shown in Figure 2-15, the simulation duration is *1* second and the number of frames is *50* as defaults. We will stay with these default values for the time being. Click *Simulate* to run a simulation.

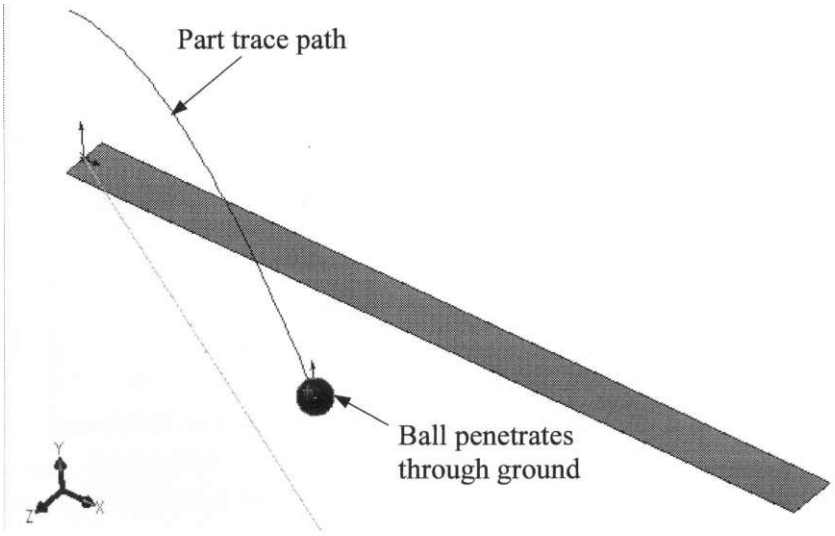


Figure 2-16 Animation: Ball Penetrating Through the Ground

After a few seconds, the ball will start moving. As shown in Figure 2-16 the ball will fall through the ground, which is not realistic. Note that the trace path that indicates the trace of the center of the ball is turned on in Figure 2-16. We will learn how to do that later. For the time being we will have to add a *3D Contact* constraint between the ball and the ground in order to make the ball bounce back when it hits the ground. The *3D Contact* constraint will create a force to prevent the ball from penetrating the ground. This constraint will be only activated if the ball comes into contact with the ground.

Defining a 3D Contact Constraint

The *3D Contact* constraint cannot be created in the *IntelliMotion Builder*. We will have to use the browser or the pull-down menu. Before creating a *3D Contact* constraint, close the *IntelliMotion Builder*, and delete the simulation result by clicking the *Delete Results* button at the bottom of the browser (see Figure 2-17 for the location of the delete button). Now, the ball should return to its initial position. Also, save your model before moving forward.

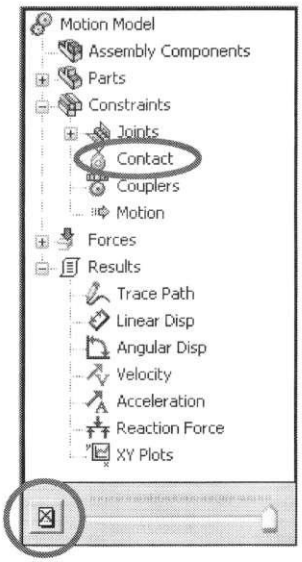


Figure 2-17

From the pull-down menu, choose

COSMOSMotion > *Contacts* > *3D Contact*

or from the browser, right click the *Contact* branch, and then select *Add 3D Contact*, as shown in Figure 2-17. The *Insert 3D Contact* dialog box will appear (Figure 2-18).

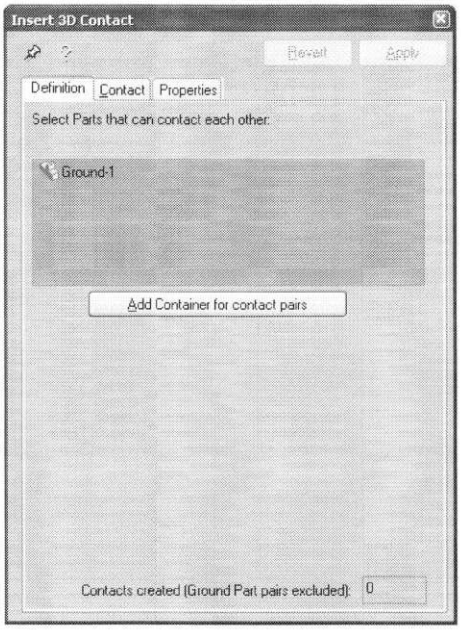


Figure 2-18 The Edit 3D Contact Dialog Box: Definition Tab

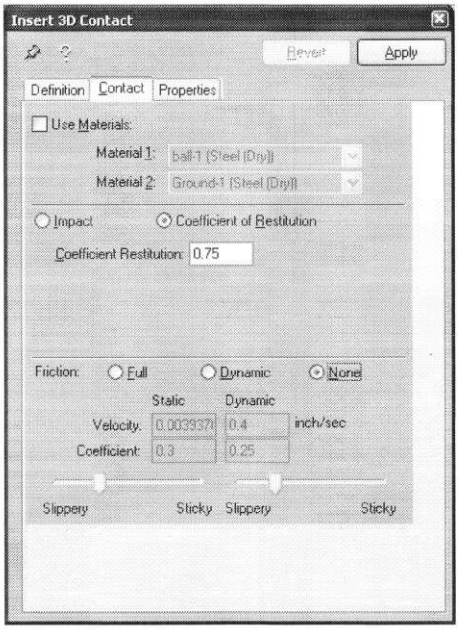


Figure 2-19 The Edit 3D Contact Dialog Box: Contact Tab

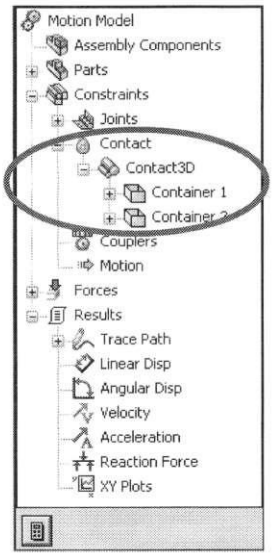


Figure 2-20

During the definition, when you pick a feature on the parts, *COSMOSMotion* selects the corresponding body (and uses it for the contact analysis). At each time frame during the simulation,

COSMOSMotion calculates whether the parts' bounding boxes (usually the rectangular box, similar to thai of Figure 2-8, but for individual parts) interfere. If they interfere, *COSMOSMotion* performs a finer interference calculation between the two bodies. At the same time, *ADAMS/Solver* computes and applies an impact force on both bodies.

From the browser, select *Ground-7,* the ground part will be listed in the first container (upper field), as shown in Figure 2-18. Click the *Add Container for contact pairs* button (in the middle), and pick *ball-1* The part *ball-1* will be listed in *Container 2* (lower field).

Click the *Contact* tab to define contact parameters. Note that we will define a coefficient of restitution for the impact. In the three sets of parameters appearing in the dialog box (Figure 2-19). turn off the *Use Materials* (deselect the entity) and *Friction* (click *None).* Choose *Coefficient of Restitution* I in the middle), and enter *0.75* for *Coefficient Restitution,* as shown in Figure 2-19. Click *Apply* to accept the *3D Contact.* The *3D Contact* constraint should appear in the browser, as shown in Figure 2-20. You will have to expand the *Constraints* branch and then the *Contact* branch to see the contact constraint.

Rerun the Simulation

Before we rerun the simulation, we will have to adjust some of the simulation parameters. Especially we will ask *COSMOSMotion* to use precise geometry to check contact in each frame of simulation.

Right click the *Motion Model* node from the browser and choose *Simulation Parameters.* In the *COSMOS Education Edition Options* dialog box appearing (Figure 2-21), choose *Use Precise* Geo/?;, *for 3D Contact,* and enter *3* seconds for *Duration* and *500* for *Number of Frames.* Note that we increase the number of frame so that we will see smooth graphs in various result displays. Increasing the number of frame will certainly increase the simulation time. However, the increment is insignificant for this simple example. Click *OK io* accept the definition and close the dialog box.

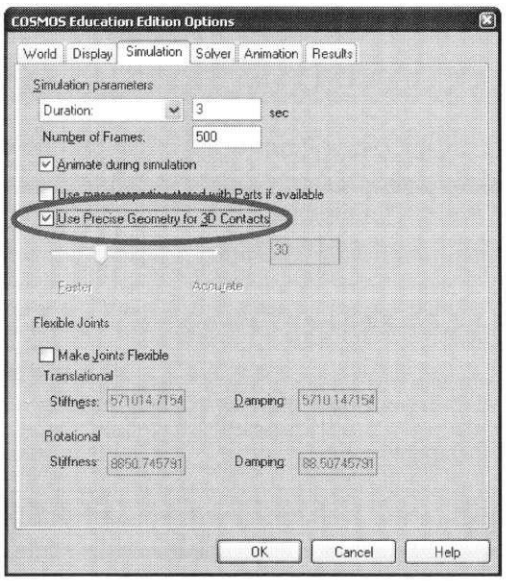


Figure 2-21 Use Precise Geometry to Detect Contact in Simulation

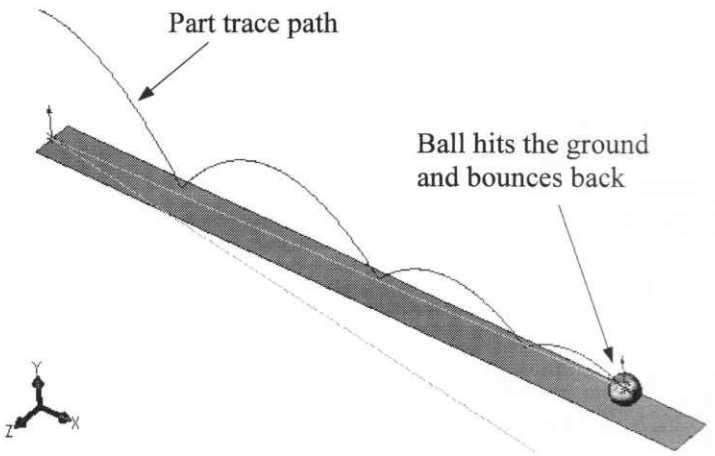


Figure 2-22 Animation: Ball Bouncing Back

Right click the *Motion Model* node from the browser and choose *Run Simulation.* After a few seconds, the ball starts moving. As shown in Figure 2-22 the ball will hit the ground and bounce back a

few times before the simulation ends. The ball did not fall through the ground this time due to the addition of the *3D Contact* constraint.

Displaying Simulation Results

COSMOSMotion allows you to graphically display the path that any point on any moving part follows. This is called a trace path. Note that the trace path of the ball was displayed in both Figures 2-16 and 2-22. We will first learn how to create a part trace path.

From the browser, right click the *Results* node, and choose *Create Trace Path* (see Figure 2-23) to bring up the *Edit Trace Path* dialog box, as shown in Figure 2-24. Note that *Assem2* should be listed in the *Select Reference Component* text box, which serves as the default reference frame for the trace path. The default reference frame is the global reference frame included as part of the ground body. No change is needed.

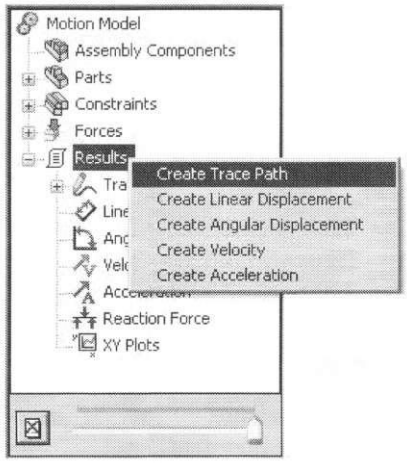


Figure 2-23 Right Click to Define Trace Path

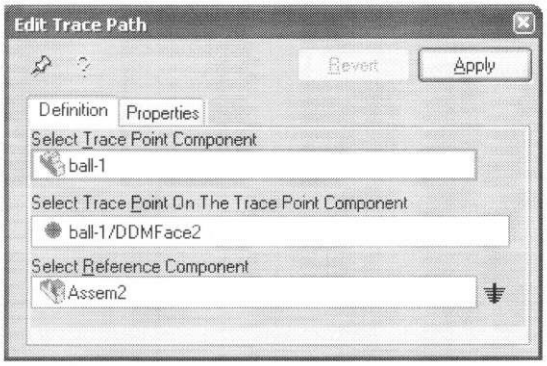


Figure 2-24 The *Edit Trace Path* Dialog Box

To select the part used to generate the trace curve, select the *Select Trace Point Component* text field (should be highlighted in red already), and then select one moving part; i.e., the ball, from the graphics screen. The part *ball-1* will be listed in the *Select Trace Point Component* text field, and *ball-l/DDMFace2* is listed in the *Select Trace Point on the Trace Point Component* text box.

Click *Apply* button, you should see the trace path appears in the graphics screen, similar to that of Figure 2-22.

Next, we will create a graph for the 7-position of the ball using the *XY Plots.* From the browser, right click *ball-1* (under *Parts, Moving Parts)* and choose *Plot > CM Position > Y.*

The *XY Plot* for the *CM* (Center of Mass) position of the ball in the 7-direction will appear, similar to that of Figure 2-25. Note that you may adjust properties of the graph, for instance the axis scales, following steps similar to those of Microsoft® Excel spreadsheet graphs.

The graph shows that the ball was thrown from *Y= 100* in. and hits the ground at *Y= 10* in. (CM of the ball) and time about *t— 0.6* seconds. The ball bounces back and moves up to an elevation determined by the coefficient of restitution. The motion continues until reaching the end of the simulation.

Note that you may click any location in the graph to bring up a fine red vertical line that correlates the graph with the position of the ball in animation. As shown in the graph of Figure 2.25, at *t = 1.4* seconds, the ball is roughly at *Y= 44* in. The snapshot of the ball at that specific time and 7-location is shown in the graphics screen.

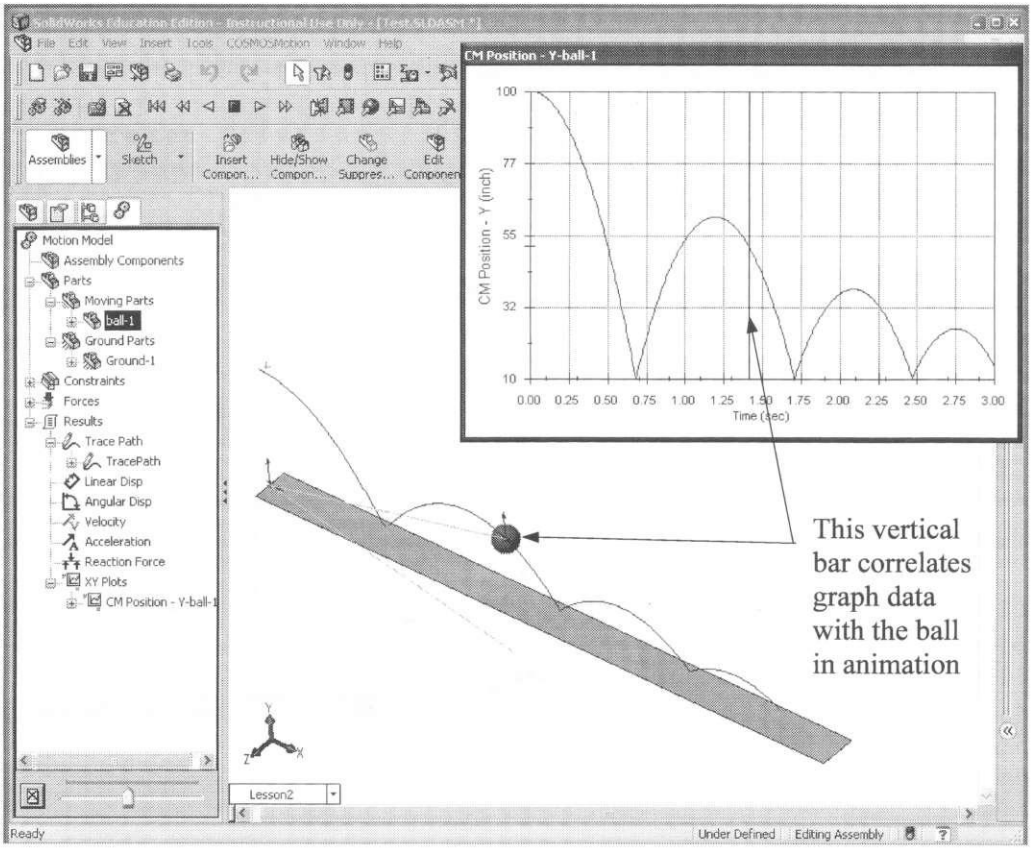


Figure 2-25 Result Graph and Motion Animation in *COSMOSMotion*

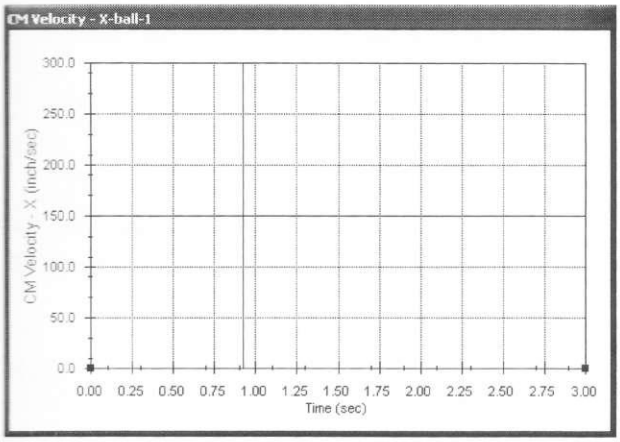


Figure 2-26 *X*-Velocity of the Ball CM

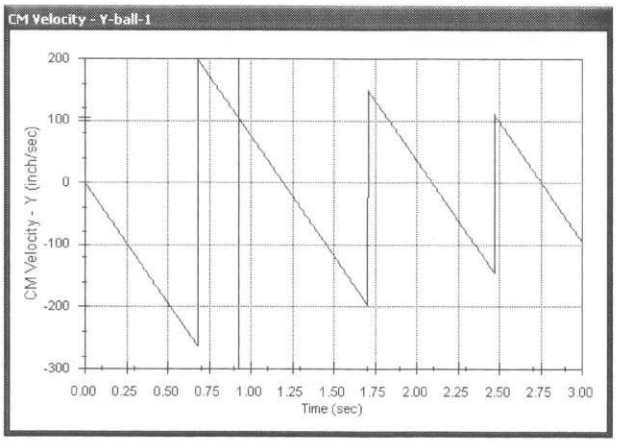
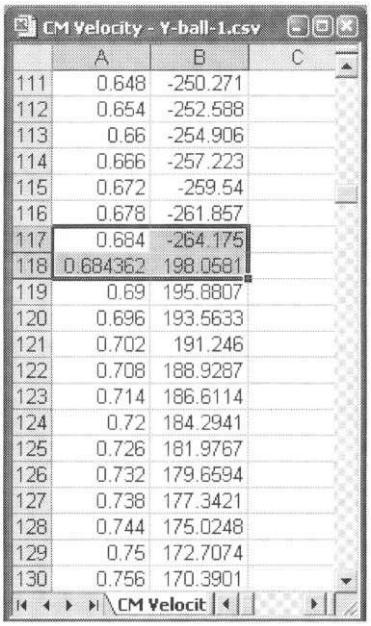


Figure 2-27 *Y*-Velocity of the Ball CM

Add two more graphs for the ball, the *X*-velocity and *Y*-velocity. As shown in Figure 2-26, the *X*-velocity of the ball is a constant *150* in/sec, which is equal to the initial velocity. This is because we

turned off all friction for the *3D Contact* constraint earlier, therefore, no energy loss due to contact. Figure 2-27 shows 7-velocity of the ball. The ball moves at a linear velocity due to gravity. The 7-velocity is about *-263* in/sec at *t = 0.67* seconds. You may see this data by moving the cursor close to the corner point of the curve and leave the cursor for a short period. The data will appear.

You may also convert the *XY Plot* data to Microsoft® Excel spreadsheet by simply moving the cursor inside the graph and right click to choose *Export CSV.* Open the spreadsheet to see more detailed simulation data. From the spreadsheet (Figure 2-28), the 7-velocity right before and after the ball hits the ground are *-264* and *198* in/sec, respectively. The ratio of *198/264* is about *0.75,* which is the coefficient of restitution we defined earlier.


CM Velocity - Y-ball-1.csv

	A	B	C
111	0.648	-250.271	
112	0.654	-252.588	
113	0.66	-254.906	
114	0.666	-257.223	
115	0.672	-259.54	
116	0.678	-261.857	
117	0.684	-264.175	
118	0.684362	198.0581	
119	0.69	195.8807	
120	0.696	193.5633	
121	0.702	191.246	
122	0.708	188.9287	
123	0.714	186.6114	
124	0.72	184.2941	
125	0.726	181.9767	
126	0.732	179.6594	
127	0.738	177.3421	
128	0.744	175.0248	
129	0.75	172.7074	
130	0.756	170.3901	

CM Velocit

Figure 2-28

In addition, you may use the *Export A VI* button on top of the graphics screen to create an *AVI* movie for the motion animation. In the *Export A VI Animation File* dialog box (Figure 2-29); simply click the *Preview* button to review the *AVI* animation. Leave all default data for *Frame* and *Time.* Click *OK* to accept the definition. An *AVI* file will be created in your current folder with a file name, *Lesson2.avi.* You may play the *A VI* animation using, for example, *Window Media Player.*

Be sure to save your model before exiting from *COSMOSMotion.*

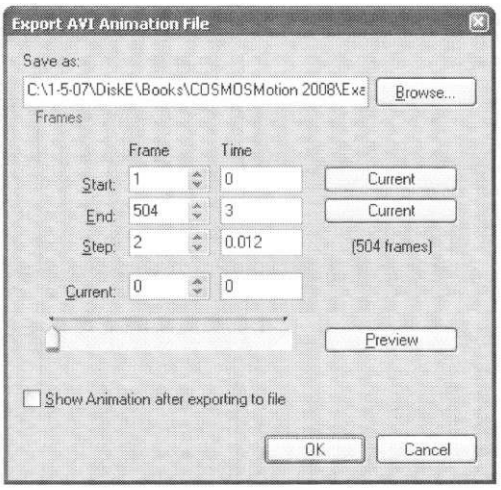


Figure 2-29 Exporting AVI Animation

2.4 Result Verifications

In this section, we will verify analysis results obtained from *COSMOSMotion* using particle dynamics theory you learned in high school *Physics.*

There are two assumptions that we have to make in order to apply the particle dynamics theory to this ball-throwing problem:

(i) The ball is of a concentrated mass, and
(ii) No air friction is present.

Equation of Motion

It is well-known that the equations that describe the position and velocity of the ball are, respectively,

$$P_x = P_{0_x} + V_{0_x} t \tag{2.1a}$$

$$P_y = P_{0_y} + V_{0_y} t - \frac{1}{2} g t^2 \tag{2.1b}$$

and

$$V_x = V_{0_x} \tag{2.2a}$$

$$V_y = V_{0_y} - gt \tag{2.2b}$$

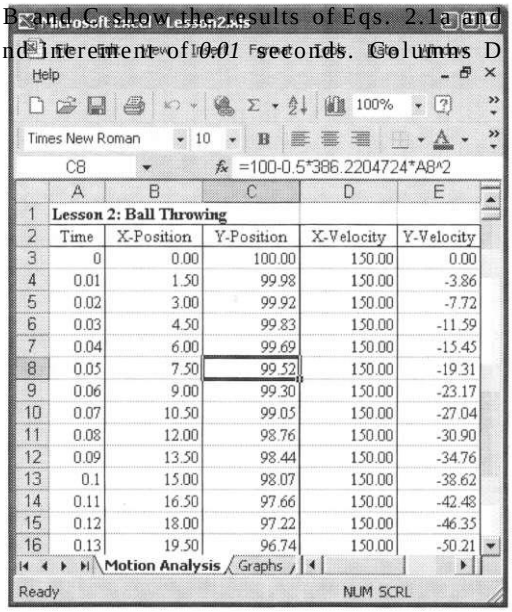
where P_x and P_y are the X- and Y-positions of the ball, respectively; V_x and V_y are the X- and Y-velocities, respectively; P_{0x} and P_{0y} are the initial positions in the X- and Y-directions, respectively; V_{0x} and V_{0y} are the initial velocities in the X- and Y-directions, respectively; and g is the gravitational acceleration.

where P_x and P_y are the X- and 7-positions of the ball, respectively; V_x and V_y are the X- and 7-velocities, respectively; P_{0x} and P_0y are the initial positions in the X- and 7-directions, respectively; V_{0x} and Vg_y are the initial velocities in the X- and 7-directions, respectively; and g is the gravitational acceleration.

These equations can be implemented using, for example, Microsoft® Excel spreadsheet shown in Figure 2-30, for numerical solutions. In Figure 2-30, Columns B and C show the results of Eqs. 2.1a and 2.1b, respectively, with a time interval from *0* to *3* seconds and increment of *0.01* seconds. Columns D and E show the results of Eqs. 2.2a and 2.2b, respectively.

Note that when the ball hits the ground we will have to reset the velocity to 75% of that at the prior time step and in the opposite direction.

Data in column C is graphed in Figure 2-31. Columns D and E are graphed in Figure 2-32. Comparing Figure 2-31 with Figure 2-25 and Figure 2-32 with Figures 2-26 and 2-27, the results obtained from theory and *COSMOSMotion* are very close, which means the dynamic model has been created properly in *JSMOSMotion*, and *COSMOSMotion* does its job and gives us good results. Note that the solution spreadsheet am be found at the publisher's website (filename: . *on2.xls).*



	A	B	C	D	E
1	**Lesson 2: Ball Throwing**				
2	Time	X-Position	Y-Position	X-Velocity	Y-Velocity
3	0	0.00	100.00	150.00	0.00
4	0.01	1.50	99.98	150.00	-3.86
5	0.02	3.00	99.92	150.00	-7.72
6	0.03	4.50	99.83	150.00	-11.59
7	0.04	6.00	99.69	150.00	-15.45
8	0.05	7.50	99.52	150.00	-19.31
9	0.06	9.00	99.30	150.00	-23.17
10	0.07	10.50	99.05	150.00	-27.04
11	0.08	12.00	98.76	150.00	-30.90
12	0.09	13.50	98.44	150.00	-34.76
13	0.1	15.00	98.07	150.00	-38.62
14	0.11	16.50	97.66	150.00	-42.48
15	0.12	18.00	97.22	150.00	-46.35
16	0.13	19.50	96.74	150.00	-50.21

Figure 2-30 The Excel Spreadsheet

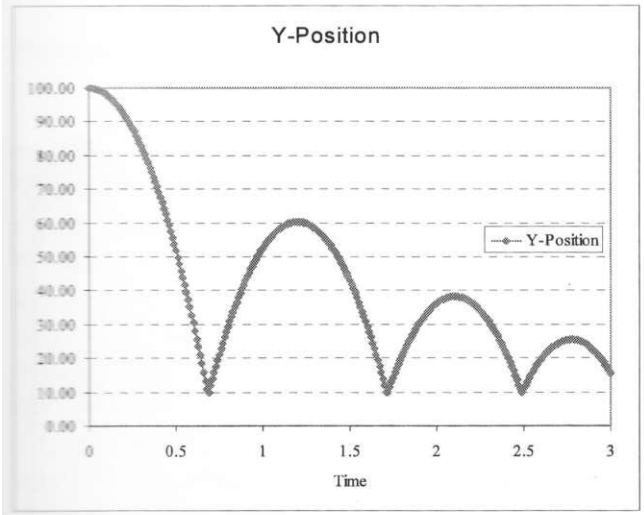


Figure 2-31 Graph of the *Y*-Position of the Ball Obtained from Spreadsheet Calculations

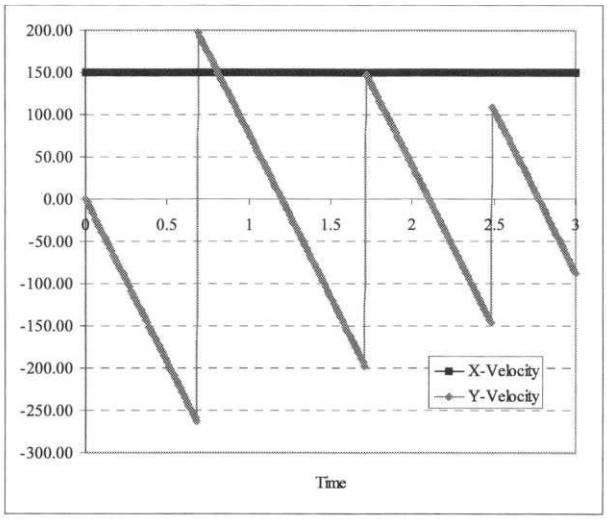


Figure 2-32 Graph of the *X*- and *Y*-Velocities of the Ball Obtained from Spreadsheet Calculations

1. Use the same motion model to conduct a simulation for a different scenario. This time the ball is thrown at an initial velocity of $V_{0_x} = 100$ in/sec and $V_{0_y} = 50$ in/sec from an elevation of 750 in., as shown in Figure E2-1.

   (i) Create a dynamic simulation model using *COSMOSMotion* to simulate the trajectory of the ball. Report position, velocity, and acceleration of the ball at *0.5* seconds in both vertical and horizontal directions obtained from *COSMOSMotion.*

   (ii) Derive and solve the equations that describe the position and velocity of the ball. Compare your solutions with those obtained from *COSMOSMotion.*

   (iii) Calculate the time for the ball to reach the ground and the distance it travels. Compare your calculation with the simulation results obtained from *COSMOSMotion.*

2. A *1 "xl "xl"* block slides from top of a 45° slope (due to gravity) without friction, as shown in Figure E2-2. The material of the block and the slope is *AL2014.*

   (i) Create a dynamic simulation model using *COSMOSMotion* to analyze motion of the block. Report position, velocity, and acceleration of the block in both vertical and horizontal directions at *0.5* seconds obtained from *COSMOSMotion.*

   (ii) Create a *LimitDistance* mate to stop the block when its front lower edge reaches the end of the slope. You may want to review *COSMOSMotion* help menu or preview *Lesson 3* to learn more about the *LimitDistance* mate.

   (iii) Derive and solve the equation of motion for the system. Compare your solutions with those obtained from *COSMOSMotion.*

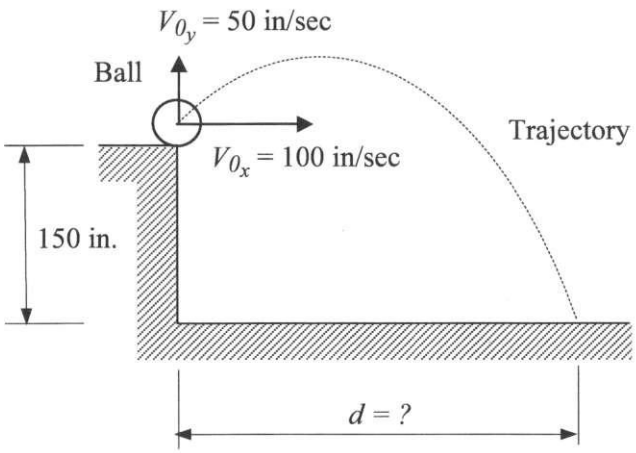


Figure E2-1 The Ball Throwing Problem

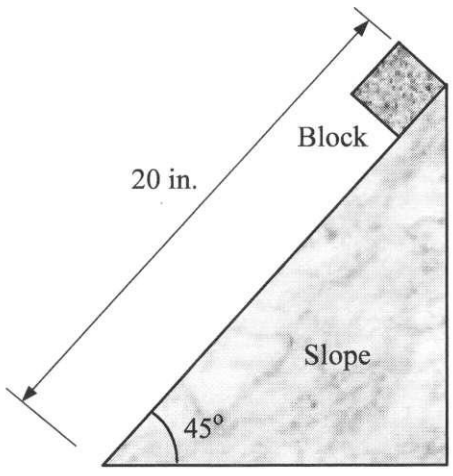
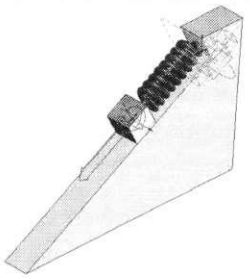


Figure E2-2 The Block Sliding Problem

Lesson 3: A Spring-Mass System



3.1 Overview of the Lesson

In this lesson, we will create a simple spring-mass system and simulate its dynamic responses under various scenarios. A schematics of the system is shown in Figure 3-1, in which a steel block of *V'xV'xl"* is sliding along a *30°* slope with a spring connecting it to the top end of the slope. The block will slide back and forth along the slope under three different scenarios. First, the block will slide due to a small initial displacement, essentially, a free vibration. For the second scenario, we will add a friction between the block and the slope face. Finally, we will remove the friction and add a sinusoidal force $p(t)$, therefore, a forced vibration. Gravity will be turned on for all three scenarios. In this lesson, you will learn how to create the spring-mass model, run a motion analysis, and visualize the analysis results. In addition, you will learn how to add a friction to a joint, in this case, a planar (or coincident) joint. The analysis results of the spring-mass example can be verified using particle dynamics theory. Similar to *Lesson* 2, we will formulate the equation of motion, solve the differential equations, graph positions of the block, and compare our calculations with results obtained from *COSMOSMotion.* Specifically, we will focus on the first and the last scenarios; i.e., free and forced vibrations, respectively.

3.2 The Spring-Mass System

Physical Model

Note that the *IPS* units system will be used for this example. The spring constant and unstretched length (or free length) are $k = 20$ lbf/in. and $U = 3$ in., respectively. As mentioned earlier, the first scenario assumes a free vibration, where the block is stretched *1* in. downward along the *30°* slope. Friction will be imposed for *Scenario* 2, in which the friction coefficient is assumed ***jU*** *= 0.25.* In the third scenario, an external force $p(t) = 10 \cos 360t$ lb, is applied to the block, as shown in Figure 3-1. All three scenarios will assume a gravity of $g = 386$ in/sec^2 in the negative 7-direction. All three scenarios will be simulated using *COSMOSMotion.*

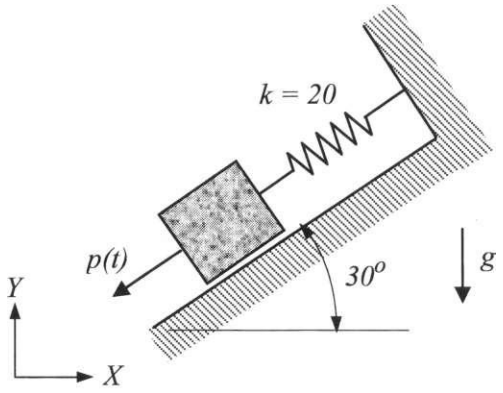


Figure 3-1 The Spring-Mass System

SolidWorks Parts and Assembly

For this lesson, the parts and assembly have been created for you in *SolidWorks.* There are six files created, *block. SLDPRT, ground. SLDPRT, Lesson3. SLDASM, Lesson3Awithresults. SLDASM, Lesson3Bwithresults.SLDASM,* and *Lesson3Cwithresults.SLDASM.* You can find these files at the publisher's web site (http://www.schroffl.com/). We will start with *Lesson3.SLDASM,* in which the block

is assembled to the ground and no motion entities have been added. In addition, the assembly files *Lesson3Awithresults. SLDASM, Lesson3Bwithresults. SLDASM,* and *Lesson3Cwithresults. SLDASM* contain the complete simulation models with simulation results for the three respective scenarios.

In the assembly models, there are three assembly mates, *Coincidentl(ground<l>,block<l>), Coincident3(ground<l>,ball<l>),* and *LimitDistancel(ground<l>,ball<l>),* as shown in Figure 3-2. The block is allowed to move along the slope face. If you choose the *Move Component* button on top of the graphics screen, and drag the block m the graphics screen, you should be able to move the block on the slope face, but not beyond the slope face. This is because the third mate is defined to restrict the block to move between the lower and upper limits. We will take a look at the assembly mate *LimitDistance 1.*

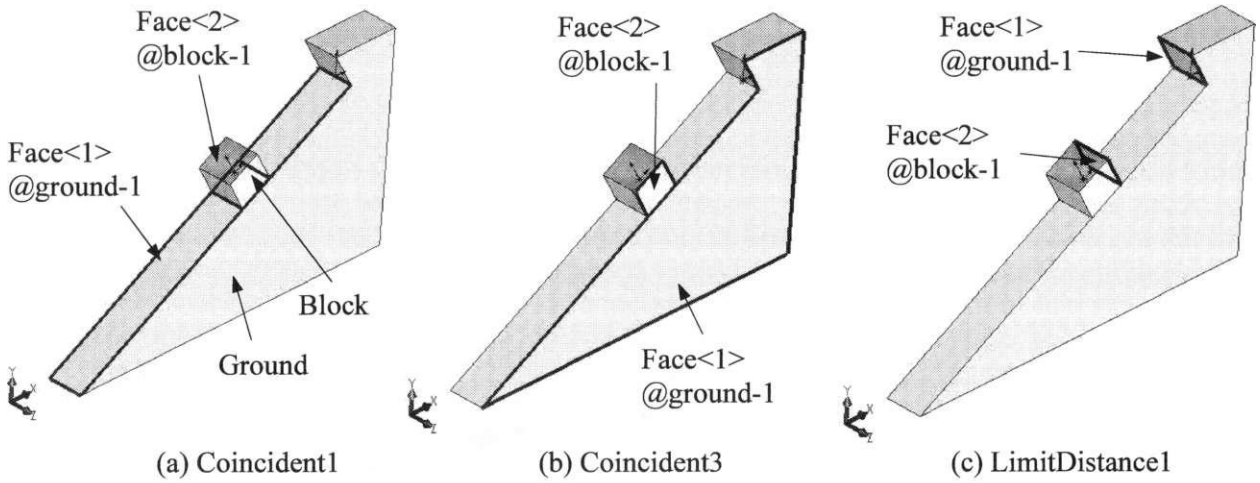


Figure 3-2 Assembly Mates Defined in *Lesson3.SLDASM*

From the browser, righ-click the third mate, *LimitDistance 1,* and choose *Edit Feature.* The mate is brought back for reviewing or editing, as shown in Figure 3-3. Note that the distance between the two faces *(Face<2>@block-l* and *Face<l>@ground-l,* see Figure 3-2c) is *3.00* in., which is the neutral position of the block when the spring is undeformed. The upper and lower limits of the distance are *9.00* and *0.00* in., respectively. The length of the slope face is *10* in., therefore, the upper limit is set to *9.00* in., so that the block will stop when its front lower edge reaches the end of the slope face. Note that you will have to choose *Advanced Mates* in order to access the limit fields.

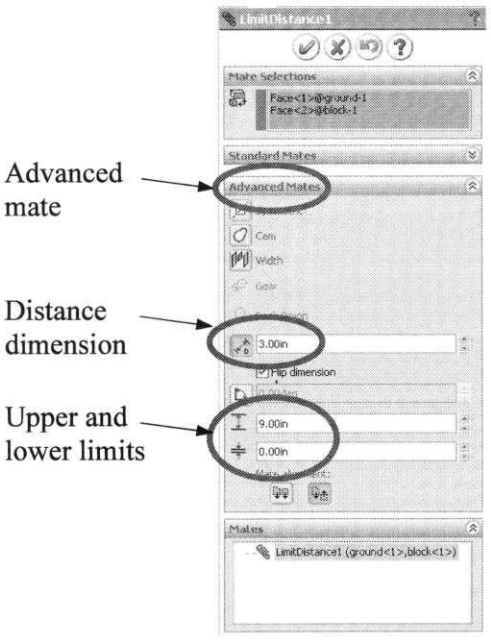


Figure 3-3

Motion Model

A spring with a spring constant $k = 20$ lb$_f$/in. and an unstretched length $U = 3$ in. will be added to connect the block *(Face<2>,* as shown in Figure 3-2c) with the ground *(Face<l>).*

By default, the ends of the spring will connect to the center of the corresponding square faces (see Figure 3-4). No reference points are needed.

This model is adequate to support a free vibration simulation under the first scenario. Note that we will move the block *1* in. downward along the slope face for the simulation. This can be accomplished by changing the distance from *3* to *4* in. in the *LimitDistancel* assembly mate.

Note that before entering *COSMOSMotion* we will suppress two assembly mates, *Coincident^* and *LimitDistancel,* and only keep *Coincident 1* in order for *COSMOSMotion* to impose a planar (or coincident) joint between the block and the slope face. You may unsuppress these mates when you want to make a change to the assembly, for example, moving the block back to its initial position.

As mentioned earlier, a friction force will be added between the block and the slope face for *Scenario* 2. In addition, a sinusoidal force *p(t) = 10 cos 360t* will be added to the block for the 3rd scenario. Gravity will be turned on for all three scenarios.

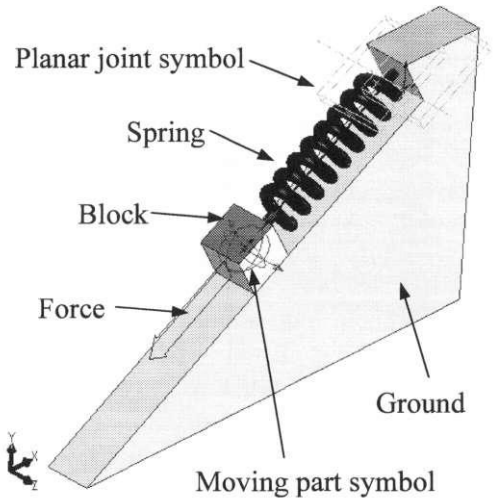


Figure 3-4 Spring Mass Dynamic Model

3.3 Using *COSMOSMotion*

Start *SolidWorks* and open assembly file *Lesson3.SLDASM.*

Before entering *COSMOSMotion,* we will suppress two assembly mates, *Coincident3* and *LimitDistancel.* From the *Assembly* browser, expand the *Mates* branch, right click *Coincident3,* and choose *Suppress.* The mate *Coincident3* will become inactive. Repeat the same to suppress *LimitDistancel.* Save your model.

From the browser, click the *Motion* button on top to enter *COSMOSMotion.*

In this lesson, instead of using *IntelliMotion Builder* (as in *Lesson* 2), we will use the browser, and basic drag-and-drop and right click activated menus to create and simulate the block motion.

Before creating any entities, always check the units system. Similar to *Lesson* 2, choose from the null-down menu

Tools > Options.

Choose the *Document Properties* tab in the *System Options - General* dialog box, click the *Units* node. You should see that the *IPS* units system has been chosen. In this units system, the gravity is *386* in/sec^2 in the negative 7-direction of the global coordinate system by default. No change is needed.

Defining Bodies

From the browser, expand the *Assembly Components* branch (right underneath the *Motion Model* node) by clicking the small + button in front of it. You should see two parts listed, *block-1* and *ground-7,*

as shown in Figure 3-5. Also expand the *Parts* branch; you should see *Moving Parts* and *Ground Parts* listed. We will move *block-1* to *Moving Parts* and *ground-1* to *Ground Parts* by using the drag-and-drop method.

Click *block-1* and drag it by holding down the left-mouse button, moving the mouse until the cursor is over the *Moving Parts* node, and then releasing the mouse button. Part *block-1* is now added to the motion model as moving parts. Repeat the same steps to move *ground-1* to the *Ground Parts* node. Now, the part *ground-1* is added to the motion model as a ground part (completely fixed, as shown in Figure 3-6).

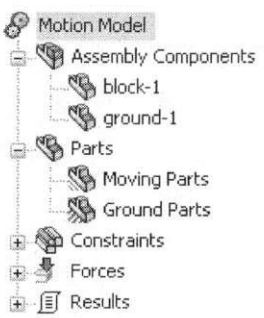


Figure 3-5

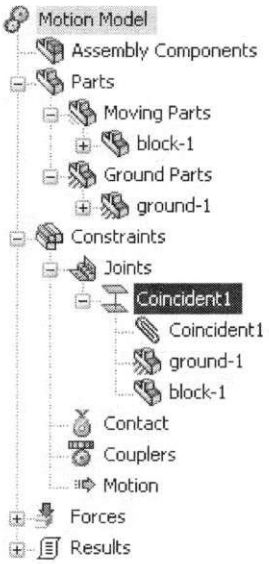


Figure 3-6

Expand the *Constraints* branch, and then the *Joints* branch. You should see that only one joint, *Coincident1,* is listed. Expand the *Coincident!* branch to see that the assembly mate is defined between *ground-!* and *block-1.* Since this coincident joint restricts the block to move on the slope face of the ground, it is therefore a *planar* joint. A planar joint symbol should appear in the motion model, as shown in Figure 3-4.

Defining Spring

From the browser, right click the *Spring* node and choose *Add Translational Spring* (see Figure 3-7). In the *Insert Spring* dialog box (Figure 3-8), the *Select 1st Component* field is highlighted in red and ready for you to pick. Pick the face at top right of the ground (see Figure 3-9), the *Select 2nd Component* field should now highlight in red, and *ground l/DDMFace4* should appear in the *Select Point on 1st Component* field, which indicates that the spring will be connected to the center point of the face.

Rotate the view, and then pick the face in the block, as shown in Figure 3-9. Now, *block-1* and *block-l/DDMFace3* should appear in the *Select 2nd Component* field and *Select Point on 2nd Component* field, respectively. Also, a spring symbol should appear in the graphics screen, connecting the center points of the two selected faces.

Enter the followings:

Stiffness: 20
Length: 3 (Note that you have to deselect the *Design* box to the right before entering this value, as shown in Figure 3-8)
Force: 0
Coil Diameter: 0.75
Number of coils: 8
Wire Diameter: 0.25

Click *Apply* to accept the definition and close the *Insert Spring* dialog box.

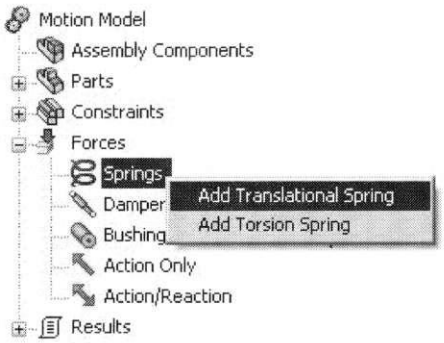


Figure 3-7

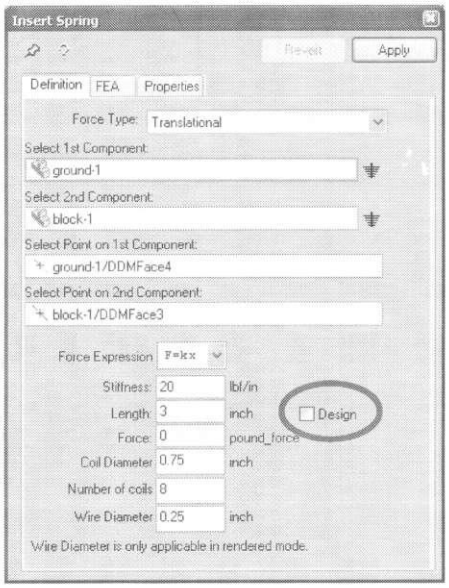


Figure 3-8 The *Insert Spring* Dialog Box

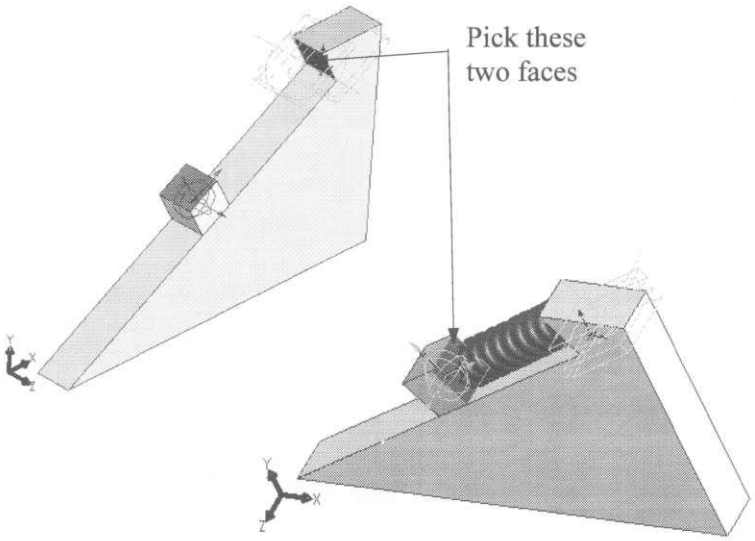


Figure 3-9 Picking Two Faces to Define the Spring

Defining Initial Position

We would like to stretch the spring *1* in. downward along the slope face as the initial position for the block. The block will be released from this position to simulate a free vibration; i.e., the first scenario. We will go back to the *Assembly* mode, unsuppress *LimitDistancel,* change the distance dimension from *3* to *4* in., and then sunnress the mate before returning to *COSMOSMotion.*

Go back to *Assembly* by clicking the *Assembly* button on top of the browser.

Expand the *Mates* branch listed in the browser, right click *LimitDistance1(ground<1>,ball<1>),* and choose *Unsuppress.* Right click the same assembly mate and choose *Edit Feature.* You should see the definition of the assembly mate in the dialog box like that of Figure 3-3. Change the distance from *3.00* to *4.00* in., and click the checkmark on top to accept the change. In the graphics screen, the block should move *1* in. downward along the slope face. Click the checkmark again to close the assembly mate box. Right click*LimitDistancel(zround<l>,ball<l>)* again and choose *Suppress.*

Go back to *COSMOSMotion* by clicking the *Motion* button *(y* . Click the *Motion Model* node, press the right mouse button and select *Simulation Parameters.* Enter *0.25* for simulation duration and *500* for the number of frames.

Click the *Motion Model* node again, press the right mouse button and select *Run Simulation.* You may also click the *Run Simulation* button |J|] right below the browser to run a simulation. You should see the block start moving back and forth along the slope face. We will graph the position of the block in terms of the magnitude (instead of X- or 7-component) next.

Displaying Simulation Results

Since there is no position graph defined for the block, we will have to create one. We will create a graph for the distance between the two faces that were selected to define the spring.

From the browser, expand the *Results* branch, right click the *Linear Disp* node, and choose *Create Linear Displacement* (Figure 3-10). In the *Insert Linear Displacement* dialog box (Figure 3-11), the *Select First Component* field should be highlighted in red and ready for you to pick. This dialog box is very similar to the upper half of the *Insert Spring* dialog box (Figure 3-8). We will select exactly the same two faces shown in Figure 3-9 for this displacement.

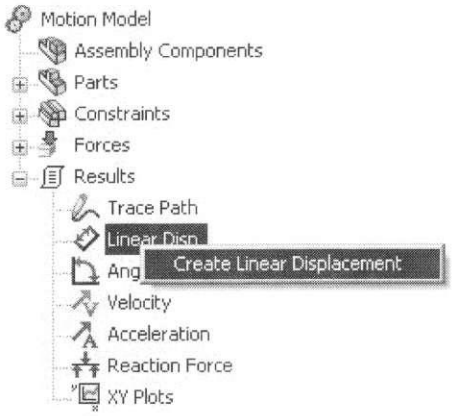


Figure 3-10

Similar to the spring, pick the face of the ground (see Figure 3-9). Rotate the view, and then pick the face in the block, as shown in Figure 3-9. A straight line that connects the center points of these two faces appears, as shown in Figure 3-12. Click *Apply* button to accept the definition.

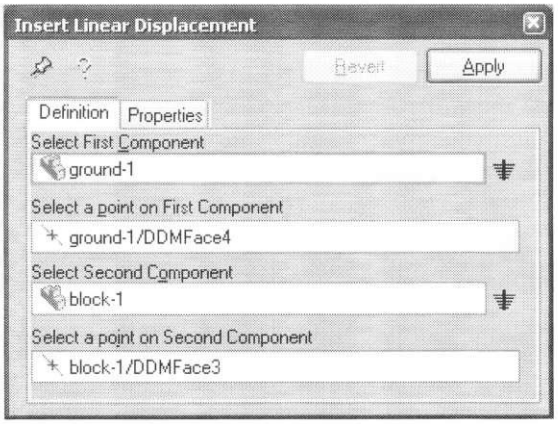


Figure 3-11 The *Insert Linear Displacement* Dialog Box

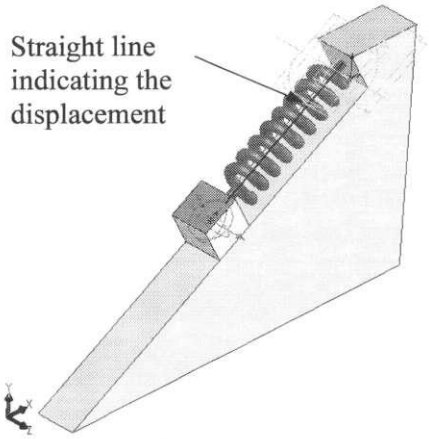


Figure 3-12 The Straight Line Showing the Linear Displacement Measure

Next, we will create a graph for the displacement of the block using the *XY Plots* option. This position graph should reveal a sinusoidal function as we have seen in many vibration examples of *Physics.*

From the browser, right click *LDisplacement* > *Plot* > *Magnitude* (see Figure 3-13). A graph like that of Figure 3-14 should appear. From the graph, the *block* moves along the slope face between 2 and *4* in. This is because the unstretched length of the spring is *3* in. and we stretched the spring *1* in. to start the motion. Also, it takes about *0.04* seconds to complete a cycle, which is small. The small vibration period can be attributed to the fact that the spring is fairly stiff *(20* lb_f/in).

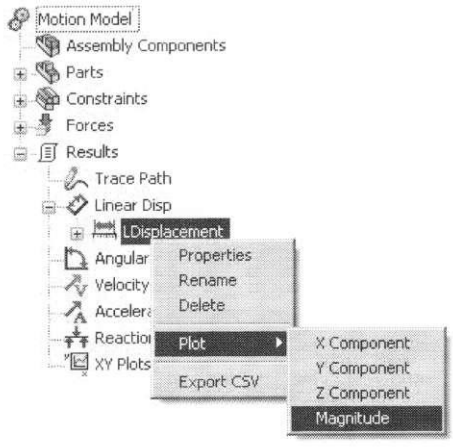


Figure 3-13

Note that you can also export the graph data, for example, by right clicking the graph and choosing *Export CSV.* Open the spreadsheet and exam the data. The time for the block to move back to its initial position; i.e., when the distance is *4* in., is *0.037* seconds.

We will carry out calculations to verify these results later in Section 3.4. Before we do that, we will work on two more scenarios: with friction and with the addition of an external force. Save your model before moving to the next scenario. You may save the model under different name and use it for the next scenario.

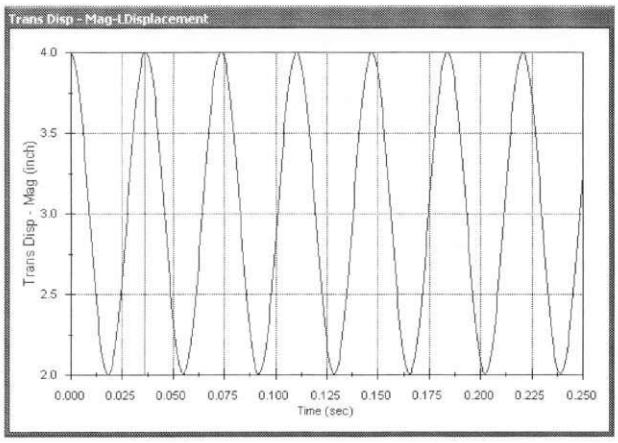


Figure 3-14 The Displacement Graph: *Scenario 1*

Scenario 2: With Friction

We will add a friction force to the planar joint *{Coincident 1)* between the block and the ground. The friction coefficient is *ju = 0.25.* Before making any change to the definition of the simulation model, we will have to delete existing simulation results. Click the *Delete Results* button  at the bottom of the browser to delete the results.

Note that for calculating friction effects, *COSMOSMotion* models a planar joint as one block sliding and rotating on the surface of another block, as illustrated in Figure 3-15, where

L is the length of the top (sliding) rectangular block,
W is the width of the top rectangular block, and
R is the radius of a circle, centered at the center of the top block face in contact with the bottom block, which circumscribes the face of the sliding block.

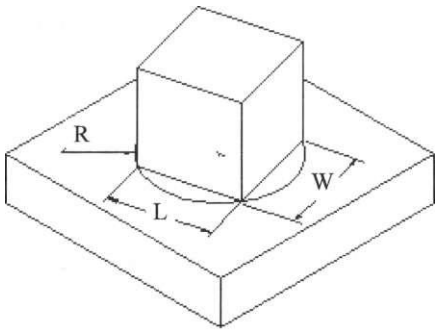


Figure 3-15

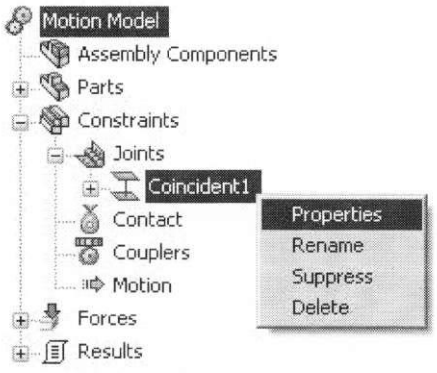


Figure 3-16

Expand the *Constraints* branch and then the *Joints* branch. Right click the *Coincident!* node and choose *Properties.* In the *Edit Mate-Defined Joint* dialog box, choose the *Friction tab,* click the *Use Friction,* enter *0.25* for *Coefficient (mu),* and enter *Joint dimensions, Length: 1, Width: 1,* and *Radius: 1.414,* as shown in Figure 3-17. Click *Apply* button to accept the definition.

Run a simulation (with the same simulation parameters as those of *Scenario T).* Graph the displacement of the block; you should see a graph similar to that of Figure 3-18. The amplitude of the graph (that is, the distance the block travels) is decreasing over time due to friction. Save your model. We will move into *Scenario 3.* You may save the model again under different name and use it for the next scenario.

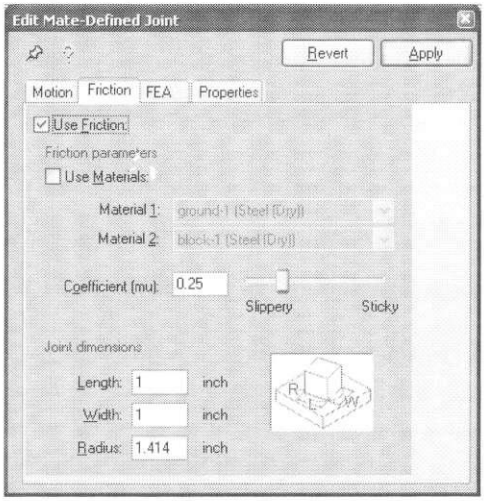


Figure 3-17 Defining Friction for the Planar Joint

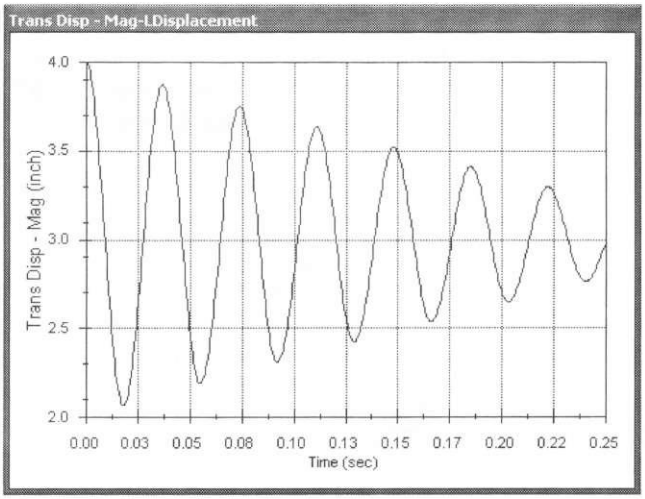


Figure 3-18 The Displacement Graph: *Scenario 2*

In this scenario we will add an external force *p(t) = 10 cos 360t* at the center of the end face of the *block* in the downward direction along the slope. At the same time, we will remove the friction in order to simplify the problem.

Before creating a force, we will delete the simulation results and remove the friction. Delete the results by clicking the *Delete Results* button  at the bottom of the browser.

Right click the *Coincidentl* node and choose *Properties.* In the *Edit Mate-Defined Joint* dialog box, choose the *Friction* tab, and deselect the *Use Friction* by clicking the small box in front of it. All parameters and selections on the dialog box should become inactive. Click *Apply* button to accept the change.

The force can be added by expanding the *Forces* branch, right clicking the *Action Only* node, and choosing *Add Action-Only Force,* as shown in Figure 3-19. In the *Insert Action-Only Force* dialog box (see Figure 3-20), the *Select Component to which Force is Applied* field will be active (highlighted in red) and ready for you pick the component.

Pick the end face of the block, as shown in Figure 3-21. The part *block-1* is now listed in the *Select Component to which Force is Applied* field, and *block-l/DDMFace8* is listed in both the *Select Location* and the *Select Direction* fields. That is, the force will be applied to the center of the selected end face and in the direction that is normal to the selected face.

Now in the *Insert Action-Only Force* dialog box, the *Select Reference Component to orient Force* field is active (highlighted in red) and is ready for selection. We will pick the ground part for reference. Pick any place in the ground part, *ground-1* will now appear in the *Select Reference Component to orient Force* field.

Figure 3-19

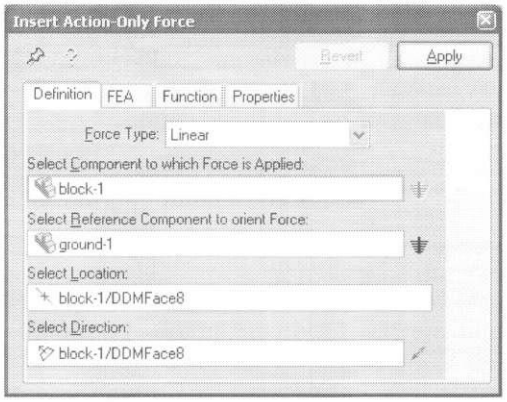


Figure 3-20 The *Insert Action-Only Force* Dialog Box

Click the *Function* tab (see Figure 3-22), choose *Harmonic,* and enter the followings:

Amplitude: 10
Frequency: 360
Phase Shift: -90

Note that the *-90* degrees entered for *Phase Shift* is to convert a sine function (default) to the desired cosine function.

Click the graph button (right most and circled in Figure 3-22); the sinusoidal force function will appear like the one in Figure 3-23. This is indeed the cosine function $p(t) = 10 \cos 360t$ we wanted to define.

Close the graph and click *Apply* button to accept the force definition. You should see a force symbol added to the block, as shown in Figure 3-4.

From the browser, right click the *Motion Model* node, and choose *Simulation Parameters.* Change the simulation duration to *0.5* seconds (in order to see a graph later that covers a larger time span). Note that the 0.5-second duration is half the harmonic function period of the force applied to the block.

From the browser, right click the *Motion Model* node again, and choose *Run Simulation.* After *2* to *3* seconds, the block starts moving.

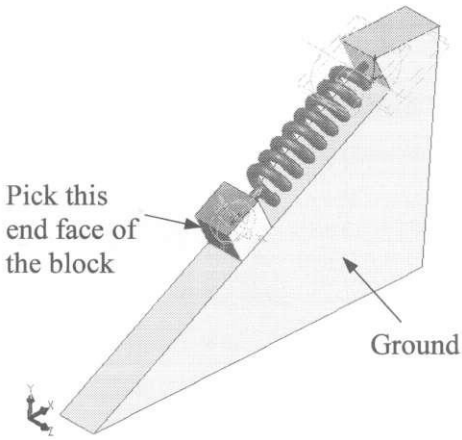


Figure 3-21 Pick Face to Defining Location and Direction of the Force

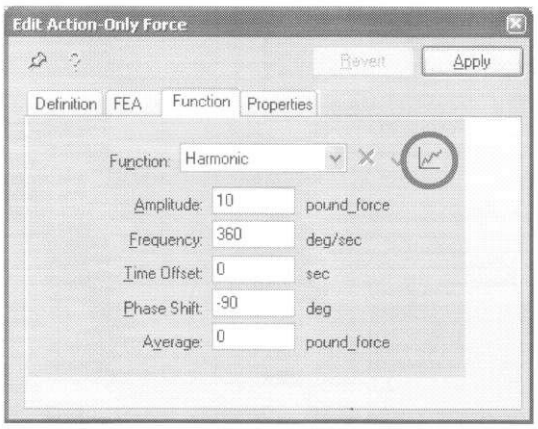


Figure 3-22 Defining Force Function

Note that in some occasions, the block may slide out of the slope face during the simulation, as shown in Figure 3-24. When this happens, simply unsuppress the assembly mate; for example, *Coincident3,* to restricts the block to stay on the slope face.

After unsuppressing *Coincident3,* the planar joint will become a translational joint (converted by *COSMOSMotion)* composed of two assembly mates, *Coincident 1* and *Coincident^*. Please refer to the assembly file *Lesson3Cwithresults.SLDASM* for the translational joint employed for this example.

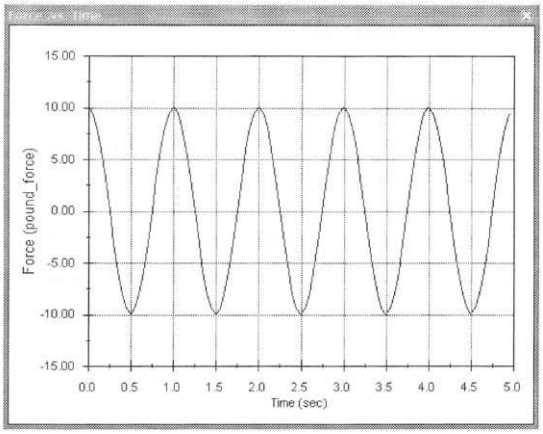


Figure 3-23 The Force Function Graph

Rerun a simulation if necessary. As soon as the simulation is completed, a graph like that of Figure 3-25 should appear. From the graph, the *block* moves along the slope face roughly for *2* in. back and forth (since friction is turned off). The vibration amplitude is enveloped by a cosine function due to the external force *p(t).* Also, it takes about *0.04* seconds to complete a cycle, which is unchanged from the previous case. We will carry out calculations to verify these results later. Save your model.

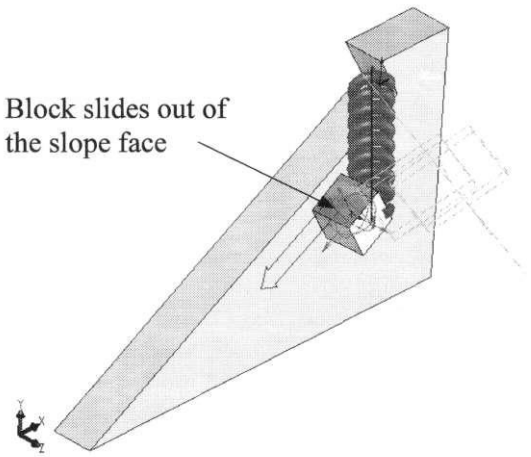


Figure 3-24 Block Slides Out of the Slope Face

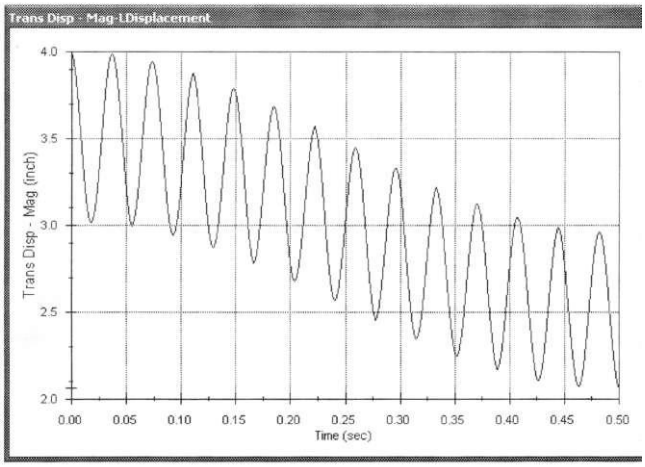


Figure 3-25 The Displacement Graph: *Scenario 3*

3.4 Result Verifications

In this section, we will verify analysis results of *Scenarios 1* and *3* obtained from *COSMOSMotion.* We will assume that the *block* is of a concentrated mass so that the particle dynamics theory is applicable.

We will start with *Scenario 1* (i.e., free vibration with gravity), and then solve the equations of motion for *Scenario 3* (forced vibration, no friction).

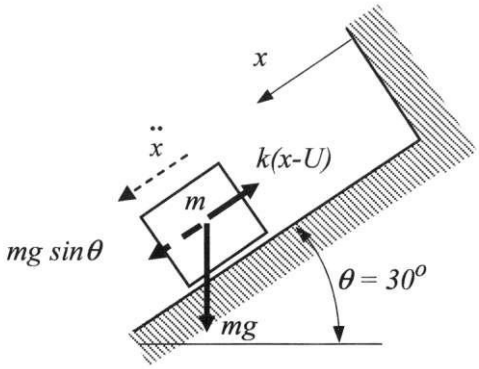


Figure 3-26 The Free-Body Diagram

Equation of Motion: Scenario 1

From the free-body diagram shown in Figure 3-26, applying Newton's Second Law and force equilibrium along the X-direction (i.e., along the 30° slope), we have

$$\sum F_x = mg\sin\theta - k(x-U) = m\ddot{x} \tag{3.1}$$

Therefore,

$$m\ddot{x} + k(x-U) = mg\sin\theta \tag{3.2}$$

where m is the mass of the block, U is the unstretched length of the spring, x is the distance between the mass center of the block and the top right end of the slope, measured from the top right end.

The double dots on top of x represent the second derivative of x with respect to time. Rearrange Eq. 3.2, we have

$$mx + kx = mg\sin 0 + Uk \tag{3.3}$$

where both terms on the right are time-independent.

This is a second-order ordinary differential equation. It is well known that the general solution of the differential equation is

$$x_g = A_1 \cos\omega_n t + A_2 \sin\omega_n t \tag{3.4}$$

where $\omega_n = \sqrt{\frac{k}{m}}$, and A_1 and A_2 are constants to be determined with initial conditions. Note that the mass of the steel block is *0.264* lb_m. This can be obtained from *SolidWorks* by opening the block part file, and choosing, from the pull-down menu, *Tools* > *Mass Properties.* From the *Mass Properties* dialog box (Figure 3-27), the mass of the block is *0.264* pounds (pound-mass, lb_m). Note that there are *2* decimal points set in *SolidWorks* by default. You may increase it through the *Document Properties - Units* dialog box (choose from pull-down menu, *Tools* > *Options).*

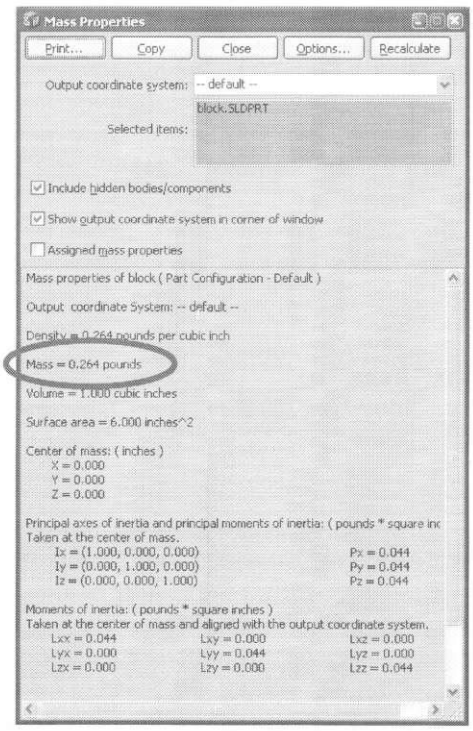


Figure 3-27 The *Mass Properties* Dialog Box

Note that the pound-mass unit lb_m is not as common as slug that we are more familiar with. The corresponding force unit of lb_m is lb_m in/sec^2 according to Newton's Second Law.

Also, since a *1*-lb_m mass block weighs *1* lb_f on earth; therefore, *1* lb_f = *386* lb_m in/sec^2. The stiffness we employed for the spring becomes k = *20* lb_f/in = *20×386* lb_m in/sec^2. Hence, $\omega_n = \sqrt{\frac{k}{m}} = \sqrt{\frac{20\times 386}{0.264}} = 171.0$ rad/sec = *9,798* degrees/sec, and the natural frequency of the system is $f_n = \omega_n/2\pi = 27.2$ Hz. The period for a complete vibration cycle is $T = 1/f_n = 0.0367$ seconds, which is very close to what was obtained by reviewing the graph (for example, graph shown in Figure 3-13) and spreadsheet data converted from the graph. For more details regarding the force and mass in English units system, please refer to Appendix B. Appendix B should clarify some of the confusion you might have in lb_m and lb_f.

The particular solution of Eq. 3.3 is

$$x_p = \frac{mg\ sin\ \theta}{k} + U \tag{3.5}$$

Therefore the total solution is

$$x = x_g + x_p = A_1 \cos \omega_n t + A_2 \sin \omega_n t + \frac{mg \sin \theta}{k} + U \tag{3.6}$$

The initial conditions for the spring-mass system are $x(0) = x_0 = 4$ in., and $\dot{x}(0) = 0$ in/sec. Plugging the initial conditions into Eq. 3.6 (you will have to take the derivative of Eq. 3.6 for $\dot{x}(0)$), we have

$$A_1 = x_0 - \left(\frac{mg \sin \theta}{k} + U \right), \text{ and}$$

$$A_2 = 0.$$

Hence, the overall solution is

$$x = \left(x_0 - \frac{mg \sin \theta}{k} - U \right) \cos \omega_n t + \frac{mg \sin \theta}{k} + U \tag{3.7}$$

Equation 3.7 can be implemented into Microsoft® Excel spreadsheet, as shown in Figure 3-28. Column B in the spreadsheet shows the results of Eq. 3.7, which is graphed in Figure 3-29. Comparing Figure 3-29 with Figure 3-14, the results obtained from theory and *COSMOSMotion* agree very well, which means the motion model has been properly defined, and *COSMOSMotion* gives us good results.

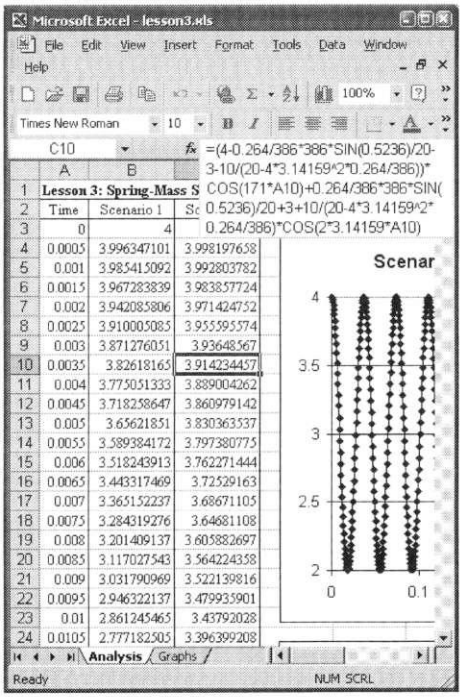


Figure 3-28 The Excel Spreadsheet

Equation of Motion: Scenario 3

Refer to the free-body diagram shown in Figure 3-26 again. For *Scenario 3* we must include the force $p = fo \cos (cot)$ along the X-direction for force equilibrium; i.e.,

$$m\ddot{x} + k(x - U) = mg \sin\theta + f_0 \cos(\omega t) \tag{3.8}$$

where $f_0 = 10$ lb_f and $\omega = 360$ degree/sec$= 2\pi$ rad/sec. Rearrange Eq. 3.8, we have

$$m\ddot{x} + kx = mg \sin\theta + Uk + f_0 \cos(\omega t) \tag{3.9}$$

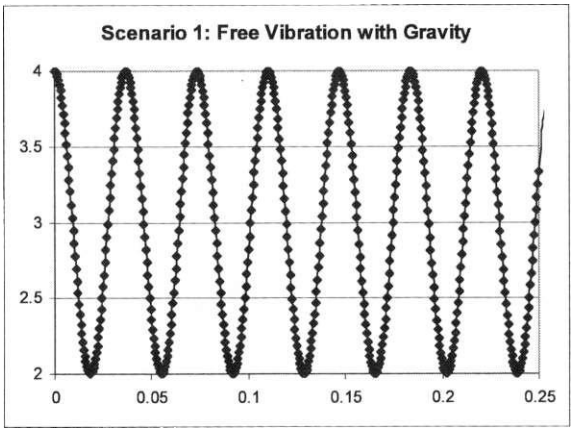


Figure 3-29 Solution from Theory: *Scenario 1*

where the right-hand side consists of constant and time-dependent terms. For the constant terms, the particular solution is identical to that of *Scenario 1*\ i.e., Eq. 3.5. For the time-dependent term, $p = f_0 \cos(\omega t)$, the particular solution is

$$x_{p_2} = \frac{f_0}{k - \omega^2 m} \cos \omega t \tag{3.10}$$

Therefore, the overall solution of Eq. 3.9 becomes

$$x = x_g + x_p + x_{p_2} = A_1 \cos \omega_n t + A_2 \sin \omega_n t + \frac{mg \sin \theta}{k} + U + \frac{f_0}{k - \omega^2 m} \cos \omega t \tag{3.11}$$

Plugging the initial conditions into the solution, we have

$$A_1 = x_0 - \left(\frac{mg \sin \theta}{k} + U + \frac{f_0}{k - \omega^2 m} \right), \text{ and } A_2 = 0.$$

Hence, the complete solution is

$$\begin{aligned} x &= \left(x_0 - \frac{mg \sin \theta}{k} - U - \frac{f_0}{k - \omega^2 m} \right) \cos \omega_n t + \frac{mg \sin \theta}{k} + U + \frac{f_0}{k - \omega^2 m} \cos \omega t \\ &= \left[\left(x_0 - \frac{mg \sin \theta}{k} - U \right) \cos \omega_n t + \frac{mg \sin \theta}{k} + U \right] + \frac{f_0}{k - \omega^2 m} \left(\cos \omega t - \cos \omega_n t \right) \end{aligned} \tag{3.12}$$

Note that terms grouped in the first bracket of Eq. 3.12 are identical to those of Eq. 3.7; i.e., *Scenario 1*. The second term of Eq. 3.12 graphed in Figure 3-30 represents the contribution of the external force *p(t)* to the block motion. The graph shows that the amplitude of the block is kept within *1* in., but the position of the block varies in time. The vibration amplitude is enveloped by a cosine function.

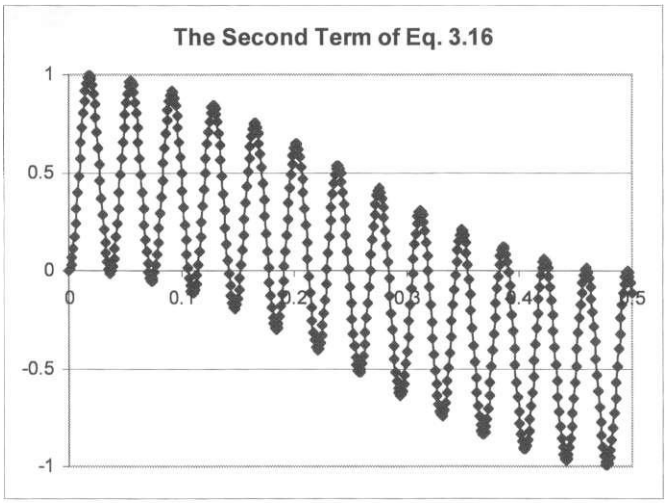


Figure 3-30 Graph of the Second Term of Eq. 3.16

The overall solution of *Scenario 3;* i.e., Eq. 3.12, is a combination of graphs shown in Figures 3-29 and 3-30. In fact, Eq. 3.12 has been implemented in Column C of the spreadsheet. The data are graphed in Figure 3-31.

Comparing Figure 3-31 with Figure 3-25, the results obtained from theory and *COSMOSMotion* are very close. Note that the spreadsheet shown in Figure 3-28 can be found at the publisher's website (filename: *lesson3.xls).*

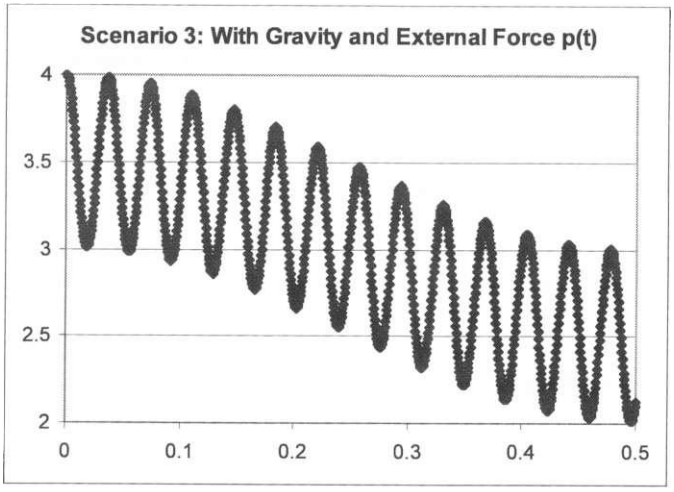


Figure 3-31 Solution from Theory: *Scenario 3*

Exercises:

1. Show that Eq. 3.12 is the correct solution of *Scenario 3* governed by Eq. 3.8 by simply plugging Eq. 3.12 into Eq. 3.8.

2. Repeat the *Scenario 3* of this lesson, except changing the external force to $p(t) = 10 \cos 9798.0t$ lb_f. Will this external force change the vibration amplitude of the system? Can you simulate this resonance scenario in *COSMOSMotion!*

3. Add a damper with damping coefficient $C = 0.01$ lb_f sec/in. and repeat the *Scenario 1* simulation using *COSMOSMotion.*

   (i) Calculate the natural frequency of the system and compare your calculation with that of *COSMOSMotion.*

   (ii) Derive and solve the equations that describe the position and velocity of the mass. Compare your solutions with those obtained from *COSMOSMotion.*

Notes:

Lesson 4: A Simple Pendulum

4.1 Overview of the Lesson

In this lesson, we will create a simple pendulum motion model using *COSMOSMotion.* The pendulum will be released from a position slightly off the vertical line. The pendulum will then rotate freely due to gravity. In this lesson, you will learn how to create the pendulum motion model, run a dynamic analysis, and visualize the analysis results. The dynamic analysis results of the simple pendulum example can be verified using particle dynamics theory. Similar to *Lessons 2* and 3, we will formulate the equation of motion; calculate the angular position, velocity, and acceleration of the pendulum; anc compare our calculations with results obtained from *COSMOSMotion.*

4.2 The Simple Pendulum Example

Physical Model

The physical model of the pendulum is composed of a sphere and a rod rigidly connected, as shown in Figure 4-1. The radius of the sphere is *10* mm. The length and radius of the thin rod are *90* mm and *0.5* mm, respectively. The top of the rod will be connected to the wall with a revolute joint. This revolute joint allows the pendulum to rotate. Both rod and sphere are made of Alunimum. Note that from the *SolidWorks* material library the *Aluminum Alloy 2014* has been selected for both sphere and rod. The *MMGS* units system is selected for this example (millimeter for length, Newton for force, and second for time). Note that in the *MMGS* units system, the gravitational acceleration is *9,806* mm/sec^2.

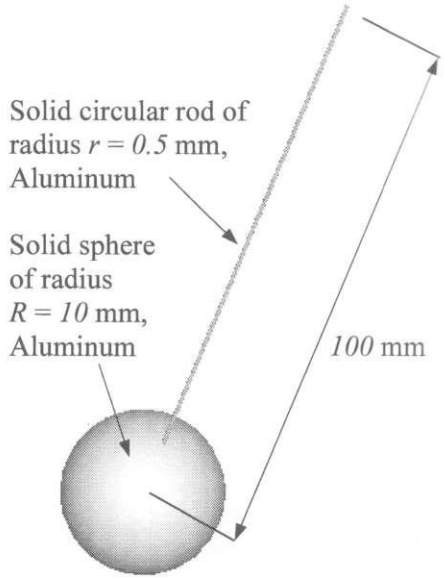


Figure 4-1 The Pendulum Physical Model

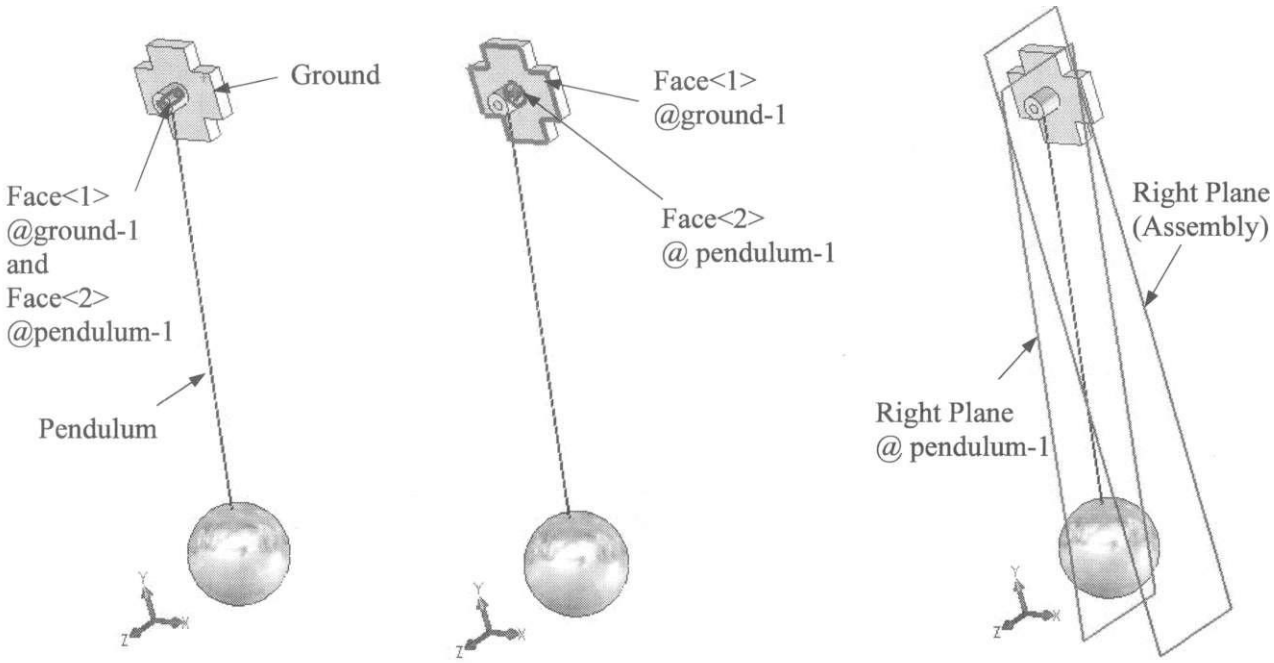
The pendulum will be released from an angular position of *10* degrees measured from the vertical position about the rotational axis of the revolute joint. The rotation angle is intentionally kept small so that the particle dynamics theory can be applied to verify the simulation result.

SolidWorks Parts and Assembly

In this lesson, *SolidWorks* parts of the pendulum example have been created for you. There are four files created, *pendulum.SLDPRT*, *ground.SLDPRT*, *Lesson4.SLDASM*, and *Lesson4withresults.SLDASM.*

You can find these files at the publisher's web site (http://www.schroffl.com/). We will start with *Lesson4.SLDASM,* in which the pendulum is fully assembled to the ground. In addition, the assembly file *Lesson4withresults.SLDASM* consists of a complete simulation model with simulation results.

In the assembly models, there are three assembly mates, *Concentric1(ground<1>,pendulum<1>)*, *Coincident1(ground<1>, pendulum <1>)*, and *Angle1(pendulum <1>, Right Plane)*, as shown in Figure 4-2. Note that the first two mates restrict pendulum to rotate about the *Z*-axis of the global coordinate system. The third mate, *Angle1*, rotates the pendulum a negative *10* degree angle about the *Z*-axis. This mate is required to position the pendulum for an initial condition. You may want to use the *Front* view (one of *SolidWorks* predefined views located on top of the graphics screen) to see how the pendulum is oriented. The pendulum is fully constrained, no movement is allowed. We will suppress the third mate *Angle1* before entering *COSMOSMotion*.



(a) Concentric1 (b) Coincident1 (c) Angle1

Figure 4-2 Assembly Mates Defined in *Lesson4.SLDASM*

Motion Model

In this motion model, the only moveable body is the pendulum. After suppressing the mate *Angle1*, a revolute joint will be added to the motion model between the pendulum and the ground, as shown in Figure 4-3. The pendulum is allowed to rotate about the *Z*-axis of the global coordinate system due to gravity and a small initial rotation angle. Note that the gravity is acting in the negative *Y*-direction, which is consistent with the default setting. Friction between the pendulum and the ground is assumed zero.

4.3 Using *COSMOSMotion*

Start *SolidWorks* and open assembly file *Lesson4.SLDASM.*

Similar to *Lesson 3*, we will use the browser, and basic drag-and-drop and right-click methods to create and simulate the pendulum motion in this lesson. Before entering *COSMOSMotion*, we will suppress the third assembly mate, *Angle1*.

From the *Assembly* browser, expand the *Mates* branch, right-click *Angle1*, and choose *Suppress* The mate *Angle1* will become inactive. Save your model.

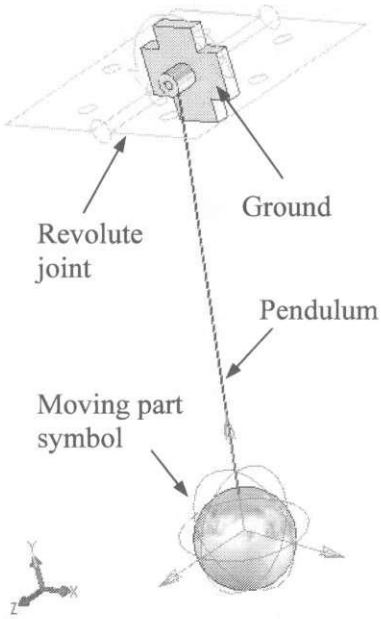


Figure 4-3 Pendulum Motion Model

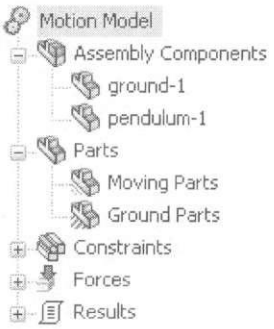


Figure 4-4

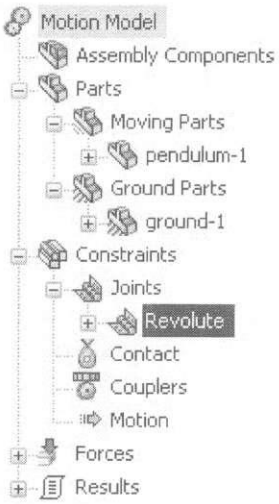


Figure 4-5

From top of the browser, click the *Motion* button to enter *COSMOSMotion.*

From top of the browser, click the *Motion* button $ to enter *COSMOSMotion.*
Before creating any entities, always check the units system. Make sure that *MMGS* is chosen.

Defining Bodies

From the browser, expand the *Assembly Components* branch (right underneath the *Motion Model* node) by clicking the small H button in front of it. You should see two parts listed, *ground-1* and *pendulum-7*, as shown in Figure 4-4.

Also expand the *Parts* branch; you should see *Moving Parts* and *Ground Parts* listed. Go ahead to move *pendulum-1* to *Moving Parts* and *ground-1* to *Ground Parts* by using the drag-and-drop method.

Expand the *Constraints* branch, and then the *Joints* branch. You should see a *Revolute* joint listed, as shown in Figure 4-5 and a revolute joint symbol should appear in the graphics screen (see Figure 4-3).

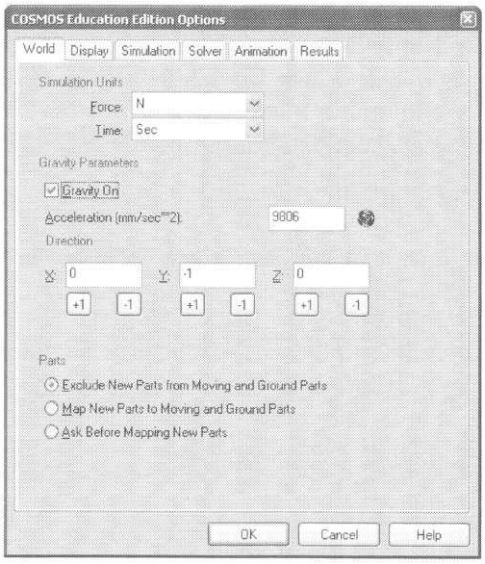


Figure 4-6 Gravity Dialog Box

Setting Gravity

We would like to make sue the gravity is set up properly. From the browser, right-click the *Motion Model* node and select *System Defaults.* In the *Options* dialog box (Figure 4-6), enter *9806* for *Acceleration (mm/sec**2)* which should appear as default already, and make sure the *Direction* is set to *-7* for *Y.* Click *OK* to accept the gravity setting.

Defining and Running Simulation

Click the *Motion Model* node, press the right mouse button and select *Simulation Parameters.* Enter *1.5* for simulation duration and then *300* for the number of frames.

Click the *Motion Model* node again, press the right mouse button and select *Run Simulation.* You should see the pendulum start moving back and forth about the axis of the revolute joint. We will graph the position, velocity, and acceleration of the pendulum next.

Displaying Simulation Results

The results of angular position, velocity, and acceleration of the pendulum can be directly obtained by right clicking the moving part, *pendulum-7,* from the browser.

From the browser, expand the *Parts* node and the then *Moving Parts* node. Right-click *pendulum-1*, and choose *Plot > Bryant Angles > Angle 3*.

Note that the *Bryant* angles are also known as *X-Y-Z Euler* angles or *Cardan* angles. They are simply the rotation angles of a spatial object along the *X*-, *Y*-, and *Z*-axes of the reference coordinate system. *Angle 3* is measured about the *Z*-axis.

The results of angular position, velocity, and acceleration of the pendulum can be directly obtained by right clicking the moving part, *pendulum-7,* from the browser.

From the browser, expand the *Parts* node and the then *Moving Parts* node. Right-click *pendulum-7,* and choose *Plot > Bryant Angles > Angle 3.*

Note that the *Bryant* angles are also known as *X-Y-ZEuler* angles or *Cardan* angles. They are simply the rotation angles of a spatial object along the 7-, and Z-axes of the reference coordinate system. *Angle 3* is measured about the Z-axis.

A graph like that of Figure 4-7 should appear. From the graph, the pendulum swings about the Z-axis between *-10* and *10* degrees, as expected (since no friction is involved). Also, it takes about *0.6* seconds to complete a cycle. Note that you can export the graph data, for example, by right clicking the graph and choosing *Export CSV.* Open the spreadsheet and exam the data. From the spreadsheet, the time for the pendulum to swing back to its original position; i.e., *-10* degrees., is *0.64* second, as shown in the spreadsheet of Figure 4-8. We will carry out calculations to verify these results later. Before we do that, we will graph the angular velocity and acceleration of the pendulum.

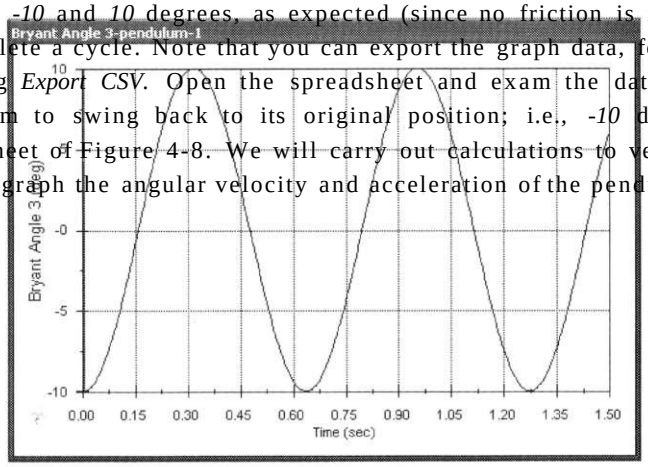


Figure 4-7 *Z*-Rotation Angle of the Pendulum

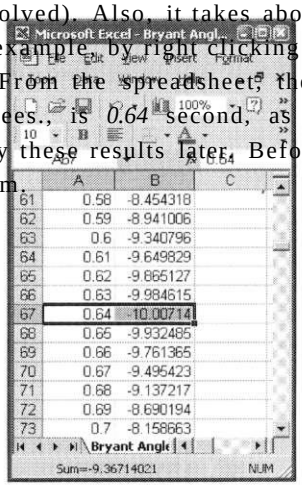


Figure 4-8

Right-click *pendulum-1*, and choose *Plot > Angular Velocity > Z Component*. Repeat the same steps and choose *Angular Acceleration > Z Component.* You should see graphs like those of Figures 4-9 and 4-10. Figure 4-9 shows that the angular velocity starts at 0, which is expected. The angular velocity varies between roughly *-100* and *100* degrees/sec. Also, the angular acceleration varies between roughly *-1,000* and *1,000* degrees/sec^2. Are these results correct? We will carry out calculations to verify if these graphs are accurate.

Save your model.

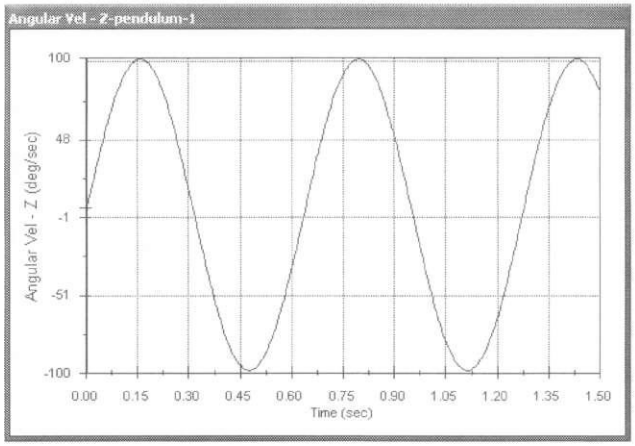


Figure 4-9 *Z*-Angular Velocity of the Pendulum

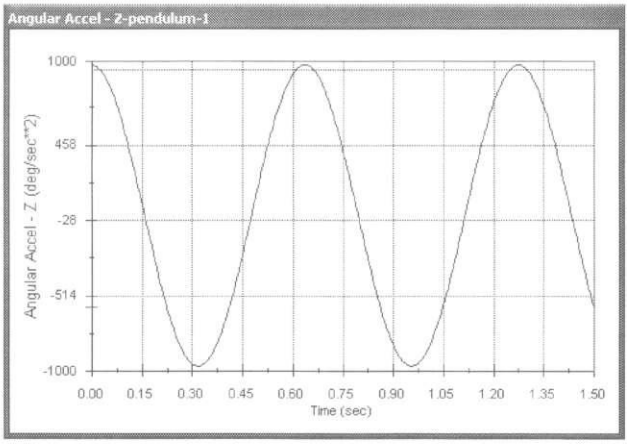


Figure 4-10 *Z*-Angular Acceleration of the Pendulum

4.4 Result Verifications

In this section, we will verify the analysis results obtained from *COSMOSMotion* using particle dynamics theory.

There are four assumptions that we have to make in order to apply the particle dynamics theory to this simple pendulum problem:

(i) Mass of the rod is negligible (this is why the diameter of the pendulum rod is very small),
(ii) The sphere is of a concentrated mass,
(iii) Rotation angle is small (remember the initial conditions we defined?), and
(iv) No friction is present.

The pendulum model has been created to comply with these assumptions as much as possible. We expect that the particle dynamics theory will give us results close to those obtained from simulation. Two approaches will be presented to formulate the equations of motion for the pendulum: energy conservation and Newton's law.

Energy Conservation

Referring to Figure 4-11, the kinetic energy and potential energy of the pendulum can be written, respectively, as

$$T = \frac{1}{2} J \dot{\theta}^2 \qquad (4.1)$$

where J is the polar moment of inertia, i.e., $J = m\ell^2$

and

$U = mg(l - \cos\theta)$ (4.2)

According to the energy conservation theory, the total mechanical energy, which is the sum of the kinetic energy and potential energy, is a constant with respect to time; i.e.,

$$\frac{d}{dt}(T+U)=0 \tag{4.3}$$

where t represents time. Hence

$$\frac{d}{dt}\left(\frac{1}{2}m\ell^2\dot{\theta}^2 + mg\ell(1-\cos\theta)\right) = m\ell^2\ddot{\theta} + mg\ell\sin\theta = 0 \tag{4.4}$$

Therefore,

$\ddot{\theta}+\frac{g}{\ell}\sin\theta=0$, and

$$\ddot{\theta}+\frac{g}{\ell}\theta=0 \tag{4.5}$$

when $\theta \approx 0$.

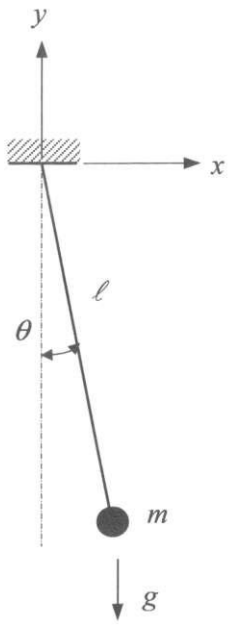


Figure 4-11 Particle Dynamics of Pendulum

Newton's Law

From the free-body diagram shown in Figure 4-12, the equilibrium equation of moment at the origin about the Z-axis (normal to the paper) can be written as:

$$\sum M = -mg\ell\sin\theta = I\ddot{\theta} = m\ell^2\ddot{\theta} \tag{4.6}$$

Hence

$\ddot{\theta}+\frac{g}{\ell}\sin\theta=0$, and

$$\ddot{\theta}+\frac{g}{\ell}\theta=0 \tag{4.7}$$

when $\theta \approx 0$.

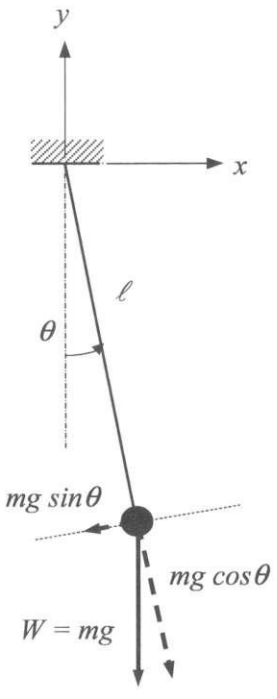


Figure 4-12 Free Body Diagram

Note that the same equation of motion has been derived from two different approaches. The linear ordinary second-order differential equation can be solved analytically.

Solving the Differential Equation

It is well known that the solution of the differential equation is

$$\theta = A_1 \cos \omega_n t + A_2 \sin \omega_n t \tag{4.8}$$

where $\omega_n = \sqrt{\frac{g}{\ell}}$, and A_1 and A_2 are constants to be determined by initial conditions. Note that $\omega_n = \sqrt{\frac{g}{\ell}} = \sqrt{\frac{9806}{100}} = 9.903$ rad/sec, and the natural frequency of the system is $f_n = \omega_n/2\pi = 1.576$ Hz. The period for a complete cycle is $T = 1/f_n = 0.634$ seconds, which is very close to that shown in all graphs (for example, Figure 4-7) and the spreadsheet data shown in Figure 4-8 generated using *Export CSV*.

The initial conditions for the pendulum are $\theta(0) = \theta_0 = -10$ degrees, and $\dot{\theta}(0) = 0$ degree/sec. Plugging the initial conditions into the solution, we have

$A_1 = \theta_0 = -10$ degrees, and

$A_2 = 0$.

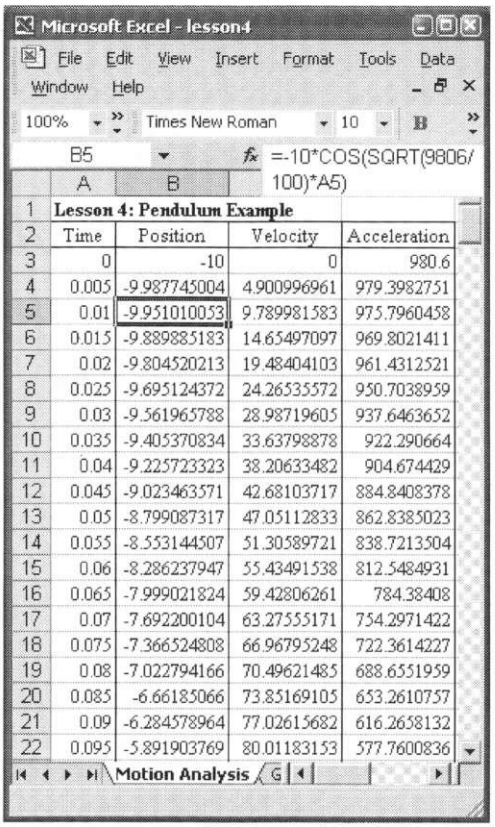
Hence, the solutions are

$$\theta = \theta_0 \cos \omega_n t \tag{4.9a}$$

$$\dot{\theta} = -\theta_0 \omega_n \sin \omega_n t \tag{4.9b}$$

$$\ddot{\theta} = -\theta_0 \omega_n^2 \cos \omega_n t \tag{4.9c}$$

The above equations represent angular position, velocity, and acceleration, of the revolute joint. These equations can be implemented into, for example, *Excel* spreadsheet shown in Figure 4-13, for numerical solutions. Columns B, C, and D in the spreadsheet show the results of Eqs. 4.9a, b, and c, respectively, between *0* and 7.5 seconds with an increments of *0.005* seconds. Data in these three columns are graphed in Figures 4-14, 15, and 16, respectively. Comparing Figures 4-14 to 16 with Figures 4-7, 4-9, and 4-10, the results obtained from theory and simulation are very close. The motion model has been properly defined, and *COSMOSMotion* gives us good results. Note that in the calculation, the angular position of the pendulum is set to zero when it aligns with the vertical axis; therefore, the pendulum swings between *-10* and *10* degrees.



	A	B	C	D
1	**Lesson 4: Pendulum Example**			
2	Time	Position	Velocity	Acceleration
3	0	-10	0	980.6
4	0.005	-9.987745004	4.900996961	979.3982751
5	0.01	-9.951010053	9.789981583	975.7960458
6	0.015	-9.889885183	14.65497097	969.8021411
7	0.02	-9.804520213	19.48404103	961.4312521
8	0.025	-9.695124372	24.26535572	950.7038959
9	0.03	-9.561965788	28.98719605	937.6463652
10	0.035	-9.405370834	33.63798878	922.290664
11	0.04	-9.225723323	38.20633482	904.674429
12	0.045	-9.023463571	42.68103717	884.8408378
13	0.05	-8.799087317	47.05112833	862.8385023
14	0.055	-8.553144507	51.30589721	838.7213504
15	0.06	-8.286237947	55.43491538	812.5484931
16	0.065	-7.999021824	59.42806261	784.38408
17	0.07	-7.692200104	63.27555171	754.2971422
18	0.075	-7.366524808	66.96795248	722.3614227
19	0.08	-7.022794166	70.49621485	688.6551959
20	0.085	-6.66185066	73.85169105	653.2610757
21	0.09	-6.284578964	77.02615682	616.2658132
22	0.095	-5.891903769	80.01183153	577.7600836

Figure 4-13 The *Excel* Spreadsheet

However, even though graphs obtained from *COSMOSMotion* and spreadsheet calculations are alike these results are not identical. This is because that the *COSMOSMotion* model is not really a simple pendulum since mass of the rod is non-zero. If you reduce the diameter of the rod, the *COSMOSMotion* results should approach those obtained through spreadsheet calculations.

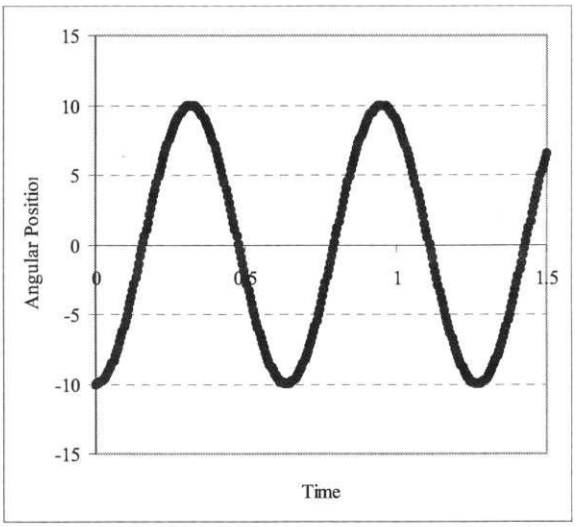


Figure 4-14 Angular Position from Theory

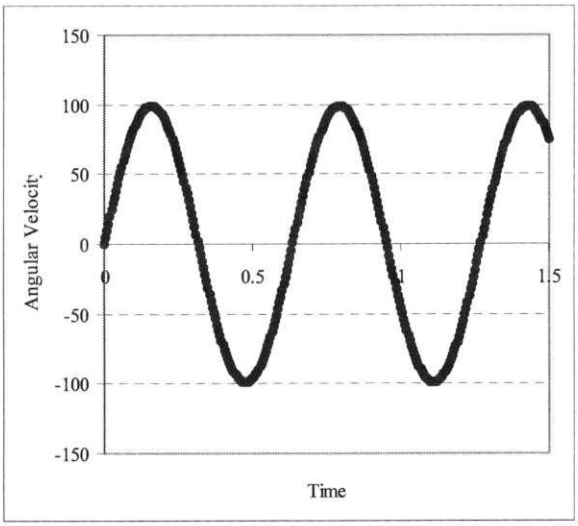


Figure 4-15 Angular Velocity from Theory

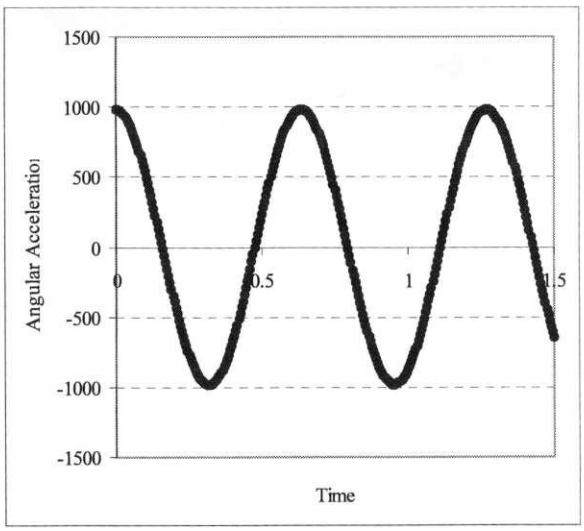


Figure 4-16 Angular Acceleration from Theory

Exercises:

1. Create a spring-damper-mass system, as shown in Figure E4-1, using *COSMOSMotion.* Note that the unstretched spring length is *3* in. The radius of the ball is *0.5* in. and the material is *Cast Alloy Steel* (mass density: *0.2637* lb_m/in^3).

   (i) Find the spring length in the equilibrium condition using *COSMOSMotion.*

   (ii) Solve the same problem using Newton's laws. Compare your results with those obtained from *COSMOSMotion.*

2. If a force $p = 2$ lb_f is applied to the ball as shown in Figure E4-1, repeat both (i) and (ii) of Problem 1.

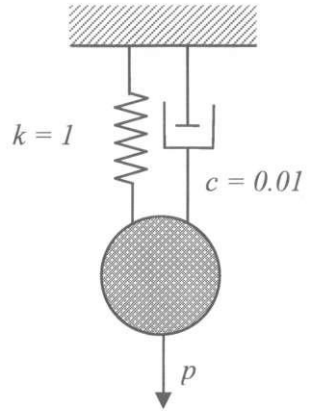
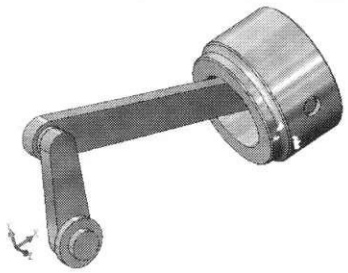


Figure E4-1 The Spring-Mass-Damper System

Notes:

Lesson 5: A Slider-Crank Mechanism



5.1 Overview of the Lesson

In this lesson, you will learn how to create simulation models for a slider-crank mechanism and conduct three analyses: kinematics, interference, and dynamics. More joint types will be introduced in this lesson. You will learn how to select assembly mates to connect parts in order to create a successful motion model. We will first drive the mechanism by rotating the crank with a constant angular velocity; therefore, conducting a kinematic analysis. After we complete a kinematic analysis, we will turn on interference checking and repeat the analysis to see if parts collide. It is very important to make sure no interference exists between parts while the mechanism is in motion. The final analysis will be dynamic, where we will add a firing force to the piston for a dynamic analysis. This lesson will start with a brief overview about the slider-crank assembly created in *SolidWorks.* At the end of this lesson, we will verify the kinematic simulation results using theory and computational methods employed for mechanism design.

5.2 The Slider-Crank Example

Physical Model

The slider-crank mechanism is essentially a four-bar linkage, as shown in Figure 5-1. They are commonly found in mechanical systems; e.g., internal combustion engine and oil-well drilling equipment. For the internal combustion engine, the mechanism is driven by a firing load that pushes the piston, converting the reciprocal motion into rotational motion at the crank.

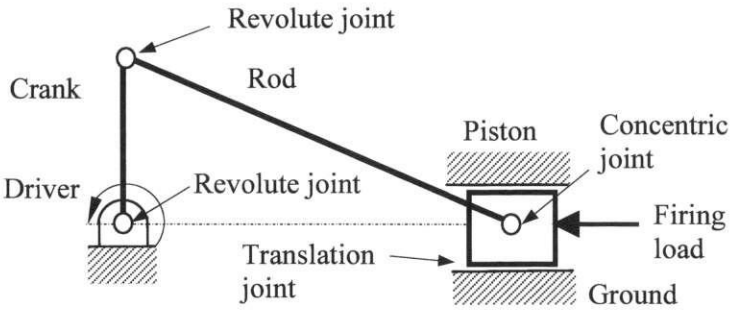


Figure 5-1 Schematic View of the Slider-Crank Mechanism

In the oil-well drilling equipment, a torque is applied at the crank. The rotational motion is converted to a reciprocal motion at the piston that digs into the ground. Note that in any case the length of the crank must be smaller than that of the rod in order to allow the mechanism to operate. This is commonly referred to as the Grashofs law. In this example, the lengths of the crank and rod are 3" and 8", respectively.

Note that the units system chosen for this example is *IPS (in-lbj-sec).* All parts are made of Aluminum, *2014 Alloy.* No friction is assumed between any pair of the components (parts or subassemblies).

SolidWorks Parts and Assembly

The slider-crank system consists of five parts and one subassembly. They are bearing, crank, rod, pin, piston, and rodandpin (subassembly, consisting of rod and pin). An exploded view of the mechanism is shown in Figure 5-2. *SolidWorks* parts and assembly have been created for you. They are *bearing.SLDPRT, crank. SLDPRT, rod.SLDPRT, pin.SLDPRTpiston.SLDPRT,* and *rodandpin.SLDASM.* In addition, there are three assembly files, *Lesson5.SLDASM, Lesson5Awithresults.SLDASM,* and *LessonSBwithresults.SLDASM.* You can find these files at the publisher's web site.

We will start with *LessonS. SLDASM,* in which all components are properly assembled. In this assembly the bearing is anchored (ground) and all other parts are fully constrained. We will suppress one assembly mate in order to allow for movement.

Same as before, the assembly file *LessonSAwithresults. SLDASM* and *Lesson5Bwithresults.SLDASM* (with firing force) consist of complete simulation models with simulation results. You may want to open these files to see the motion animation of the mechanism. In these assembly files with complete simulation results, a mate has been suppressed. You can also see how the parts move by moving the cursor to the graphics screen and press the right mouse button. In the menu option appearing next, choose *Move Component,* and drag a movable part, for example drag the crank to rotate it with respect to the bearing. The whole mechanism will move accordingly.

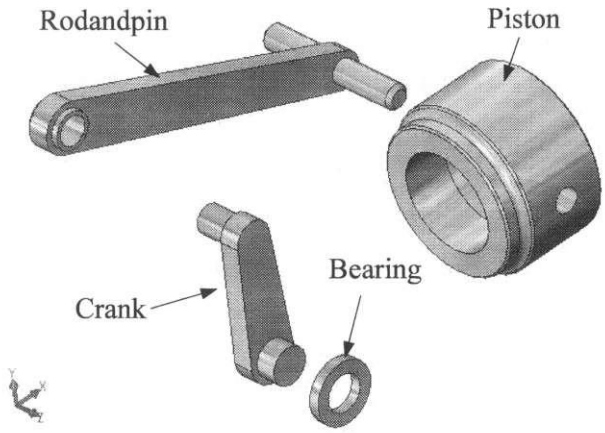
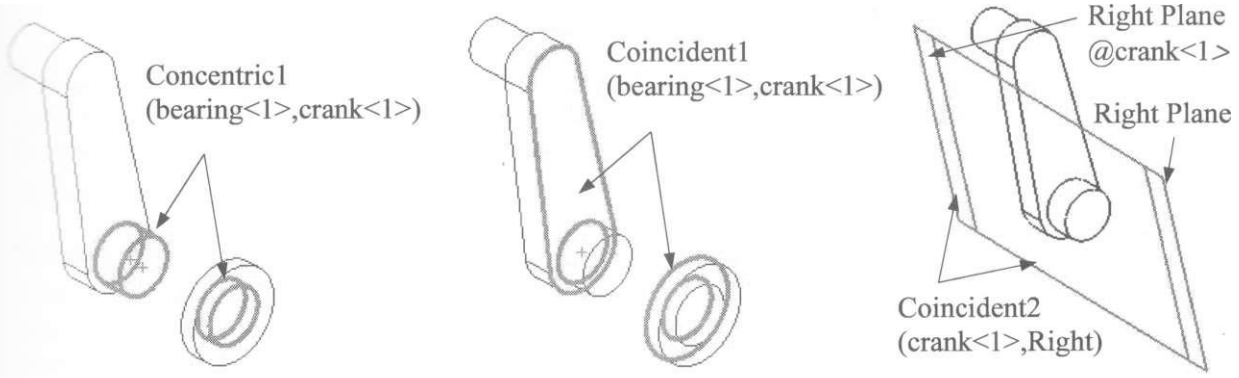


Figure 5-2 Slider-Crank Assembly (Exploded View)

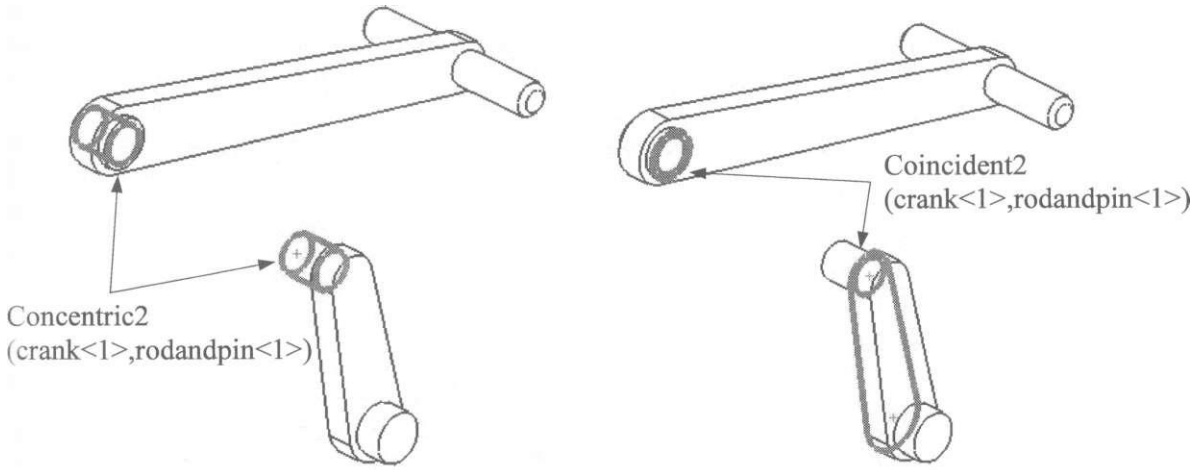
There are eight assembly mates, including five coincident and three concentric, defined in the assembly. You may want to expand the *MateGroupl* branch in the browser to see the list of mates. Move your cursor on any of the mates; you should see the entities selected for the assembly mate highlighted in the graphics screen.

The first three mates *(Concentric 1, Coincident 1,* and *Coincident!)* assemble the crank to the fixed bearing, as shown in Figure 5-3a. As a result, the crank is completely fixed. Note that the mate *Coincident!* orients the crank to the upright position. This mate will be suppressed before entering *COSMOSMotion.* Suppressing this mate will allow the crank to rotate with respect the bearing. *COSMOSMotion* will convert these two mates, *Concentricl* and *Coincident 1,* to a revolute joint.

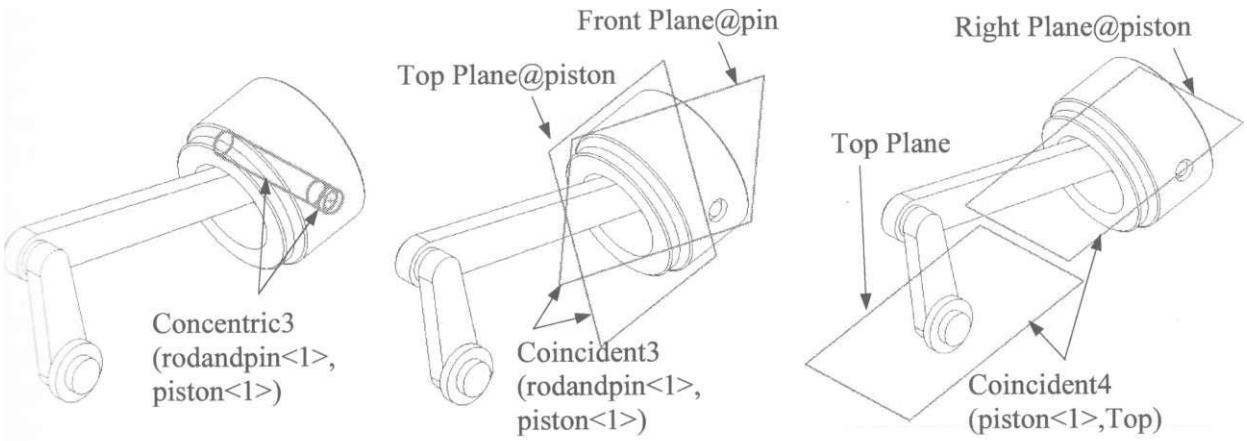
The next two mates *(Concentric2* and *Coincident^)* assemble the rod to the crank, as shown in Figure 5-3b. Unlike the crank, the rod is allowed to rotate with respect to the crank, leading to another revolute joint in *COSMOSMotion.* The next two mates *(Concentric3* and *Coincident4)* assemble the piston to the pin, allowing the piston to rotate about the pin. As a result, a revolute joint (or a concentric joint in some cases) will be added between the piston and the pin. The final mate *(CoincidentS)* eliminates the rotation by mating two planes, *Right Plane* of the piston and the *Top Plane* of the assembly, as shown in Figure 5-3c. *COSMOSMotion* will add a translational joint between the piston and the ground. Since the translational joint is composed of *Coincident5* and *Coincident4, Concentric3* will be carried over to *COSMOSMotion* as it is. Therefore, instead of a revolute joint, a cylindrical joint appears in the motion model.



(a) Assemble Mates for Crank (Exploded View)



(b) Assemble Mates for Rod (Exploded View)



(d) Assemble Mates for Piston (Unexploded View)

Figure 5-3 Assembly Mates Defined for the Mechanism

Simulation Model

In this example, after suppressing *Coincident* *COSMOSMotion* converts assembly mates to four joints: *Concentric3* (directly carrying over from assembly mate), *Revolute, Revolute2,* and *Translational,* as shown in Figure 5-4. The total number of degrees of freedom of the slider-crank mechanism can be calculated as follows:

3 (bodies) x6 (dofs/body) - 2 (revolute) X 5 (dofs/revolute) - 1 (translational) X 5 (dofs/translational) - 1 (concentric) x4 (dof s/concentric)
= 18-19 = -1

Apparently, there are two redundant dofs embedded in the motion model. Kinematically, this mechanism is identical to that of the single piston engine presented in *Lesson 1.* In *Lesson 1,* instead of defining two revolute joints, one concentric joint, and one translational joint, the engine example employs three cylindrical joints and one translational joint, resulting one free degree of freedom. Since *COSMOSMotion* will automatically detect and remove redundant dofs during motion simulation, we will not make any changes to the joints converted from the assembly mates.

Since the mechanism has one free degree of freedom, either rotating the crank or the rod (about the axis of the revolute joints), or moving the piston horizontally (along the translational joint) will be sufficient to uniquely determine the position, velocity, and acceleration of any parts in the mechanism.

The mechanism will be first driven by rotating the crank at a constant angular velocity of *360* degrees/sec. Gravity will be turned off. This will be essentially a kinematic simulation. This model will also serve for interference check. It is very important to make sure no interference exists between parts while the mechanism is in motion. The simulation results are included in *LessonSAwithresults.SLDASM.*

The next and final analysis will be dynamic, where we will add a firing force to the piston for a dynamic simulation. The results are included in *LessonSBwithresults. SLDASM.*

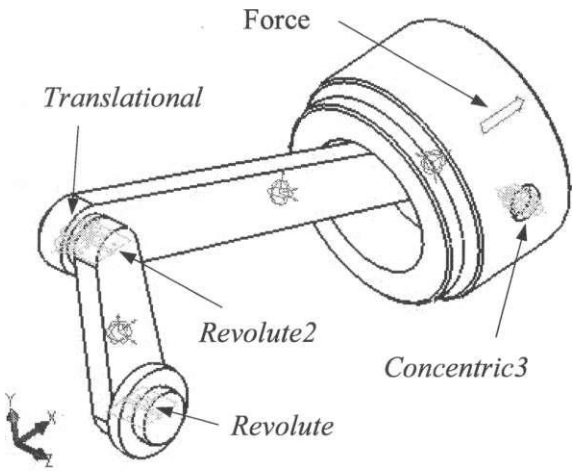


Figure 5-4 The Simulation Model

5.3 Using *COSMOSMotion*

Start *SolidWorks* and open assembly file *Lesson5.SLDASM.*

Before entering *COSMOSMotion,* we will suppress the third assembly mate, *Coincident2.* In this lesson, we will use drag-and-drop as well as right-click activated menus, instead of the *IntelliMotion Builder.*

From the *Assembly* browser, expand the *Mates* branch, right click *Coincident2,* and choose *Suppress.* The mate *Coincident2* will become inactive. Save your model.

From the browser, click the *Motion* button to enter *COSMOSMotion*.

Before creating any entities, always check the units system. Make sure *IPS* units system is chosen for this example.

Defining Bodies

From the browser, expand the *Assembly Components* branch (right underneath the *Motion Model* node) by clicking the small ⊞ button in front of it. You should see four components listed, *bearing-1*, *crank-1*, *piston-1*, and *rodandpin-1*, as shown in Figure 5-5. Also expand the *Parts* branch; you should see *Moving Parts* and *Ground Parts* listed. We will move *bearing-1* to *Ground Parts* and the remaining three parts to *Moving Parts* by using the drag-and-drop method.

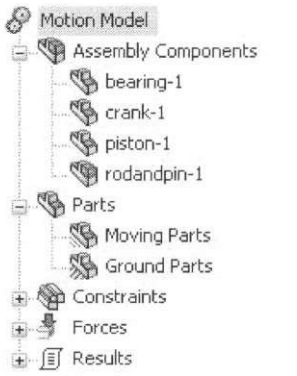


Figure 5-5

From the browser, click *bearing-1* and drag and drop it to the *Ground Parts* node. Click the first part under the *Assembly Components* node (should be *crank-1),* press the *Shift* key, and click the last part listed under the *Assembly Components* node (should be *rodandpin-1).* All three components will be selected. Drag and drop them to the *Moving Parts* node.

Expand the *Constraints* branch, and then the *Joints* branch. You should see that four joints, *Concentric3, Revolute, Revolute2,* and *Translational,* are listed (Figure 5-6). All joint symbols should appear in the graphics screen, similar to that of Figure 5-4. Expand all joints in the browser and identify the parts they connect. Take a look at the joint *Revolute* (connecting crank to bearing), where we will add a driver next.

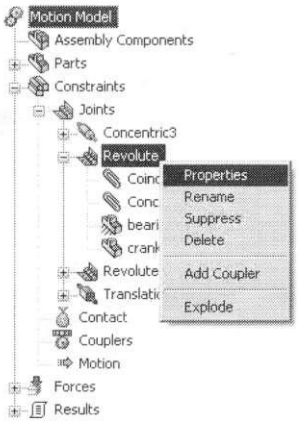


Figure 5-7

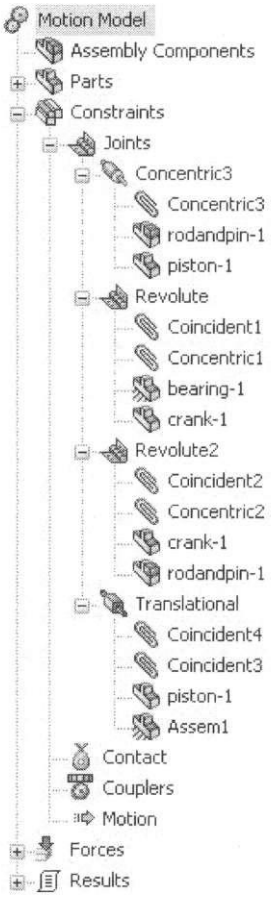


Figure 5-6

Driving Joint

From the browser, expand the *Constraints* node and then the *Joints* node. Right click the *Revolute* node and choose *Properties* (see Figure 5-7). In the *Edit Mate-Defined Joint* dialog box (Figure 5-8), under the *Motion* tab, choose *Velocity* for *Motion Type,* choose *Constant* for *Function,* and enter *360* degrees/sec for *Angular Velocity* (should appear as defaults), as shown in Figure 5-8. Click *Apply* to accept the definition. We are ready to run a simulation.

Turning Off Gravity

Click the *Options* button on top of the graphics screen. Alternatively, you may bring up the *IntelliMotion Builder* by clicking the *IntelliMotion Builder* button, and then choose the *Gravity* tab (Please refer to *Lesson 2* for more details in using the *IntelliMotion Builder*). In the *Options* dialog box,

click *Gravity On* to deselect it (or enter *0* for acceleration) and click *OK* (see Figure 5-9). The gravity should now be turned off.

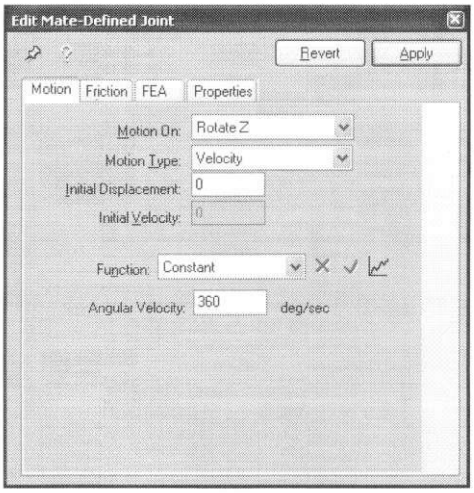


Figure 5-8 Adding Motion Drive to Joint *Revolute*

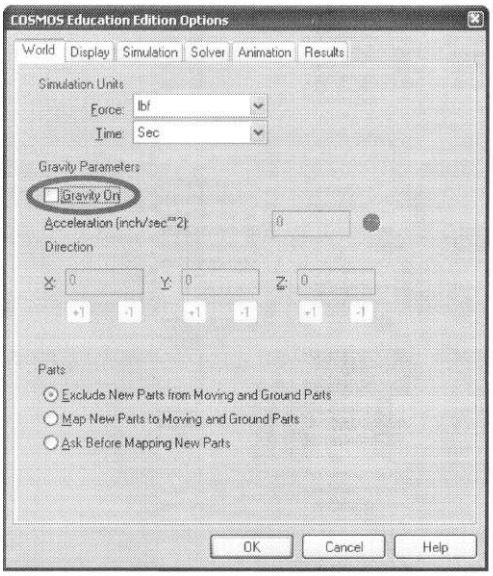


Figure 5-9 Turning Off Gravity

Running Simulation

We will use all default simulation parameters for the kinematic analysis.

Click the *Motion Model* node, press the right mouse button and select *Run Simulation.* After a few seconds, you should see the mechanism starts moving. The crank rotates *360* degrees as expected and the piston moves a complete cycle, similar to that of Figure 5-10, since the default simulation duration is *1* second.

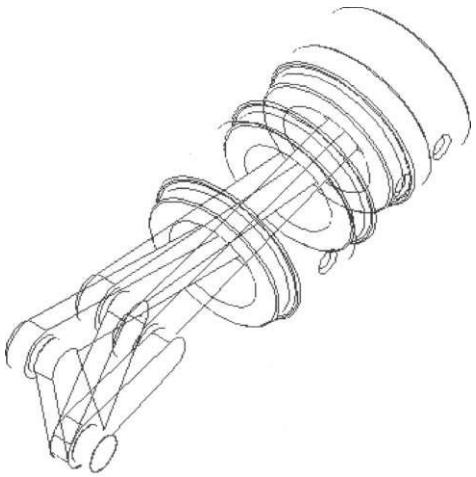


Figure 5-10 Motion Animation

Saving and Reviewing Results

We will create four graphs for the mechanism: Ap-position, *X*-velocity, and X-acceleration of the piston; and angular velocity of *Revolute2* (between crank and rod).

From the browser, expand the *Parts* branch and then the *Moving Parts* branch. Right click the *piston-1* node, and choose *Plot* > *CM Position* > X(see Figure 5-11). The graph should be similar to that of Figure 5-12. Note that from the graph, the piston moves between about 5 and *11* in. horizontally, in reference to the global coordinate system, in which the origin of the coordinate system coincides with the center point of the hole in the bearing.

At the starting point, the crank is at the upright position, and the piston is located at *7.42* in. (that is, $\sqrt{8^2 - 3^2}$) to the right of the origin of the global coordinate system. Note that the lengths of the crank and

rod are *3* and *8* in., respectively. When the crank rotates to *90* degrees counterclockwise, the position becomes *5* (which is *8-3).* When the crank rotates *270* degrees, the piston position is *11* (which is *8+3).*

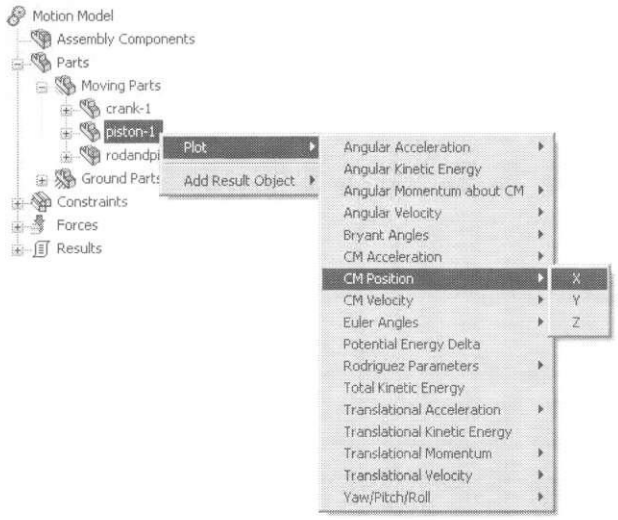


Figure 5-11

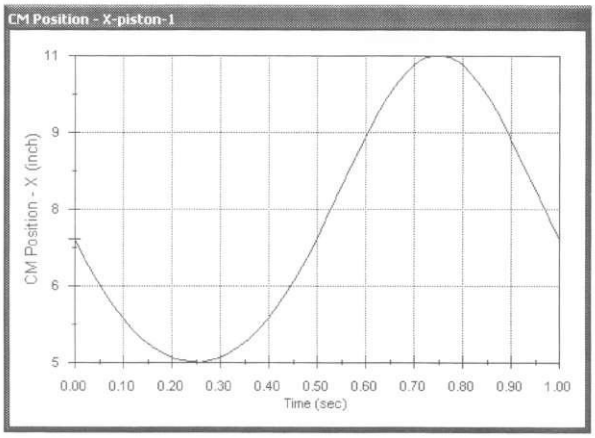


Figure 5-12 *X*-Position of the Piston

Repeat the same steps to create graphs for the velocity and acceleration of the piston in the in-direction. The graphs should be similar to those of Figures 5-13 and 5-14, respectively. If you expand the *piston-1* node in the browser, you should see there are three entities listed, *CM Accel - X-piston-1, CM Position -X-piston-1,* and *CM Velocity-X-piston-1,* as shown in Figure 5-15.

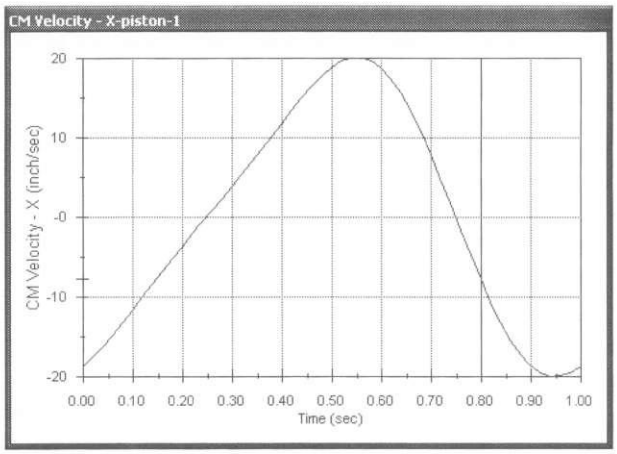


Figure 5-13 *X*-Velocity of the Piston

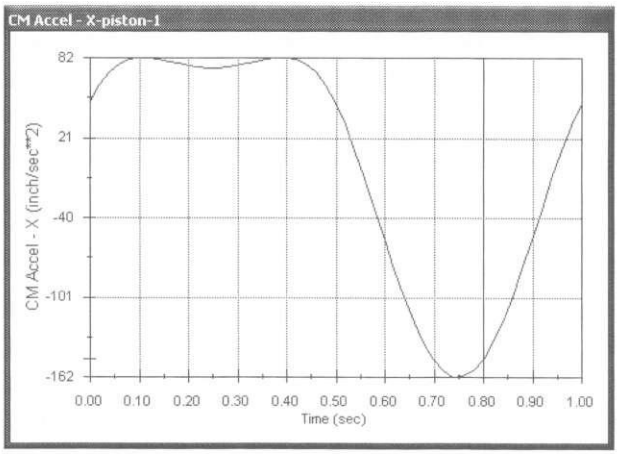


Figure 5-14 *X*-Acceleration of the Piston

The graph of the angular velocity of the joint *Revolute2* can be created by expanding the *Constraints* branch and then the *Joints* branch, right clicking *Revolute2* and selecting *Plot* > *Angular Velocity* > *Z-Component.* The graph should be similar to that of Figure 5-16. An entity, *Angular Vel - Z-Revolute2,* is added under the joint *Revolute2* in the browser.

Interference Check

Next we will learn how to perform interference check. *COSMOSMotion* allows you to check for interference in your mechanism as the parts move. You can check any of the components in your

SolidWorks assembly model for possible interference, regardless of whether a component participates in the motion model. Using the interference detection capability, you can find:

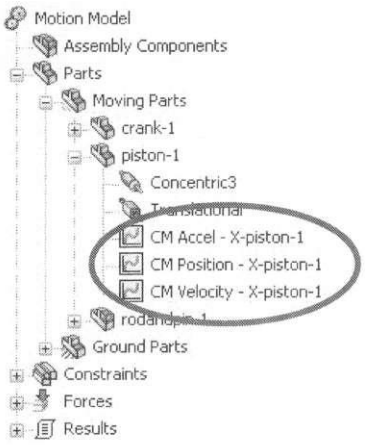


Figure 5-15

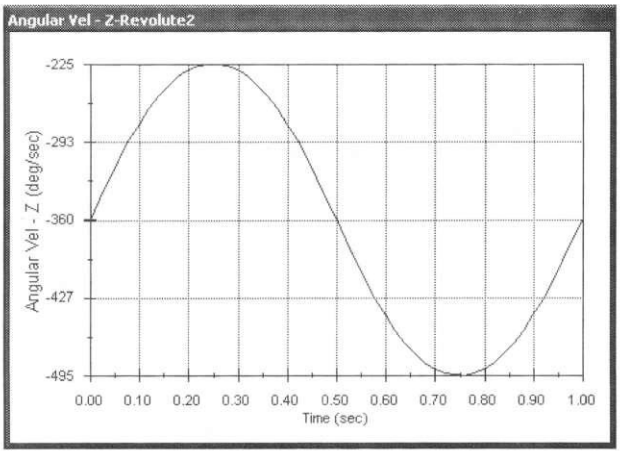


Figure 5-16 Angular Velocity of *Revolute2*

(1) All the interference that occur between the selected components as the mechanism moves through a specified range of motion, or

(2) The place where the first interference occurs between the selected components. The assembly is moved to the position where the interference occurred.

Make sure you have completed a simulation before proceeding to the interference check. Right click the *Motion Model* node in the browser and then select *Interference Check.* The *Find Interferences Over Time* dialog box appears (Figure 5-17).

To select the parts to include in the interference check, select the *Select Parts to test* text field and then pick all four components from the graphics screen (or from the browser). The *Start Frame, End Frame,* and *Increment* allow you to specify the motion frame used as the starting position, final position, and increment in between for the interference check. We will use the default numbers; i.e., *1, 51,* and *2,* for the *Start Frame, End Frame,* and *Increment,* respectively. Click the *Find Now* button (circled in Figure 5-17) to start the interference check.

After pressing the *Find Now* button, the mechanism starts moving, in which the crank rotates a complete cycle. At the same time, the *Find Interferences Over Time* dialog box expands. The list at the lower half of the dialog box shows all interference conditions detected. The frame, simulation time, parts that caused the interference, and the volume of the interference detected are listed.

Selecting the *Index* number of an interference (for example, *5*, as shown in Figure 5-18) enables the *Details* button and the *Magnifying* button at the lower right corner. Selecting *Details* displays more information about the interference. Click the *Magnifying* button to zoom in on the location of the interference. As shown in Figure 5-19, the interference occurs between the inner face of the piston and the end face of the rod. You may want to choose the *Front* view to see the interference (Figure 5-20).

Close the dialog box and save your model. After saving the model, you may want to save it again under a different name and use it for the next simulation.

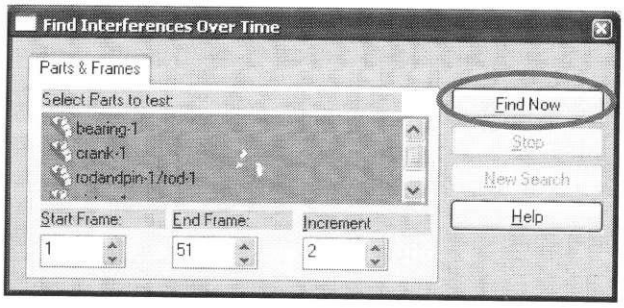


Figure 5-17 The *Find Interference Over Time* Dialog Box

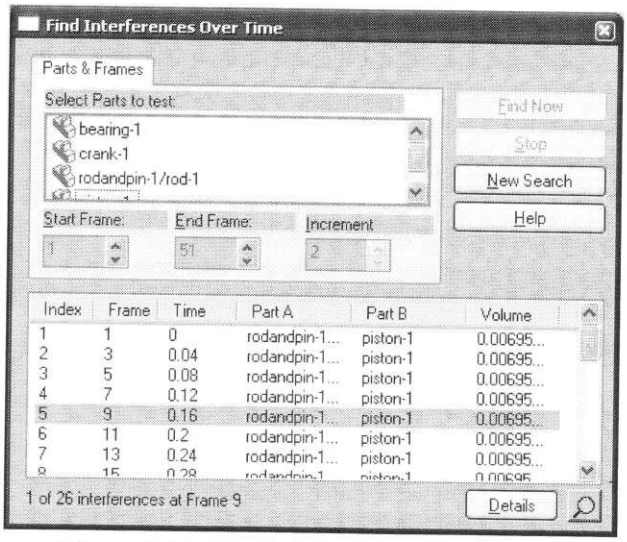


Figure 5-18 Interference Check Results

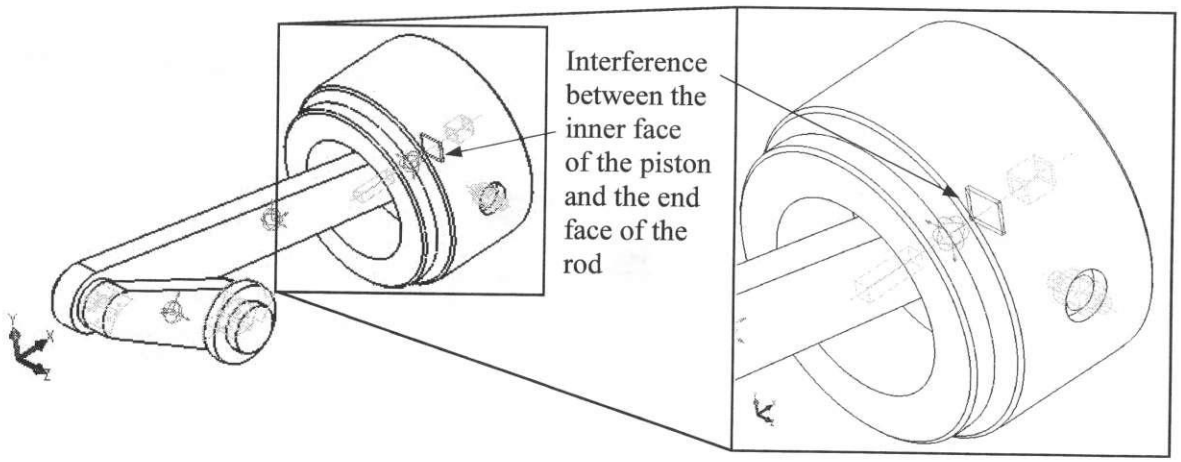


Figure 5-19 Interference Identified (View: *Lesson5*)

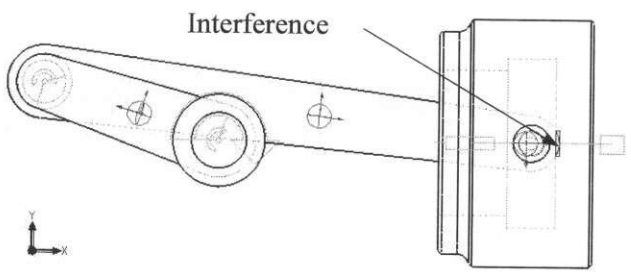


Figure 5-20 Interference (Front View)

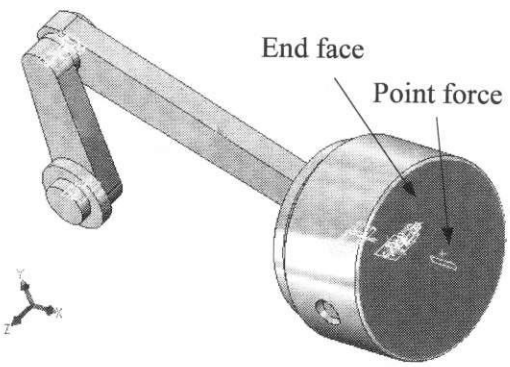


Figure 5-21 Pick the End Face for Force Application

Creating and Running a Dynamic Analysis

A force simulating the engine firing load (acting along the negative X-direction) will be added to the piston for a dynamic simulation. It will be more realistic if the force can be applied when the piston starts moving to the left (negative X-direction) and can be applied only for a selected short period. In order to do so, we will have to define measures that monitor the position of the piston for the firing load to be activated. Unfortunately, such a capability is not available in *COSMOSMotion.* Therefore, the force is simplified as a step function of *3* lb, along the negative X-direction applied for *0.1* seconds. The force will be defined as a point force at the center point of the end face of the piston, as shown in Figure 5-21.

Before we add the force, we will turn off the angular velocity driver defined at the joint *Revolute2* in the previous simulation. We will have to delete the simulation before we can make any changes to the simulation model. Delete the simulation result by clicking the *Delete Results* button at the bottom of the browser.

From the browser, expand the *Constraints* branch, and then the *Joints* branch. Right click *Revolute* to bring up the *Edit Mate-Defined Joint* dialog box (Figure 5-22). Pull-down the *Motion Type* and choose *Free.* Click *Apply* to accept the change.

Note that if you run a simulation now, nothing will happen since there is no motion driver or force defined (gravity has been turned off).

Now we are ready to add the force. The force can be added from the browser by expanding the *Forces* branch, right clicking the *Action Only* node, and choosing *Add Action-Only Force,* as shown in Figure 5-23. In the *Insert Action-Only Force* dialog box, the *Select Component to which Force is Applied* field (see Figure 5-24) will be active (highlighted in red) and ready for you to pick the component.

Pick the end face of the piston, as shown in Figure 5-21. The part *piston-1* is now listed in the *Select Component to which Force is Applied* field, and *piston-1/DDMFacelO* is listed in both the *Select Location* and the *Select Direction* fields. That is, the force will be applied to the center of the end face and in the direction that is normal to the face; i.e., in the positive X-direction.

Figure 5-23

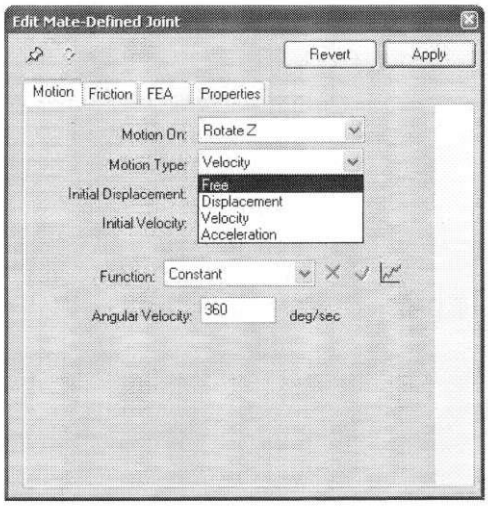


Figure 5-22 Turning Off the Motion Driver

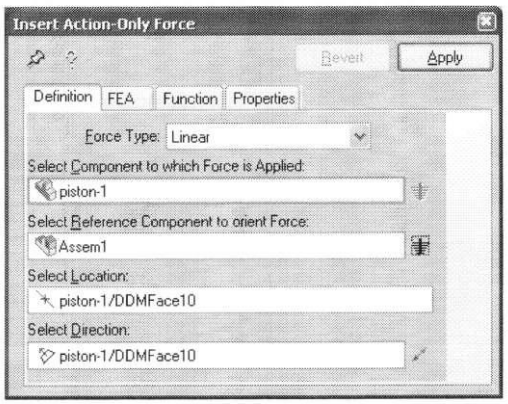


Figure 5-24 The *Insert Action-Only Force* Dialog Box

Now in the *Insert Action-Only Force* dialog box (Figure 5-24), the *Select Reference Component to orient Force* field is active (highlighted in red) and is ready for selection. We will pick the bearing (ground) for reference. Or you may simply click the ground button to the right of the field. After that, you should see *As semi* appear in the *Select Reference Component to orient Force* field.

Click the *Function* tab (see Figure 5-25), choose *Step* for function, and enter the followings:

Initial Value: -3
Final Value: 0
Start Step Time: 0
End Step Time: 0.1

Figure 5-25 Defining Force Function

Note that the negative sign for the initial value is to reverse the force direction to the negative *X*-direction.

Click the graph button (right most, as shown in Figure 5-25), the step function will appear like the one in Figure 5-26. Note that this is a smoothed step function with time extrapolated to the negative domain. The force varies from *-3* at *0* second to *0 at 0.1* seconds. During the simulation, the force will be activated at the beginning; i.e., *0* second.

Close the graph and click *Apply* button to accept the force definition. You should see a force symbol added to the piston, as shown in Figure 5-4.

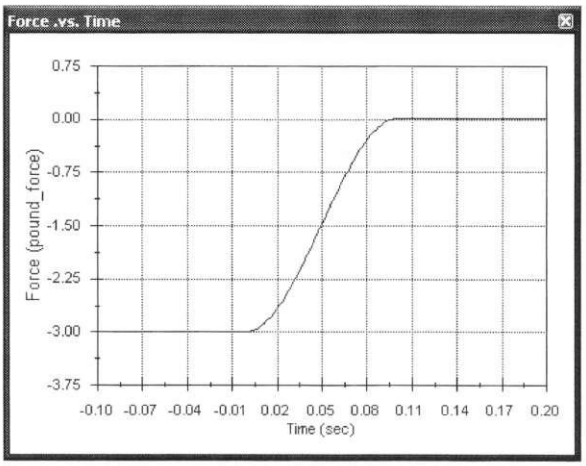


Figure 5-26 The Force Function Graph

Before running a simulation, we will increase the number of frames in order to see more refined results and graphs. From the browser, right click the *Motion Model* node, and choose *Simulation Parameters.* Change the number of frames to *500.* Accept the change and then right click the *Motion Model* node, and choose *Run Simulation.* The mechanism will move and the crank will make several turns before reaching the end of the simulation duration; i.e., *1* second by default.

Graphs created in previous simulation, such as the piston position, etc., should appear immediately at the end of the simulation. As shown in Figure 5-27, the piston moves along the.negative X-direction for about *0.15* seconds before reversing its direction. It is also evident in the velocity graph shown in Figure 5-28 that the velocity changes signs at two instances (close to *0.15* and *0.48* seconds). Recall that the force was applied for the first *0.1* seconds. Had the force application lasted longer, the piston could be continuously pushed to the left (negative X-direction) even when the piston reaches the left end and tries to move to the right (due to inertia). As a result, the crank would have been oscillating at the left of the center of the bearing; i.e., between *0* and *180* degrees about the Z-axis, without making a complete turn.

Graph the reaction force for the applied force at the piston by expanding the *Forces* branch, then the *Action Only* branch, right clicking the *ForceAO,* and choosing *Plot.* The reaction force that represents the actual force applied to the piston appears, as shown in Figure 5-29. The graph shows that the force of *3* lbf was applied at the beginning of the simulation. The force gradually decreases to *0* in the 0.7-second period, which is what we expected and is consistent with the force function, as seen in Figure 5-26.

Graph the reaction force at the joint *Revolute* (between crank and the ground) along the X-direction; you should see a graph like that of Figure 5-30. There are three peaks in Figure 5-30 representing when the largest reaction forces occur at the joint, which occurs at close to *0.15, 0.48,* and *0.8* seconds; i.e., when the piston reverses its moving direction. The results make sense.

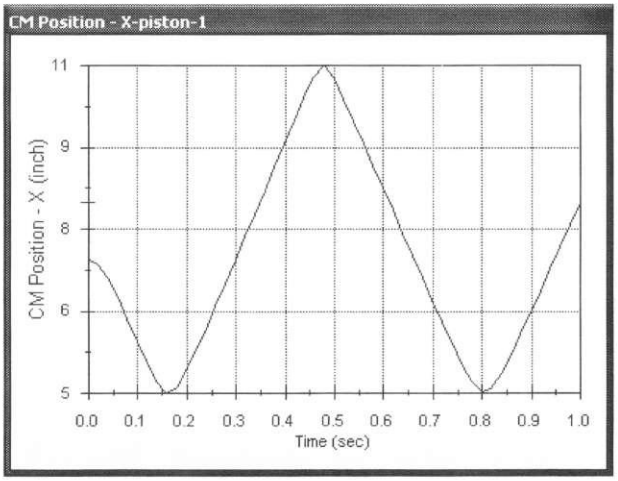


Figure 5-27 *X*-Position of the Piston

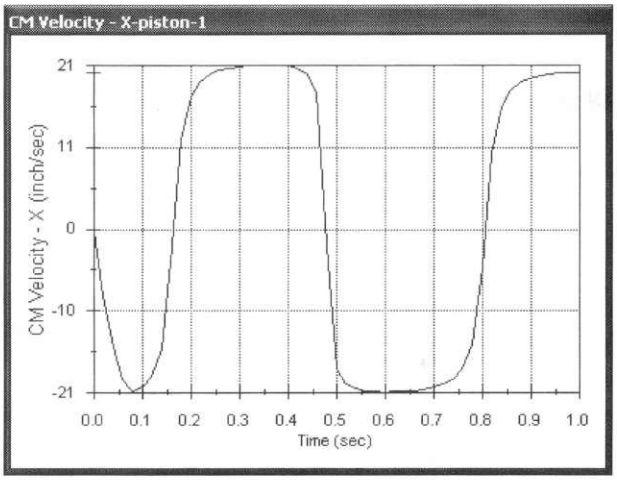


Figure 5-28 *X*-Velocity of the Piston

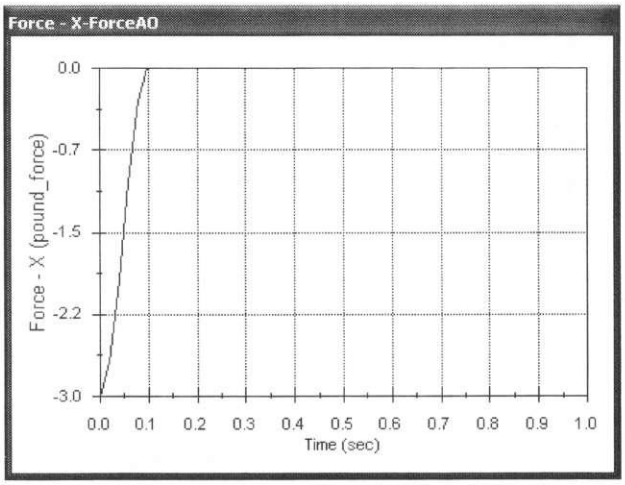


Figure 5-29 *X*-Reaction Force of *ForceAO*

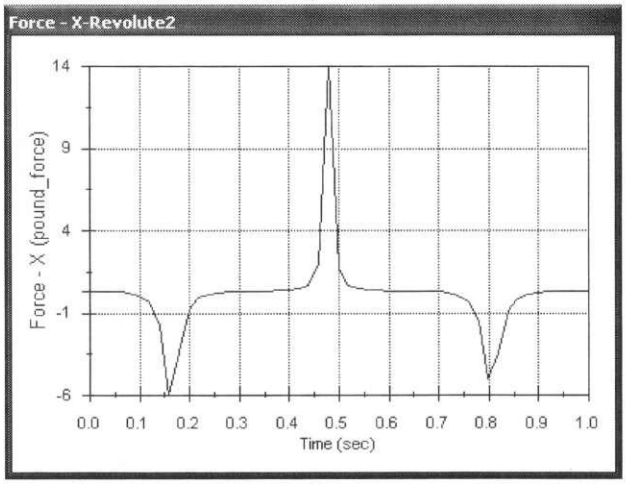


Figure 5-30 *X*-Reaction Force of *Revolute2*

Save you model. We will carry out theoretical calculations to verify the simulation results. We will focus on kinematic analysis.

5.4 Result Verifications

In this section, we will verify the motion analysis results using kinematic analysis theory often found in mechanism design textbooks. Note that in kinematic analysis, position, velocity, and acceleration of given points or axes in the mechanism, are analyzed.

In kinematic analysis, forces and torques are not involved. All bodies (or links) are assumed massless. Hence, mass properties defined for bodies are not influencing the analysis results.

The slider-crank mechanism is a planar kinematic analysis problem. A vector plot that represents the positions of joints of the planar mechanism is shown in Figure 5-31. The vector plot serves as the first step in computing position, velocity, and accelerations of the mechanism.

The position equations of the system can be described by the following vector summation,

$$\boldsymbol{Z}_1 + \boldsymbol{Z}_2 = \boldsymbol{Z}_3 \qquad (5.1)$$

where

$$\boldsymbol{Z}_1 = Z_1 \cos\theta_A + iZ_1 \sin\theta_A = Z_1 e^{i\theta_A}$$
$$\boldsymbol{Z}_2 = Z_2 \cos\theta_B + iZ_2 \sin\theta_B = Z_2 e^{i\theta_B}$$
$$\boldsymbol{Z}_3 = Z_3, \text{ since } \theta_C \text{ is always 0.}$$

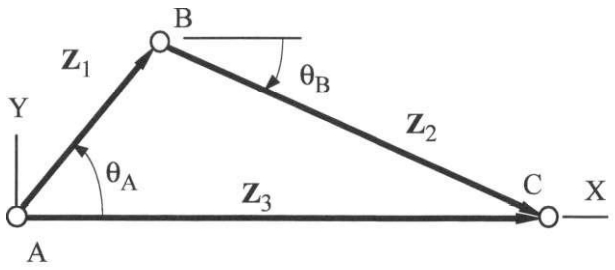


Figure 5-31 Vector Plot of the Slider-Crank Mechanism

The real and imaginary parts of Eq. 5.1, corresponding to the *X* and / components of the vectors, can be written as

$$Z_1 \cos\theta_A + Z_2 \cos\theta_B = Z_3 \qquad (5.2a)$$
$$Z_1 \sin\theta_A + Z_2 \sin\theta_B = 0 \qquad (5.2b)$$

In Eqs. 5.2a and 5.2b, Zj, Z_2, and 6_A are given. We are solving for Z_3 and 6_B. Equations 5.2a and 5.2b are non-linear. Solving them directly for Z_3 and 0_B is not straightforward. Instead, we will calculate Z_3 first, using trigonometric relations; i.e.,

$$Z_2^2 = Z_1^2 + Z_3^2 - 2Z_1 Z_3 \cos\theta_A$$

Hence,

$$Z_3^2 - 2Z_1 \cos\theta_A Z_3 + Z_1^2 - Z_2^2 = 0$$

Solving Z_3 from the above quadratic equation, we have

$$Z_3 = \frac{2Z_1 \cos\theta_A \pm \sqrt{(2Z_1 \cos\theta_A)^2 - 4\left(Z_1^2 - Z_2^2\right)}}{2} \qquad (5.3)$$

where two solutions of Z_3 represent the two possible configurations of the mechanism shown in Figure 5-32. Note that point C can be either at C or C for any given Z_1 and θ_A.

From Eq. 5.2b, θ_B can be solved by

$$\theta_B = sin^{-1}\left(\frac{-Z_1 \sin\theta_A}{Z_2}\right) \tag{5.4}$$

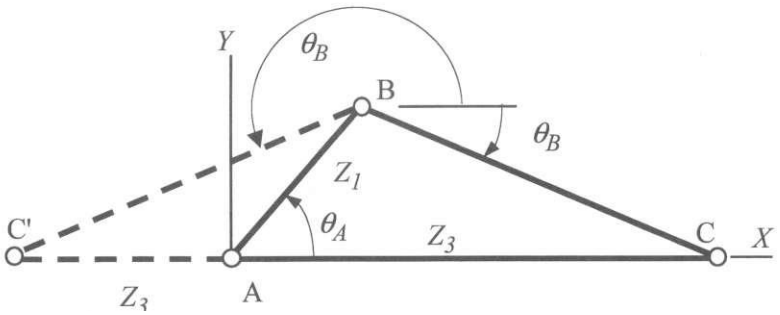


Figure 5-32 Two Possible Configurations

Similarly, θ_B has two possible solutions, corresponding to vector Zj.

Taking derivatives of Eqs. 5.2a and 5.2b with respect to time, we have

$$-Z_1 \sin\theta_A \dot{\theta}_A - Z_2 \sin\theta_B \dot{\theta}_B = \dot{Z}_3 \tag{5.5a}$$

$$Z_1 \cos\theta_A \dot{\theta}_A + Z_2 \cos\theta_B \dot{\theta}_B = 0 \tag{5.5b}$$

where $\dot{\theta}_A = \frac{d\theta_A}{dt} = \omega_A$ is the angular velocity of the rotation driver, which is a constant. Note that Eqs. 5.5a and 5.5b are linear functions of $\dot{Z}_3$ and $\dot{\theta}_B$. Rewrite the equations in a matrix from

$$\begin{bmatrix} Z_2 \sin\theta_B & 1 \\ Z_2 \cos\theta_B & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_B \\ \dot{Z}_3 \end{bmatrix} = \begin{bmatrix} -Z_1 \sin\theta_A \dot{\theta}_A \\ -Z_1 \cos\theta_A \dot{\theta}_A \end{bmatrix} \tag{5.6}$$

Equation 5.6 can be solved by

$$\begin{bmatrix} \dot{\theta}_B \\ \dot{Z}_3 \end{bmatrix} = \begin{bmatrix} Z_2 \sin\theta_B & 1 \\ Z_2 \cos\theta_B & 0 \end{bmatrix}^{-1} \begin{bmatrix} -Z_1 \sin\theta_A \dot{\theta}_A \\ -Z_1 \cos\theta_A \dot{\theta}_A \end{bmatrix}$$

$$= \frac{1}{-Z_2 \cos\theta_B} \begin{bmatrix} 0 & -1 \\ -Z_2 \cos\theta_B & Z_2 \sin\theta_B \end{bmatrix} \begin{bmatrix} -Z_1 \sin\theta_A \dot{\theta}_A \\ -Z_1 \cos\theta_A \dot{\theta}_A \end{bmatrix}$$

$$= \frac{1}{-Z_2 \cos\theta_B} \begin{bmatrix} Z_1 \cos\theta_A \dot{\theta}_A \\ Z_1 Z_2 \cos\theta_B \sin\theta_A \dot{\theta}_A - Z_1 Z_2 \sin\theta_B \cos\theta_A \dot{\theta}_A \end{bmatrix}$$

$$= \begin{bmatrix} -\dfrac{Z_1 \cos\theta_A \dot{\theta}_A}{Z_2 \cos\theta_B} \\ -\dfrac{Z_1\left(\cos\theta_B \sin\theta_A \dot{\theta}_A - \sin\theta_B \cos\theta_A \dot{\theta}_A\right)}{\cos\theta_B} \end{bmatrix} \tag{5.7}$$

Hence

$$\dot{\theta}_B = -\frac{Z_1 \cos\theta_A \dot{\theta}_A}{Z_2 \cos\theta_B} \tag{5.8}$$

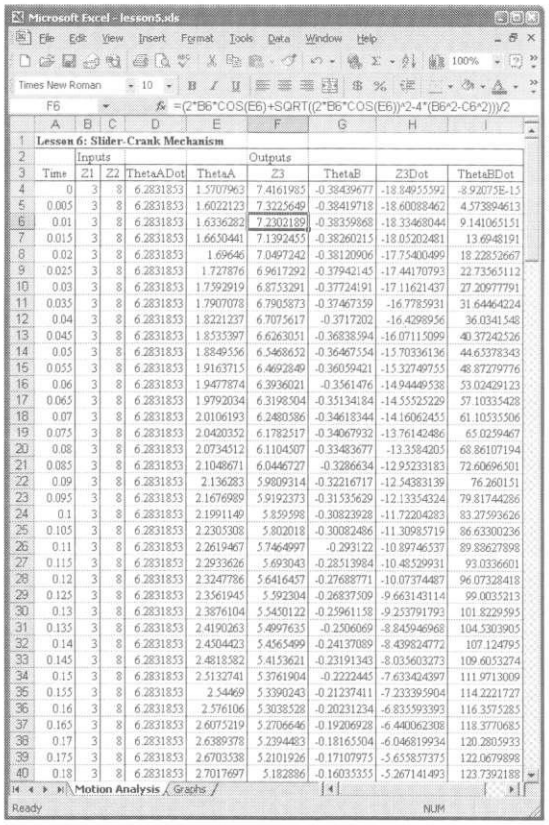
and

$$\dot{Z}_3 = Z_1\left(\tan\theta_B \cos\theta_A \dot{\theta}_A - \sin\theta_A \dot{\theta}_A\right) \tag{5.9}$$

In this example, $Z_1 = 3$, $Z_2 = 8$, and the initial conditions are $\theta_A(0) = \pi/2$ and $\theta_B(0) = \sin^{-1}(3/8)$.

The solutions can be implemented using a spreadsheet. The *Excel* spreadsheet file, *lesson5.xls*, can be found at the publisher's web site. As shown in Figure 5-33, Columns A to I represent time, Z_1, Z_2, $\dot{\theta}_A$, θ_A, Z_3, θ_B, $\dot{Z}_3$, and $\dot{\theta}_B$, respectively. Note that in this calculation, $Z_3(0)>0$ is assumed, hence $\theta_B(0)<0$ (clockwise), as illustrated in Figure 5-32. This is consistent with the initial conditions we defined for the motion model.

Figures 5-34 to 5-36 show the graphs of data in Columns F, H, and I, respectively. Comparing Figures 5-34 to 5-36 with Figures 5-12, 5-13, and 5-16, the simulation analysis results are verified.



Microsoft Excel - lesson5.xls

File Edit View Insert Format Tools Data Window Help

Times New Roman 10 100%

F6 =(2*B6*COS(E6)+SQRT((2*B6*COS(E6))^2-4*(B6^2-C6^2)))/2

	A	B	C	D	E	F	G	H	I
1	**Lesson 6: Slider-Crank Mechanism**								
2		Inputs				Outputs			
3	Time	Z1	Z2	ThetaADot	ThetaA	Z3	ThetaB	Z3Dot	ThetaBDot
4	0	3	8	6.2831853	1.5707963	7.4161985	-0.38439677	-18.84955592	-8.92075E-15
5	0.005	3	8	6.2831853	1.6022123	7.3225649	-0.38419718	-18.60088462	4.573894613
6	0.01	3	8	6.2831853	1.6336282	7.2302189	-0.38359868	-18.33468044	9.141065151
7	0.015	3	8	6.2831853	1.6650441	7.1392455	-0.38260215	-18.05202481	13.6948191
8	0.02	3	8	6.2831853	1.69646	7.0497242	-0.38120906	-17.75400499	18.22852667
9	0.025	3	8	6.2831853	1.727876	6.9617292	-0.37942145	-17.44170793	22.73565112
10	0.03	3	8	6.2831853	1.7592919	6.8753291	-0.37724191	-17.11621437	27.20977791
11	0.035	3	8	6.2831853	1.7907078	6.7905873	-0.37467359	-16.7785931	31.64464224
12	0.04	3	8	6.2831853	1.8221237	6.7075617	-0.3717202	-16.42989856	36.0341548
13	0.045	3	8	6.2831853	1.8535397	6.6263051	-0.36838594	-16.07115099	40.37242526
14	0.05	3	8	6.2831853	1.8849556	6.5468652	-0.36467554	-15.70336136	44.63378343
15	0.055	3	8	6.2831853	1.9163715	6.4692849	-0.36059421	-15.32749755	48.87279776
16	0.06	3	8	6.2831853	1.9477874	6.3936021	-0.3561476	-14.94449538	53.02429123
17	0.065	3	8	6.2831853	1.9792034	6.3198504	-0.35134184	-14.55525229	57.10335428
18	0.07	3	8	6.2831853	2.0106193	6.2480586	-0.34618344	-14.16062455	61.10535506
19	0.075	3	8	6.2831853	2.0420352	6.1782517	-0.34067932	-13.76142486	65.0259467
20	0.08	3	8	6.2831853	2.0734512	6.1104507	-0.33483677	-13.3584205	68.86107194
21	0.085	3	8	6.2831853	2.1048671	6.0446727	-0.3286634	-12.95233183	72.60696501
22	0.09	3	8	6.2831853	2.136283	5.9809314	-0.32216717	-12.54383139	76.260151
23	0.095	3	8	6.2831853	2.1676989	5.9192373	-0.31535629	-12.13354324	79.81744286
24	0.1	3	8	6.2831853	2.1991149	5.859598	-0.30823928	-11.72204283	83.27593626
25	0.105	3	8	6.2831853	2.2305308	5.802018	-0.30082486	-11.30985719	86.63300236
26	0.11	3	8	6.2831853	2.2619467	5.7464997	-0.293122	-10.89746537	89.88627898
27	0.115	3	8	6.2831853	2.2933626	5.693043	-0.28513984	-10.48529931	93.0336601
28	0.12	3	8	6.2831853	2.3247786	5.6416457	-0.27688771	-10.07374487	96.07328418
29	0.125	3	8	6.2831853	2.3561945	5.592304	-0.26837509	-9.663143114	99.0035213
30	0.13	3	8	6.2831853	2.3876104	5.5450122	-0.25961158	-9.253791793	101.8229595
31	0.135	3	8	6.2831853	2.4190263	5.4997635	-0.2506069	-8.845946968	104.5303905
32	0.14	3	8	6.2831853	2.4504423	5.4565499	-0.24137089	-8.439824772	107.124795
33	0.145	3	8	6.2831853	2.4818582	5.4153621	-0.23191343	-8.035603273	109.6053274
34	0.15	3	8	6.2831853	2.5132741	5.3761904	-0.2222445	-7.633424397	111.9713009
35	0.155	3	8	6.2831853	2.54469	5.3390243	-0.21237411	-7.233395904	114.2221727
36	0.16	3	8	6.2831853	2.576106	5.3038528	-0.20231234	-6.835593393	116.3575285
37	0.165	3	8	6.2831853	2.6075219	5.2706646	-0.19206928	-6.440062308	118.3770685
38	0.17	3	8	6.2831853	2.6389378	5.2394483	-0.18165504	-6.046819934	120.2805933
39	0.175	3	8	6.2831853	2.6703538	5.2101926	-0.17107975	-5.655857375	122.0679898
40	0.18	3	8	6.2831853	2.7017697	5.182886	-0.16035355	-5.267141493	123.7392188

Motion Analysis / Graphs /

Ready NUM

Figure 5-33 The *Excel* Spreadsheet

Note that the angular velocity in Figures 5-16 and 5-36 are in different units. In Figure 5-16, the angular is in degrees/sec. Whereas, in Figure 5-36, it is in rad/sec. After converting the unit to degree/sec, the relative magnitudes (Figure 3-37) matches well. However, the angular velocity is zero at $t = 0$ in Figure 5-37 and is -360 degree/sec in Figure 5-16. This is because that the angular velocity of the joint *Revolute2* reported in Figure 5-16 is a relative measure, which measures the angular velocity of the rod with respect to the crank. When the crank rotates *360* degree/sec, the rod rotates -360 degree/sec

relatively. In Figure 3-37, the angular velocity θ_B is the angular velocity of the rod referring to the ground. Therefore, it is zero when the crank is in the upright position.

Note that the accelerations of a given joint in the mechanism can be formulated by taking one more derivative of Eqs. 5.5a and 5.5b with respect to time. The resulting two coupled equations can be solved, using *Excel* spreadsheet. This is left as an exercise.

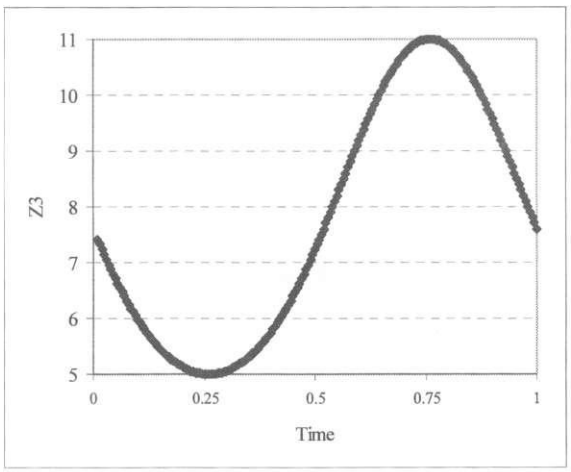


Figure 5-34 Position of the Piston (Column F)

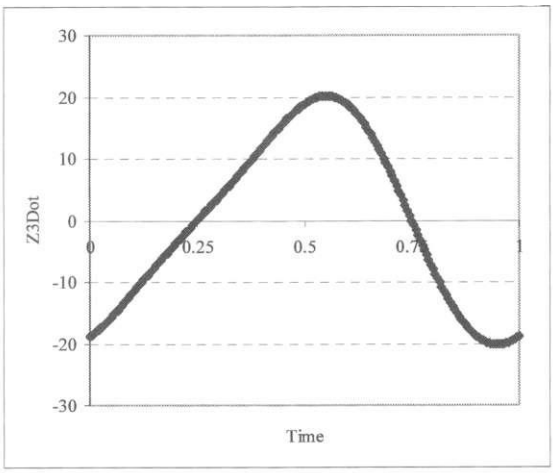


Figure 5-35 Velocity of the Piston (Column H)

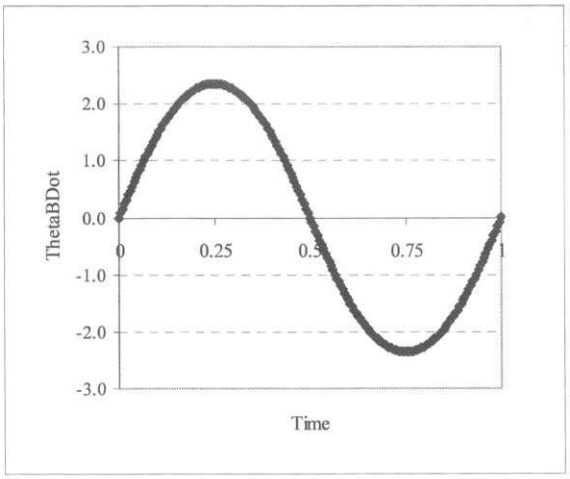


Figure 5-36 Angular Velocity of the Joint: *Revolute2* (Column I)

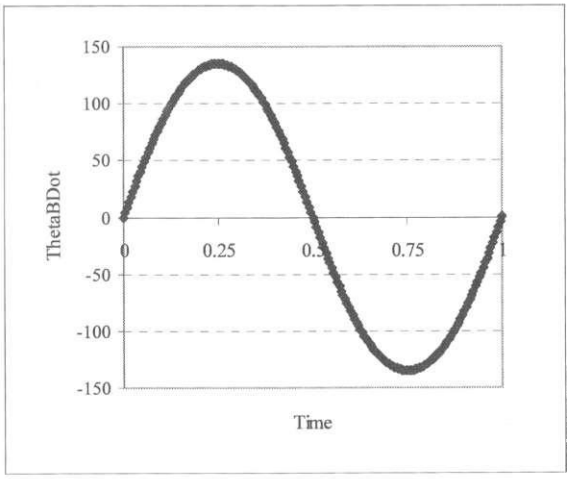


Figure 5-37 Angular Velocity of the Joint: *Revolute2* (in degree/sec)

Exercises:

1. Derive the acceleration equations for the slider-crank mechanism, by taking derivatives of Eqs, 5.5a and 5.5b with respect to time. Solve these equations for the linear acceleration of the piston and the angular acceleration of the joint *Revolute2,* using a spreadsheet. Compare your solutions with those obtained from *COSMOSMotion.*

2. Use the same slider-crank model to conduct a static analysis using *COSMOSMotion.* The static analysis in *COSMOSMotion* should give you equilibrium configuration(s) of the mechanism due to gravity (turn on the gravity). Show the equilibrium configuration(s) of the mechanism and use the energy method you learned from Sophomore *Statics* to verify the equilibrium configuration(s).

3. Change the length of the crank from *3* to *5* in. in *SolidWorks.* Repeat the kinematic analysis discussed in this lesson. In addition, change the crank length in the spreadsheet *{Microsoft Excel* file, *lesson5.xls).* Generate position and velocity graphs from both *COSMOSMotion* and the spreadshee: Do they agree with each other? Does the maximum slider velocity increase due to a longer crank? Is there any interference occurring in the mechanism?

4. Download five *SolidWorks* parts from the publisher's web site to your computer (folder name *Exercise 5-4).*

   (i) Use these five parts, i.e., bearing, crankshaft, connecting rod, piston pin, and piston (see Figure E5-1), to create an assembly like the one shown in Figure E5-2. Note that the crankshaft mus: orient at *45°* CCW, as shown in Figure E5-2.

   (ii) Create a motion model for kinematic analysis. Conduct motion analysis by defining a driva that drives the crankshaft at a constant angular speed of *1,000* rpm.

   (iii) Use the spreadsheet *lesson5.xls* to calculate the piston velocity. Compare your calculations with those obtained from *COSMOSMotion.*

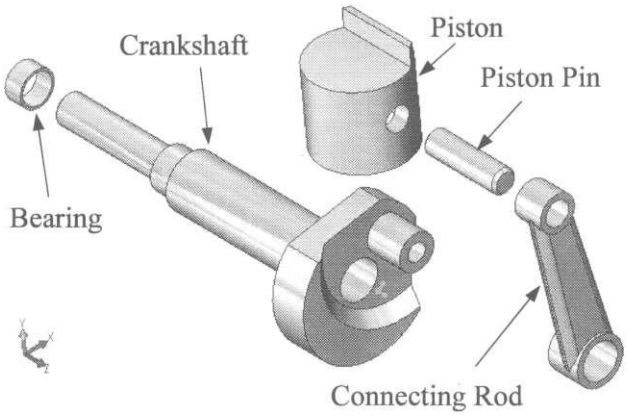


Figure E5-1 Five *SolidWorks* Parts

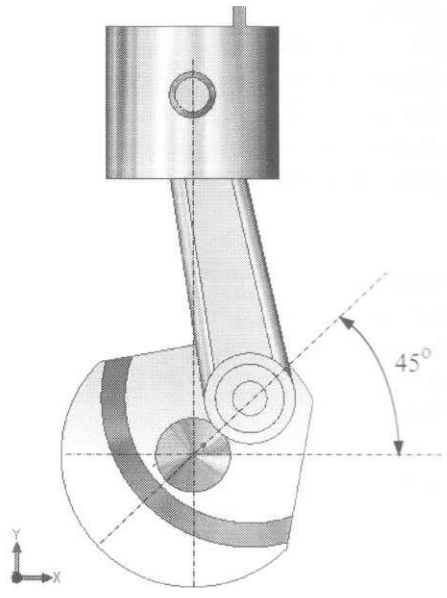
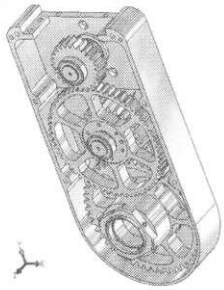


Figure E5-2 Assembled Configuration

Notes:

Lesson 6: A Compound Spur Gear Train



6.1 Overview of the Lesson

In this lesson we will discuss how to simulate motion of a spur gear train. A gear train is a set or system of gears arranged to transfer torque or energy from one part of a mechanical system to another. A gear train consists of driving gears that are mounted on the input shaft, driven gears mounted on the output shaft, and idler gears that interpose between the driving and driven gears in order to maintain the direction of the output shaft to be the same as the input shaft or to increase the distance between the drive and driven gears. There are different kinds of gear trains, such as simple gear train, compound gear train, epicylic gear train, etc., depending on how the gears are shaped and arranged as well as the fuctions they intend to perform. The gear train we are simulating in this lesson is a compound gear train, in which two or more gears are used to transmit torque or energy. All gears included in this lesson are spur gears; therefore, the shafts that these gears mounted on are in parallel.

In *COSMOSMotion,* gear pair is defined as a special coupler constraint. Joint couplers allow the motion of a revolute, cylindrical, or translational joint to be coupled to the motion of another revolute, cylindrical or translational joint. The two coupled joints may be of the same or different types. For example, a revolute joint may be coupled to a translational joint. The coupled motion may also be of the same or different type. For example, the rotary motion of a revolute joint may be coupled to the rotary motion of a cylindrical joint, or the translational motion of a translational joint may be coupled to the rotary motion of a cylindrical joint. To create a gear pair, we will be coupling two revolute joints. Usually a concentric and a coincident mates will lead to a revolute joint, as seen in previous lessons. Coupling two revolute joints for a gear pair will be carried out in *SolidWorks* using the advance assembly mate option, where two axes that pass through the respective revolute joints (or gears) are picked for the gear mate. The gear mate will be mapped to a gear mate joint in *COSMOSMotion.*

In fact, neither *SolidWorks* nor *COSMOSMotion* cares about the detailed geometry of the gear pair; i.e., if the gear teeth mesh adequately. You may simply uses cylinders or disks to represent the gears. No detailed tooth profile is necessary for any of the computations involved. Apparently, force and moment between a pair of teeth in contact will not be calculated in gear train simulations. However, there are other important data being calculated by *COSMOSMotion,* such as reaction force exerting on the driven shaft (for a dynamic analysis), which is critical for mechanism design. In any case, pitch circle diameters are essential for defining gear pair and gear trains in *COSMOSMotion.* Gear ratio of the gear train, which is defined by the ratio of the angular velocities of the output and input gears, is determined by the pitch circle diameters of the individual gear pairs in the gear train.

Although cylinders or disks are sufficient to model gears in *SolidWorks,* we will use more realistic gears throughout this lesson. All gears in the example are shown with detailed geometry, including teeth. In addition, detailed parts, including shafts, bearing, screws and aligning pins are included for a realistic gear train system, as shown in Figure 6-1. In this gear train simulation, we will focus more on graphical

animation, less on computations of physical quantities. We will add a motion driver to drive the input shaft.

6.2 The Gear Train Example

Physical Model

The gear train example we are using for this lesson is part of a gearbox designed for an experimental lunar rover. The gear train is located in a gear box which is part of the transmission system of the rover, driven by a motor powered by solar energy. The purpose of the gear train is to convert a high-speed rotation and small torque generated by the motor to a low speed rotation and large torque output in order to drive the wheels of the rover. The gear train consists of four spur gears mounted on three parallel shafts, as shown in Figure 6-1.

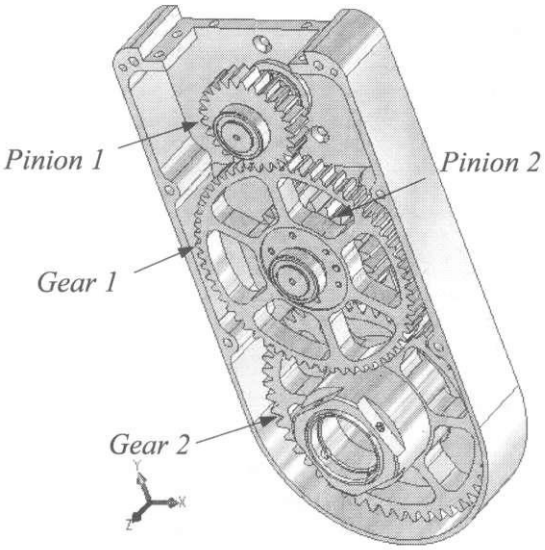


Figure 6-1 The Gear Train System in Rover

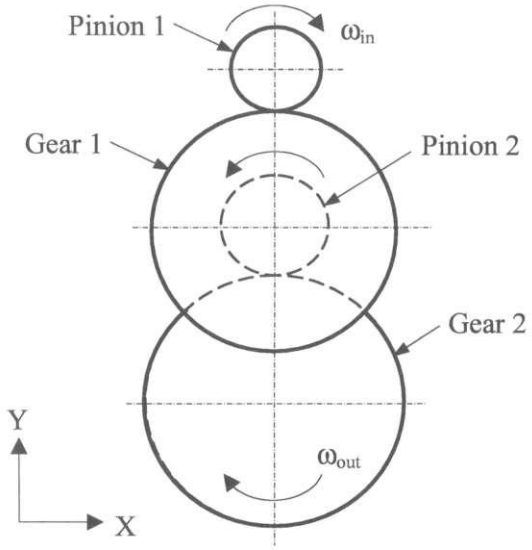


Figure 6-2 Schematic View of the Gear Train

The four spur gears form two gear pairs: *Pinion 1* and *Gear* 7, and *Pinion 2* and *Gear* 2, as depicted in Figure 6-1 and 6-2. Note that *Pinion 1* is the driving gear that connects to the motion driver; e.g., a motor. The motor rotates in a clockwise direction, therefore, driving *Pinion 1*. *Gear 1* is the driven gear of the first gear pair, which is mounted on the same shaft as *Pinion 2*. Both rotate in a counterclockwise direction. *Gear 2* is driven by *Pinion* 2, and rotates in a clockwise direction. Note that the diameters of the pitch circles of the four gears are: 50, 120, 60, and 125 mm, respectively; and the numbers of teeth are 25, 60, 24, and 50, respectively. Therefore, the circular pitch P_c, the diametral pitch P_d, and module m of the first gear pair are, respectively

$$p_c = \frac{\pi d_{p1}}{N_{p1}} = \frac{\pi d_{g1}}{N_{g1}} = \frac{\pi(50)}{25} = \frac{\pi(120)}{60} = 6.283 \text{ mm}, \quad p_d = \frac{N_{p1}}{d_{p1}} = \frac{N_{g1}}{d_{g1}} = \frac{25}{50} = \frac{60}{120} = 0.5 \text{ mm}^{-1},$$
$$m = \frac{d_{p1}}{N_{p1}} = \frac{d_{g1}}{N_{g1}} = \frac{50}{25} = \frac{120}{60} = 2 \text{ mm}. \tag{6.1}$$

For the 2nd gear pair, we have

$$p_c = \frac{\pi d_{p2}}{N_{p2}} = \frac{\pi d_{g2}}{N_{g2}} = \frac{\pi(60)}{24} = \frac{\pi(125)}{50} = 7.854 \text{ mm}, \; p_d = \frac{N_{p2}}{d_{p2}} = \frac{N_{g2}}{d_{g2}} = \frac{24}{60} = \frac{50}{125} = 0.4 \text{ mm}^{-1},$$
$$m = \frac{d_{p1}}{N_{p1}} = \frac{d_{g1}}{N_{g1}} = \frac{50}{24} = \frac{125}{50} = 2.5 \text{ mm}. \quad (6.2)$$

Therefore, the gear ratio g_r of the gear train is:

$$g_r = \frac{\omega_{out}}{\omega_{int}} = \frac{N_{p1}}{N_{g1}}\frac{N_{p2}}{N_{g2}} = \frac{25}{60}\frac{12}{50} = \frac{1}{5}, \text{ or}$$
$$= \frac{d_{p1}}{d_{g1}}\frac{d_{p2}}{d_{g2}} = \frac{50}{120}\frac{60}{125} = \frac{1}{5} \quad (6.3)$$

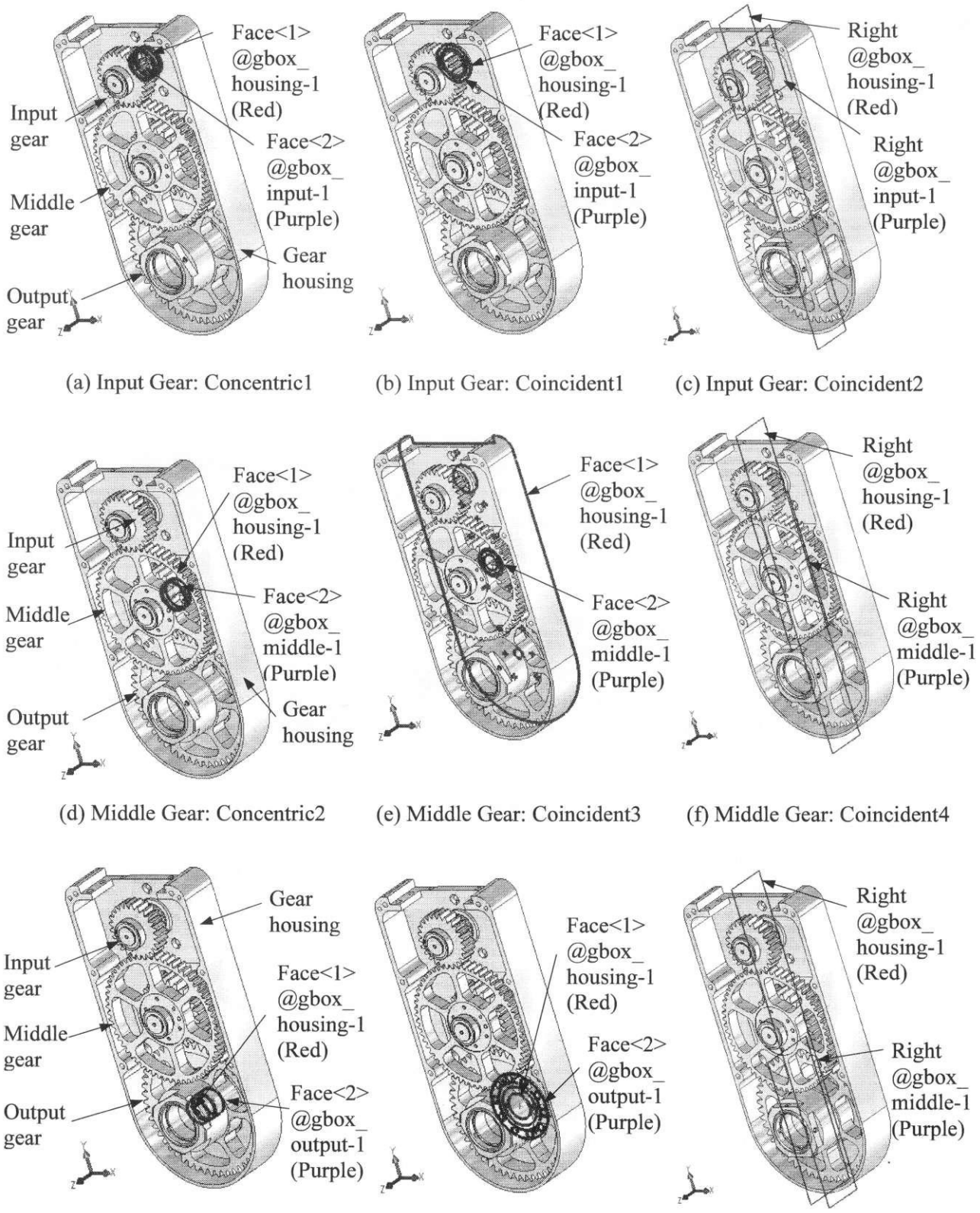
where ω_{out} and ω_{in} are the output and input angular velocities of the gear train system, respectively; d_{p1}, d_{g1}, d_{p2}, and d_{g2} are the pitch diameters of the respective four gears; and N_{p1}, N_{g1}, N_{p2}, and N_{g2} are the number of teeth of the four respective gears. The gear ratio of the gear train shown in Figure 1 is *1:5;* i.e., the angular velocity is reduced 5 times at the output. Theoretically, the torque output will increase 5 times if there is no loss due to; e.g., friction. Note that we will use *MMGS* units system for this lesson.

SolidWorks Parts and Assembly

In this lesson, *SolidWorks* parts of the gear train have been created for you. There are six files created, *gbox housing.SLDPRT, gbox input.SLDPRT, gboxjniddle.SLDPRT, gbox_output.SLDPRT, Lesson6.SLDASM,* and *Lesson6withresults.SLDASM.* You can find these files at the publisher's web site (http://www.schroffl.com/). We will start with *Lesson6.SLDASM,* in which the gears are assembled to the housing. In addition, the assembly file *Lesson6withresults.SLDASM* consists of a complete simulation model with simulation results.

Note that the housing part in *SolidWorks* was converted directly from *Pro/ENGINEER* part. The three gear parts in *SolidWorks* were converted from respective *Pro/ENGINEER* assemblies. There were nine, nine, and six distinct parts within the three gear assemblies, respectively. These three *Pro/ENGINEER* assemblies (and associated parts) were first converted to *SolidWorks* as assemblies. Parts in each assembly were then merged into a single gear part in *SolidWorks.* The detailed part and assembly conversions as well as merging multiple parts into a single part in *SolidWorks* can be found in Appendix C.

In the *SolidWorks* assembly models *Lesson6.SLDASM* (and *Lesson6withresults.SLDASM),* there are nine assembly mates. The first three mates, *Concentric1, Coincidentl,* and *Coincident2* assemble the input gear to the housing. The input gear is fully constrained. Note that before entering *COSMOSMotion,* we will suppress *Coincident2* in order to allow rotational degree of freedom for the input gear about the Z-axis (see Figures 6-3a, b, and c). Similarly, the next three mates, *Concentric2, Coincident^* and *Coincident4* assemble the middle gear to the housing. Again, we will suppress *Coincident4* to allow the middle gear to rotate about the Z-axis (see Figures 6-3d, e and f). The third set of mates, *Concentric3, Coincident5,* and *Coincident6* does the same for the output gear (see Figures 6-3g, h and i). Similarly, *Coincident6* will be suppressed to allow for rotation. Note that the three suppressed mates are created to properly orient the three gears, so that the gear teeth mesh well between pairs.



(a) Input Gear: Concentric1 (b) Input Gear: Coincident1 (c) Input Gear: Coincident2

(d) Middle Gear: Concentric2 (e) Middle Gear: Coincident3 (f) Middle Gear: Coincident4

(g) Output Gear: Concentric3 (h) Output Gear: Coincident5 (i) Output Gear: Coincident6

Figure 6-3 Assembly Mates Defined in *Lesson6.SLDASM*

As mentioned earlier, one important factor for the animation to "look right" is to mesh the gear teeth properly. You may want to use the *Front* view and zoom in to the tooth mesh areas to check if the two pairs of gears mesh well (see Figure 6-4). They should mesh well, which is accomplished by the three coincident mates that will be suppressed before entering *COSMOSMotion.* Note that the tooth profile is represented by straight lines, instead of more popular ones such as involutes, just for simplicity.

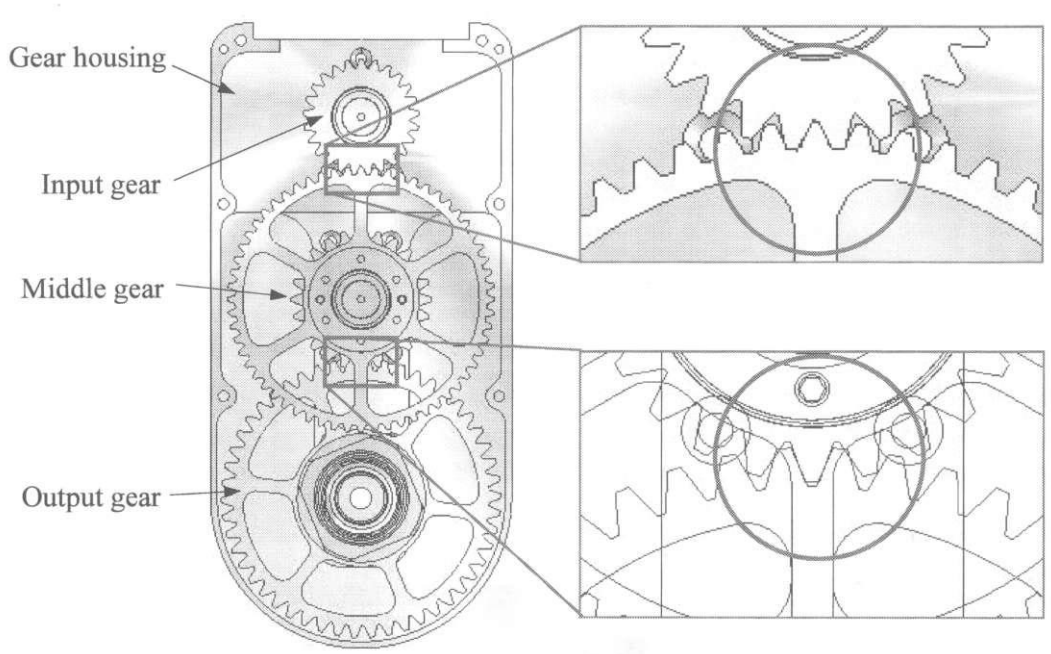


Figure 6-4 Gear Teeth Properly Meshed

If you turn on the axis display *(View > Axes),* axes that pass through the center of the gears about the Z-axis are defined for each gear. These axes are necessary for creating gear mates.

Simulation Model

The gear housing will be defined as the ground part. All three gears will rotate with respect to their respective axes. The four gears will be meshed into two gear pairs; *Pinion 1* with *Gear* 7, and *Pinion 2* with *Gear* 2, as discussed earlier. In *SolidWorks,* gear pairs are created by selecting two axes of respective gears (or cylinders) using *Advanced Mates* capability. Before the gear mates can be created, we will suppress the three coincident mates that help properly orient the gears; i.e., *Coincident2, Coincident4* and *Coincident6,* in order to allow desired gear rotation motion. When these mates are suppressed, revolute joints will be created between the housing and the three gear parts in *COSMOSMotion,* as shown in Figure 6-5. The revolute joint between the housing and the input gear will be driven at a constant angular velocity of 360 degrees/sec. We will basically conduct a kinematic analysis for this example.

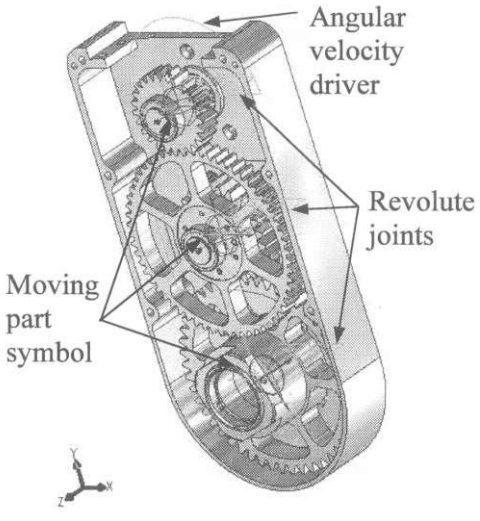


Figure 6-5 Gear Train Motion Model

6.3 Using *COSMOSMotion*

Start *SolidWorks* and open the assembly file *Lessond. SLDASM.*

Note that when you open the assembly, you will see a message window, as shown in Figure 6-6, indicating that *SolidWorks* is unable to locate *gbox input, sldasm. SolidWorks* is trying to locate the assembly from which the input gear part was created. Since the input gear assembly and its associated parts are not available in the *Lesson 6* folder, *SolidWorks* is unable to locate it. It is fine to click *No* and not to locate *gbox_input.sldasm.* Not locating the assembly file for the input gear part will not affect the motion simulation in this lesson. After clicking *No, SolidWorks* will ask you to locate the middle gear assembly and output gear assembly. Choose *No* for both. The assembly files that *SolidWorks* is looking for are actually located in the subfolder under *Lesson 6* as well as the Appendix C folder. You may choose *Yes* from the message window and locate the missing files in one ofthese two folders.

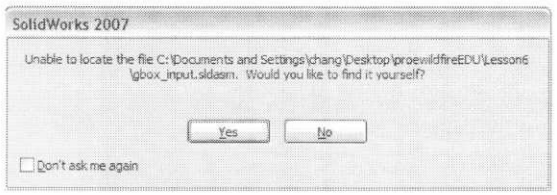


Figure 6-6

Before entering *COSMOSMotion* there are two things need to be done. First, we will suppress three assembly mates, *Coincident2, Coincident4* and *Coincident6.* Second, we will create two gear mates for the two gear pairs, respectively.

From the *Assembly* browser, expand the *Mates* branch, right-click *Coincident2,* and choose *Suppress.* The mate *Coincident2* will become inactive. Repeat the same to suppress *Coincident4* and *Coincident6.* Save your model.

Next, turn on the axis view by choosing from the pull-down menu, *View* > *Axes.* All three axes, one for each gear, will appear in the graphics screen.

We will create two gear mates. Choose from the pull-down menu *Insert* > *Mate.* In the *Mate* dialog box (overlapping with the browser), the *Mate Selections* field will be active (in red), and is ready for you to pick entities. Pick the axes of the input and middle gears from the graphics screen. Choose *Advanced Mates,* click *Gear,* and enter *50mm* and *120mm* for *Ratio,* as shown in Figure 6-7. Click the checkmark button on top to accept the mate definition, and click the same button one more time to close the dialog box.

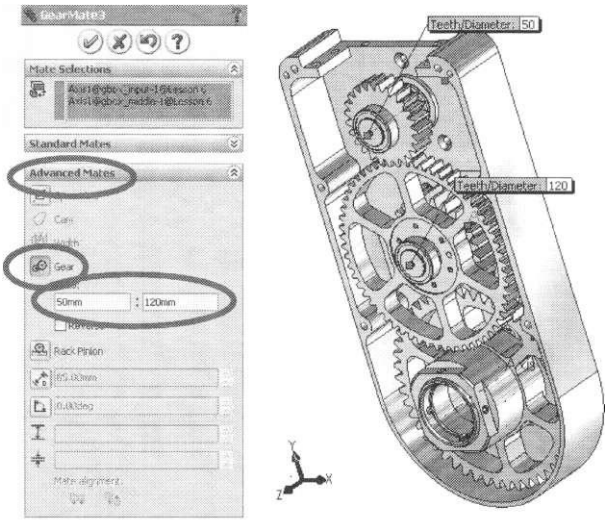


Figure 6-7 Defining Gear Mate

Note that if the axes of the two gears are pointing in the opposite direction, you will have to click *Reverse* (right below the *Ratio* text field in the *Mate* dialog box) to correct the rotation direction. In this example, all three axes are pointing in the same direction. Therefore, do not choose *Reverse.*

Repeat the same steps to define the second gear mate. This time, pick the axes of the middle and output gears, and enter *60mm* and *125mm* for *Ratio.* Two new mates, *Gear Mate 1 (gbox_input<l>, gbox_middle<l>)* and *GearMate2(gbox_middle<l>,gbox_output<l>),* are now listed under *Mates.*

Now we are ready to enter *COSMOSMotion.*

Click the *Motion* button enter *COSMOSMotion.*

Now we are ready to enter *COSMOSMotion.*

Click the *Motion* button # enter *COSMOSMotion.*

Similar to previous lessons, we will use the browser, and basic drag-and-drop and right-clicl^ methods to create and simulate the gear train motion in this lesson.

Before creating any entities, always check the units system. Make sure the units system chosen is *MMGS.*

Defining Bodies

From the browser, expand the *Assembly Components* branch. You should see four parts listed, *gbox housing-1, gboxjnput-l, gboxmiddle-1,* and *gbox_output-l,* as shown in Figure 6-8.

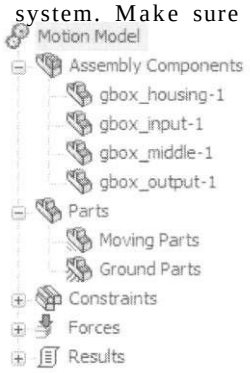


Figure 6-8

Also expand the *Parts* branch; you should see *Moving Parts* and *Ground Parts* listed. Go ahead to move *gboxjiousing-l* to *Ground Parts* and move the three gears to *Moving Parts* by using the drag-and-drop method.

Expand the *Constraints* branch, and then the *Joints* branch. You should see two gear mates and three revolute joints listed, as shown in Figure 6-9. In addition, you should see the revolute joint symbols appear in the graphics screen, similar to those of Figure 6-5. Expand all joints in the browser and identify the parts they connect. Take a look at the joint *Revolute2* (connecting the input gear to the gear housing), where we will add a driver next. Note that you may see a different revolute joint connecting the input gear to the gear housing. Make sure you pick the right one.

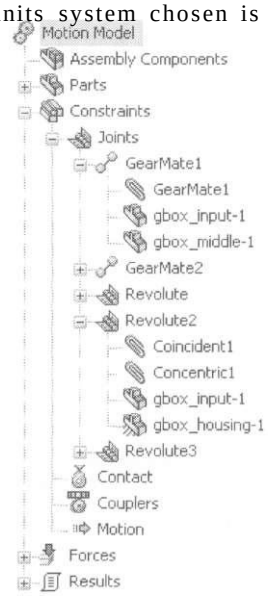


Figure 6-9

Driving Joint

From the browser, expand the *Constraints* node and then the *Joints* node. Right-click the *Revolute2* node and choose *Properties.* In the *Edit Mate-Defined Joint* dialog box (Figure 6-10), under the *Motion* tab (default), choose *Rotate Z* for *Motion On,* choose *Velocity* for *Motion Type,* choose *Constant* for *Function,* and enter *360* degrees/sec for *Angular Velocity* (should appear as defaults). Click *Apply* to accept the definition. A motion driver symbol should appear at *Revolute2,* as shown in Figure 6-5.

We are ready to run a simulation. We will use all default simulation parameters.

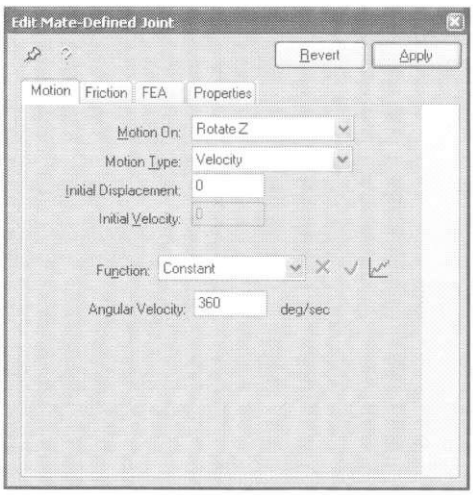


Figure 6-10 Adding Motion Driver to Joint *Revolute2*

Running Simulation

Click the *Motion Model* node, press the right mouse button and select *Run Simulation.* After a few seconds, you should see the gears start turning. The input gear rotates *360* degrees as expected since the default simulation duration is 7 second.

Saving and Reviewing Results

We will graph the angular velocity of the output gear.

From the browser, expand the *Parts* branch and then the *Moving Parts* branch. Right-click the *ghox_output-l* node, and choose *Plot > Angular Velocity > Z Component* (Figure 6-11). The graph should appear and is similar to that of Figure 6-12, which shows that the output velocity is a constant of *72* degrees/sec. Note that this magnitude is one fifth of the input velocity since the gear ratio is 7:5. Both the input *(Pinion* 7) and output gears *(Gear 2)* rotate in the same direction. *COSMOSMotion* gives good results.

Save your model.

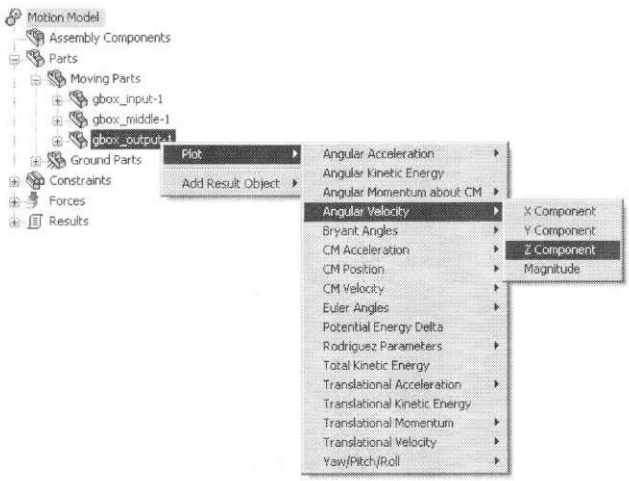


Figure 6-11

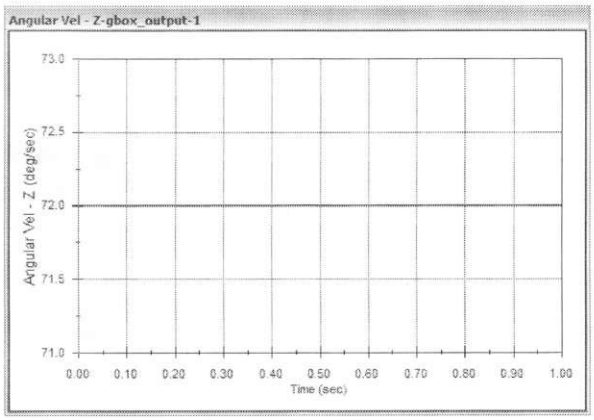


Figure 6-12 Angular Velocity of Output Gear

Exercises:

1. The same gear train will be used for this exercise. Create a constant torque for the input gear *(gboxinput.SLDPRT)* about the Z-axis. Turn on friction for all three axles *(Steel-Dry/Steel-Dry).* Define and run a 2-second dynamic simulation for the gear train.

   (i) What is the minimum torque that is required to rotate the input gear, and therefore, the entire gear train?

   (ii) If the torque applied to the input gear is *100* mm N, what is the output angular velocity of the gear train at the end of the 2-second simulation? Verify the simulation result using your own calculation.

   (iii) Create a graph for the reaction moment between gears of the first gear pair *{GearMatel)* due to the *100* mm N torque. What is the reaction moment obtained from simulation?

Notes:

Lesson 7: Cam and Follower

7.1 Overview of the Lesson

In this lesson, we will learn cam and follower, or cam-follower. A cam-follower is a device for converting rotary motion into linear motion. The simplest form of a cam is a rotating disc with a variable radius, so that its profile is not circular but oval or egg-shaped. When the disc rotates, its edge (or side face) pushes against a follower (or cam follower), which may be a small wheel at the end of a lever or the end of the lever or rod itself. The follower will thus rise and fall at exactly the same amount as the variation in radius. By profiling a cam appropriately, a desired cyclic pattern of straight-line motion, in terms of position, velocity, and acceleration, can be produced.

We will learn to create a motion model and simulate the control of opening and closing of an inlet or exhaustive valve, usually found in internal combustion engines, using cam-follower connections. In a design such as that of Figure 7-1, the drive for the camshaft is taken from the crankshaft through a timing chain, which keeps the cams synchronized with the movement of the piston so that the valves are opened or closed at a precise instant. The mechanism we will be working with consists of bushings, camshaft, pushrod, rocker, valve, valve guide and spring, as shown in Figure 7-1. The cam-follower connects the camshaft and the pushrod. When the cam on the camshaft pushes the pushrod up, the rocker rotates and pushes the valve on the other side downward. The spring surrounding the valve gets compressed, and opens up the inlet for air to flow into the combustion chamber.

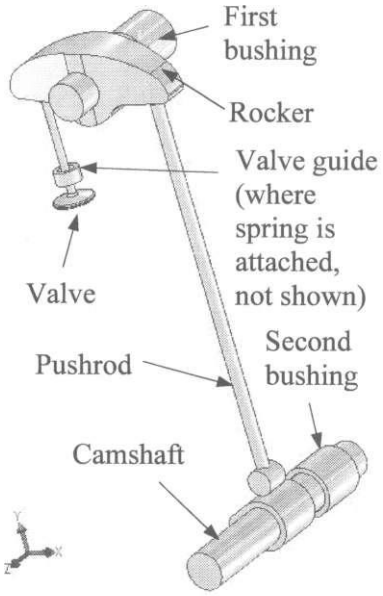


Figure 7-1 The Mechanism of Engine Inlet or Outlet Valve

7.2 The Cam and Follower Example

Physical Model

The camshaft and the rocker will rotate about the axes of their respective revolute joints connecting them to their respective bearings (defined as ground body). The camshaft is driven by a motor of constant velocity of *600* rpm (or *10* rev/sec). The profile of the cam consists of two circular arcs of *0.25* and *0.5* in. radii, respectively, as shown in Figure 7-2. The lower arc is concentric with the shaft, and the center of the upper arc is *0.52* in. above the center of the shaft. When the camshaft rotates, the cam mounted on the shaft pushes the pushrod up by up to *0.27* in. (that is, *0.52+0.25-0.5 = 0.27).* As a result, the rocker will rotate and push the valve at the other end downward at a frequency of *10* times/sec. The valve will move again up to *0.27* in. downward since the pushrod and the valve are positioned at an equal distance from the rotation axis of the rocker. When the camshaft rotates where the larger circular arc *(0.5* in. radius) of

the cam is in contact with the follower (in this example, the pushrod), the pushrod has room to move downward. At this point, the rocker will rotate back since the spring is being uncompressed. As a result, the valve will move up, and therefore, close the inlet. The valve will be open for about *120* degree per cycle, based on the cam design shown in Figure 7-2.

The unit system chosen for this example is *IPS* and all parts are made up of steel.

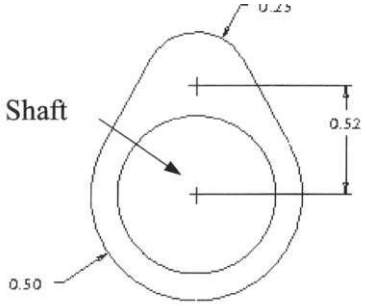


Figure 7-2 The Cam Profile

SolidWorks Parts and Assembly

The cam-follower system consists of seven parts, bushing (two), valve guide, camshaft, pushrod, rocker, and valve, as shown in Figure 7-1. In addition, there are two assembly files, *Lesson7.SLDASM* and *Lesson7withresults.SLDASM* that you may download from the publisher's web site.

We will start with *Lesson* 7. *SLDASM,* in which the parts are adequately assembled. In this assembly the first bushing is anchored (ground) and the second busing and the valve guide are fully constrained. These three parts will be assigned as ground parts. The remaining four parts will be defined as movable parts in *COSMOSMotion.*

Same as before, the assembly file *Lesson 7withresults. SLDASM* contains a complete simulation model with simulation results. You may want to open the assembly to see the motion animation of the mechanism. In the assembly where a motion model is completely defined, a mate has been suppressed to allow movement between components. You can also see how the parts move by right clicking in the graphics screen, choosing *Move Component,* and dragging any movable parts; for example the camshaft to rotate with respect to the second bushing. The whole mechanism will move accordingly.

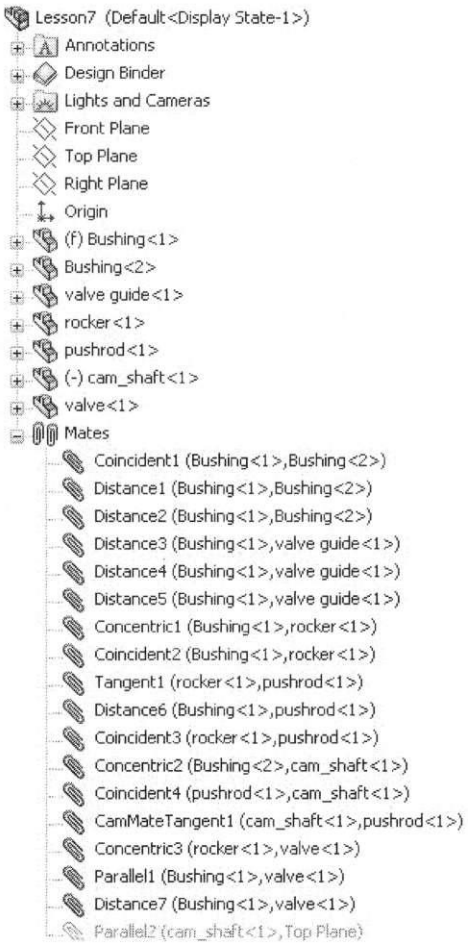


Figure 7-3 Assembly Mates Listed in the Browser

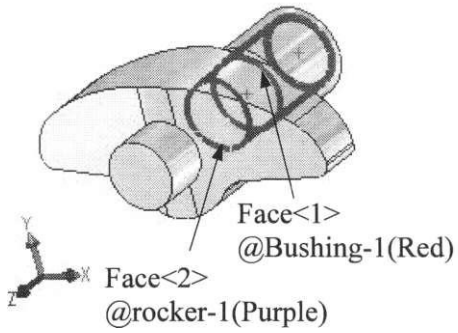
There are eighteen assembly mates, including four coincident, three concentric, seven distance, two parallel, one tangent, and one cam-mate-tangent, as listed in the browser (see Figure 7-3). You may want to expand the *Mates* branch in the browser to review the list of assembly mates. Move the cursor over any of the mates; you should see the entities selected for the assembly mate highlighted in the graphics screen.

As mentioned earlier, the first bushing, *bushing<l>,* shown in the browser is anchored to the assembly. The second bushing *(bushing<2>)* and the valve guide *(valve guide<l>)* were fully assembled to *bushing<l>*. The first three mates, *Coincident 1, Distance 1,* and *Distance2,* were employed to assemble *bushing<2>* to *bushing<l>*. And the next three distance mates assemble the valve guide to *bushing<l>*.

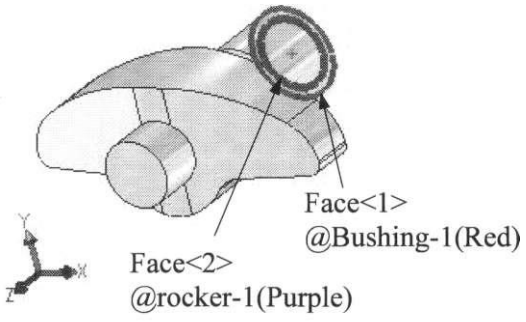
Note that the distance mates are essentially coincident mate with distance between entities. The distance mates were created to properly position the second bushing with respect to the first bushing. All three parts will be defined as the ground part in *COSMOSMotion.*

The next two mates, *Concentric 1* and *Coincident!,* assemble the rocker *{rocker<l>)* to the first bushing *(hushing<l>),* allowing a rotation degree of freedom about the Z-axis, as shown in Figures 7-4a and b. *COSMOSMotion* will map a revolute joint between the rocker and the first bushing.

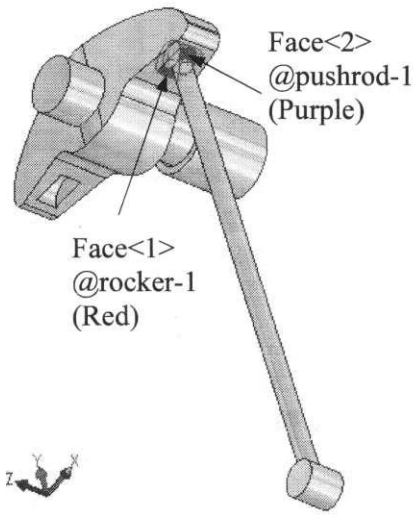
The next part assembled is the pushrod *(pushrod<l>).* The pushrod was assembled to the rocker using *Tangent/, Distance6,* and *Coincident3* mates, as shown in Figures 7-4c, d, and e, respectively. As a result, the pushrod is allowed to move vertically at a distance of *1.25* in. from the *Right* plane of the first bushing *(Distanced),* at the same time, maintaining tangency between the top of the cylindrical surface of the pushrod and the socket surface of the rocker.



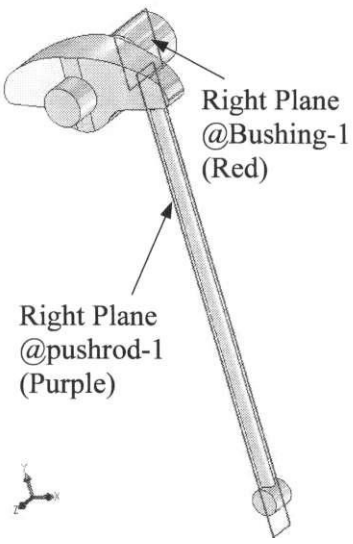
(a) Rocker: Concentric1



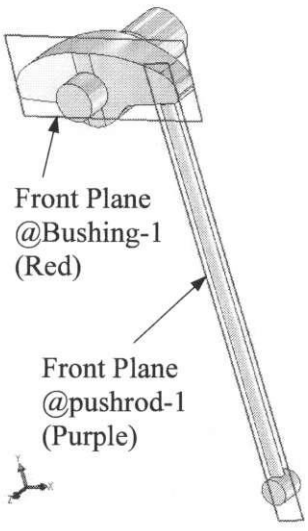
(b) Rocker: Coincident2



(c) Pushrod: Tangentl

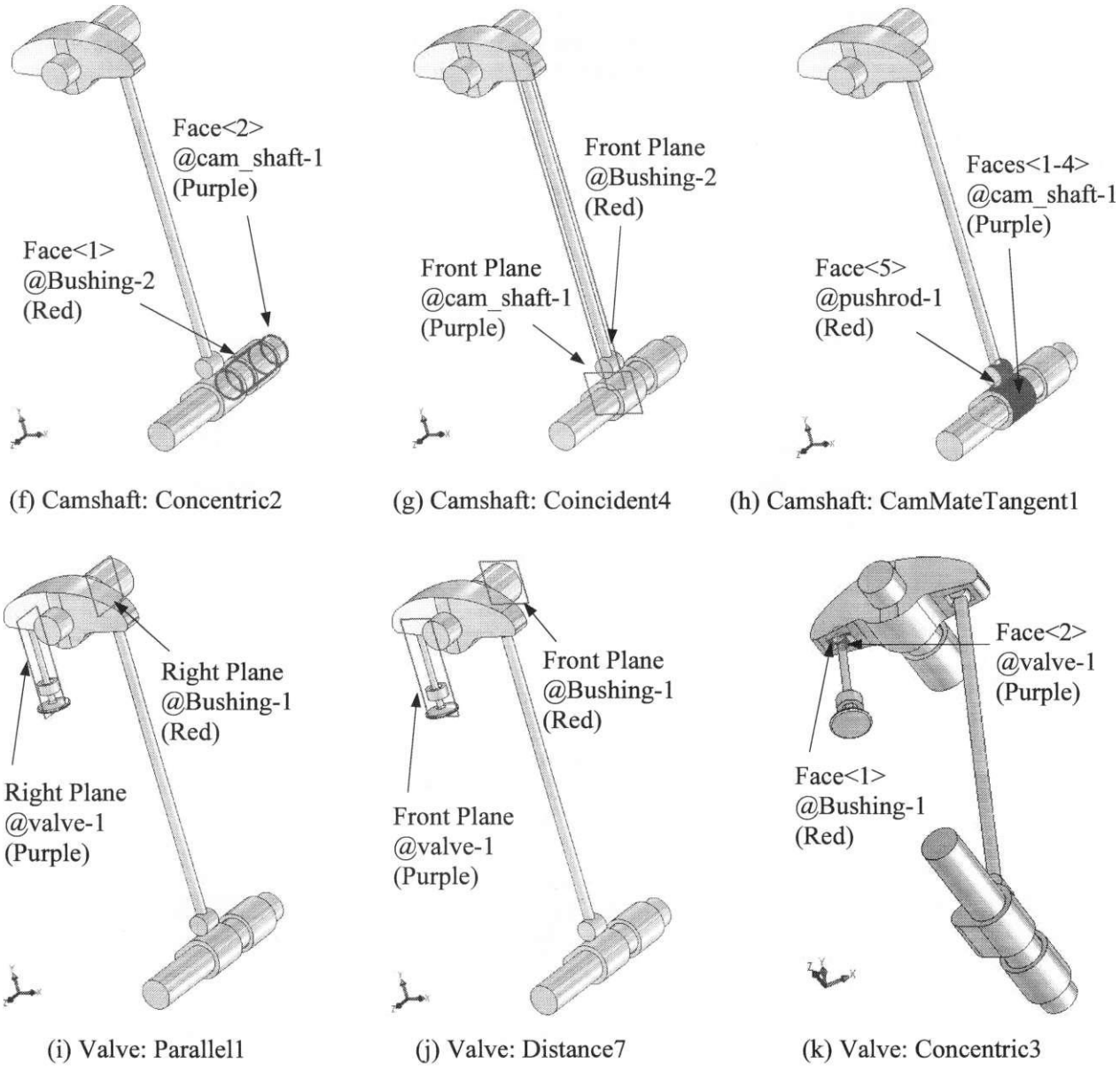


(d) Pushrod: Distance6



(e) Pushrod: Coincident3

Figure 7-4 Assembly Mates Defined for the Mechanism



(f) Camshaft: Concentric2 (g) Camshaft: Coincident4 (h) Camshaft: CamMateTangent1

(i) Valve: Parallel1 (j) Valve: Distance7 (k) Valve: Concentric3

Figure 7-4 Assembly Mates Defined for the Mechanism (Cont'd)

The next part is the camshaft *(cam_shaft<l>).* The camshaft was first assembled to the second bushing *(bushing<2>),* and then to the pushrod. *Concentric2* aligns the camshaft and the second bushing. *Coincident4* mates the center plane of the camshaft to that of pushrod, as shown in Figures 7-4f and g. As a result, the camshaft is allowed to rotate about the Z-axis.

In addition, the *CamMateTangentl* defines a cam and follower between the cam surface of the camshaft and the cylindrical surface at the bottom of the pushrod. Note that all surrounding surfaces on the cam and follower must be selected for the cam-follower joint. In this case, four surfaces are selected for the cam (mounted on the camshaft) and one surface is included on the follower, as shown in Figure 7-4h. Note that the assembly should have overall one degree of freedom at this point. You may either rotate the camshaft, the rocker, or move the pushrod vertically to see the relative motion of the assembly. Use

the *Undo* button to restore the assembly to its original configuration. Note that the last mate *Parallel2* listed in the browser was inactive. This mate was defined to properly orient the camshaft to the *Top* plane of the assembly. This mate is mainly for setting up an initial condition for simulation.

The next and the final part assembled is the valve. The valve was assembled to the first bushing using *Parallel1* and *Distance7* mates, which restrict the valve to move vertically on the *X-Y* plane, as shown in Figures 7-4i and j. The third mate, *Concentric3*, aligns the top cylindrical surface of the valve with the other socket surface of the rocker, as shown in Figure 7-4k. Note that the valve will move vertically along the *Y*-direction, and slightly off the nominal distance of *1.25* in. between its *Right* plane and the *Right* plane of the first bushing. This slight movement is necessary due to the rotation of the rocker. That is why the mate *Parallel1* (between the *Right* plane of the valve and the *Right* plane of the first bushing) is defined instead of a distance mate.

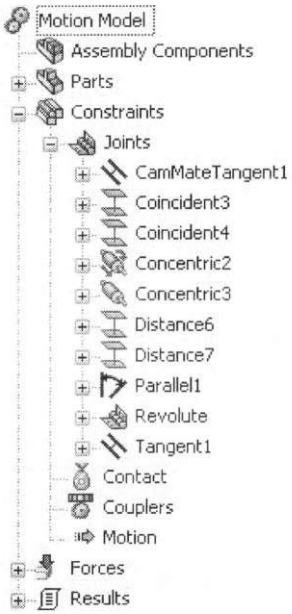


Figure 7-5

Note that this set of assembly mates employed is not the only way to assemble the individual parts for a valid motion model. You may try different combinations to mate these parts. As long as the assembly is physically valid, *COSMOSMotion* is usually able to simulate the motion of a valid assembly that is physically meaningful.

Simulation Model

In this example, *COSMOSMotion* carries over nine assembly mates and converts two mates (*Concentric1* and *Coincident2*) to a revolute joint (between the rocker and the first bushing), as listed in the browser (see Figure 7-5). The total number of degrees of freedom of the cam-follower mechanism can be calculated as follows:

4 (bodies) × 6 (dof's/body) – 1 (CamMateTangent) × 1 (dof/CamMateTangent) – 2 (coincident) × 3 (dof's/coincident) – 2 (concentric) × 4 (dof's/concentric) – 2 (distance) × 3 (dof's/distance) – 1 (parallel) × 2 (dof's/parallel) – 1 (revolute) × 5 (dof's/revolute) – 1 (tangent) × 1 (dof/tangent)
= 24 – 29 = –5

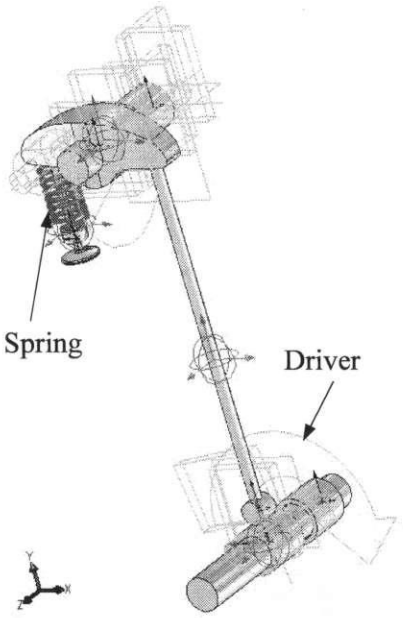


Figure 7-6 The Simulation Model

Apparently, this result implies that there are redundant dofs created in the system. This is fine since *COSMOSMotion* filters out the redundant dofs. You may want to check the redundancy by choosing, from the pull-down menu, *COSMOSMotion* > *Show Simulation Panel*, as discussed in Appendix A. You may want to review Appendix A for more information about defining joints and calculating degrees of freedom.

The final motion model is shown in Figure 7-6, where the Z-rotation of the concentric joint, *Concentric2*, between the camshaft and the second bushing is driven by a constant angular velocity of

3,600 degrees/sec; i.e., 600 rpm, about the Z-axis of the global coordinate system. In addition, a spring surrounding the valve will be created in order to provide a vertical force to push the rocker up, therefore, close the valve. The spring has a spring constant of *10* lbf/in and an unstretched length of *1.25* in. The spring is created between the bottom face of the rocker and top face of the valve guide.

7.3 Using *COSMOSMotion*

Start *SolidWorks* and open assembly file *Lesson 7. SLDASM.*

From the browser, click the *Motion* button if to enter *COSMOSMotion.*

Again, always check the units system. Make sure that *IPS* units system is chosen for this example.

Defining Bodies

From the browser, expand the *Assembly Components* branch. You should see seven entities listed, *Bushing-1, Bushing-2, cam_shaft-1, pushrod-1, rocker-7, valve guide-1,* and *valve-1,* as shown in Figure 7-7. Also expand the *Parts* branch; you should see *Moving Parts* and *Ground Parts* listed. We will move *Bushing-1, Bushing-2,* and *valve guide-1* to *Ground Parts* and the remaining four parts to *Moving Parts* by using the drag-and-drop method.

From the browser, click *Bushing-1.* Press the *Ctrl* key and click *Bushing-2* and *valve guide-1.* All three parts should be selected. Drag and drop them to the *Ground Parts* node.

Repeat the same to select the remaining four parts under the *Assembly Component* branch. Drag and drop them to the *Moving Parts* node.

Expand the *Constraints* branch, and then the *Joints* branch. You should see that ten joints are listed (see Figure 7-5). All joint symbols should appear in the graphics screen, similar to that of Figure 7-6. Expand all joints in the browser and identify the parts they connect. Take a look at the joint *Coincident2* (connecting camshaft to *Bushing-2),* where we will add a driver next.

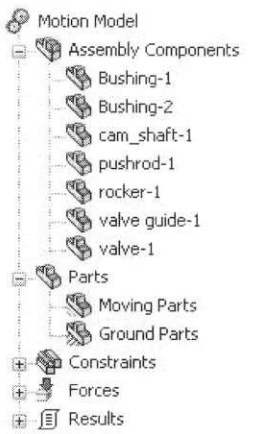


Figure 7-7

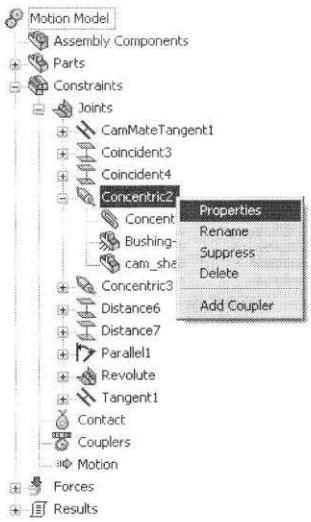


Figure 7-8

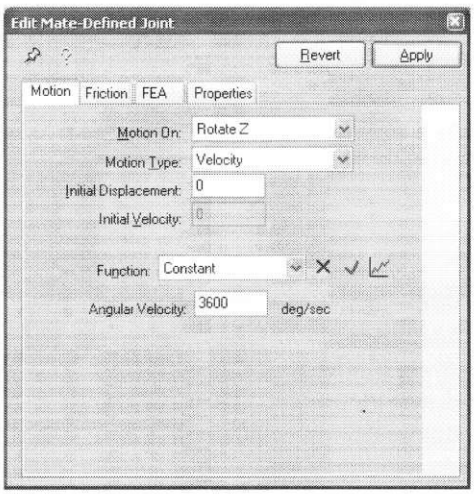


Figure 7-9 Adding Motion Driver to Joint *Concentric2*

Driving Joint

Right click the *Concentric2* node and choose *Properties* (see Figure 7-8). In the *Edit Mate-Defined Joint* dialog box (Figure 7-9), under the *Motion* tab (default), choose *Rotate Z* for *Motion On,* choose *Velocity* for *Motion Type,* choose *Constant* for *Function,* and enter *3600* degrees/sec for *Angular Velocity,* as shown in Figure 7-9. Click *Apply* to accept the definition.

Defining Spring

From the browser, expand the *Forces* branch, right click the *Spring* node and choose *Add Translational Spring* (see Figure 7-10). In the *Input Spring* dialog box (Figure 7-11), the *Select 1st Component* field should be highlighted in red and ready for you to pick. Rotate the view and pick the bottom face of the rocker (see Figure 7-12), the *Select 2nd Component* field should now highlight in red, and *rocker-1/DDMFace 19* should appear in the *Select Point on 1st Component* field, which indicates that the spring will be connected to the center point of the face selected.

In case you picked a wrong entity, simply select the entire text in the respective text field in the dialog box, and press the *Delete* key to delete the text. The text field will turn back to red and will be ready for you to pick another entity.

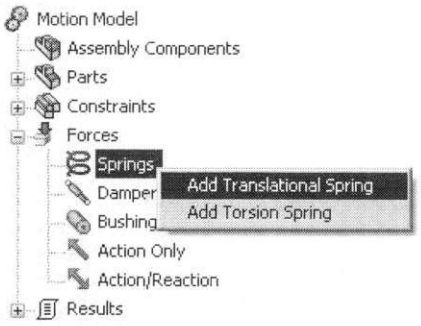


Figure 7-10

Rotate the view back, and then pick the top face in the valve guide, as shown in Figure 7-12. Now, *valve guide-1* and *valve guide-l/DDMFace20* should appear in the *Select 2nd Component* field and *Select Point on 2nd Component* field, respectively. Also, a spring should appear in the graphics screen, connecting the center points of the two faces.

Enter the followings:

Stiffness: 10
Length: 1.25 (Note that you have to deselect the *Design* box to the right before entering this value)
Force: 0
Coil Diameter: 0.75
Number of coils: 8
Wire Diameter: 0.1

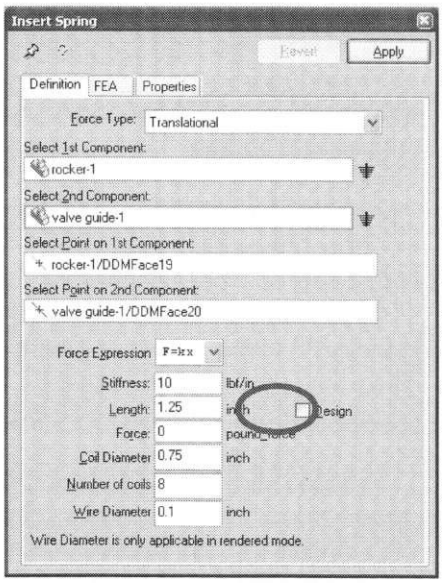


Figure 7-11 The *Insert Spring* Dialog Box

Click *Apply* to accept the spring definition and close the *Insert Spring* dialog box.

Defining and Running Simulation

Click the *Motion Model* node, press the right mouse button and select *Simulation Parameters.* Enter *0.5* for simulation duration and *500* for the number of frames.

Click the *Motion Model* node again, press the right mouse button and select *Run Simulation.* You should see that the camshaft starts rotating, the pushrod is moving up and down, which drives the rocker,

and then the valve. The camshaft rotates 5 times in the 0.5-second simulation duration. We will graph the position, velocity, and acceleration of the valve next.

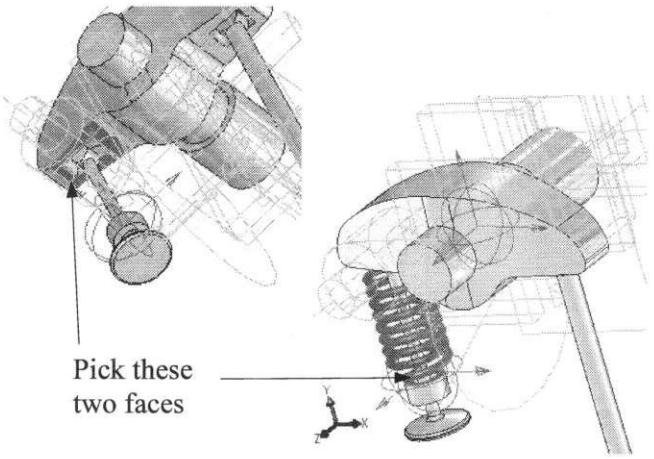


Figure 7-12 Picking Two Faces to Define the Spring

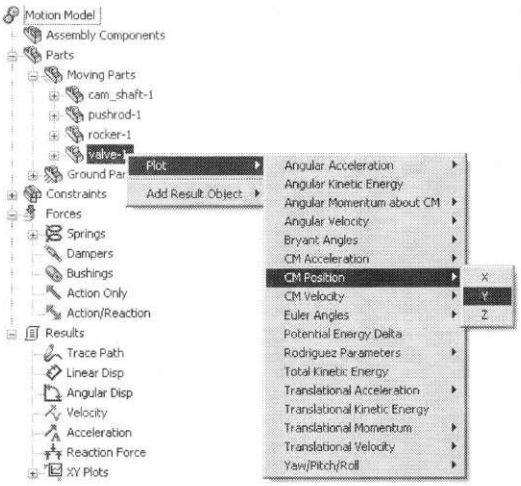


Figure 7-13

Displaying Simulation Results

From the browser, expand the *Parts* branch, and then the *Moving Parts* branch. Right click the *valve-1* node, and choose *Plot > CM Position > Y* (see Figure 7-13). The graph of the *Y*-position of the mass center of the valve will appear, similar to that of Figure 7-14, where the valve is moving between *–1.96* and *–1.70* in., traveling about *0.26* in., which is about what was expected, as discussed in Section 7.2.

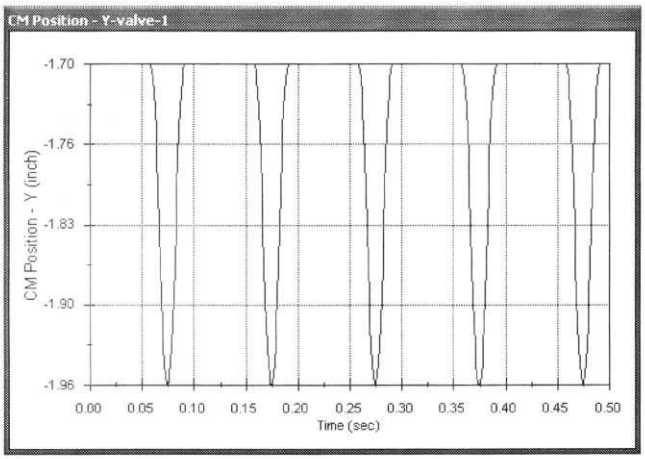


Figure 7-14 Graph of Valve Position

As shown in Figure 7-14, the flat portion on top indicates that the valve stays completely closed, which spans about *0.066* seconds, approximately *240* degrees of the camshaft rotation in a complete cycle. Therefore, the valve will open for about *0.034* seconds per cycle, roughly *120* degrees.

Graph the 7-velocity and 7-acceleration of the valve by choosing *Plot > CM Velocity* (and *CM Acceleration) > Y Component.* The graphs of the velocity and acceleration are shown in Figures 7-15 and 16, respectively. As shown in Figure 7-15, there are two velocity spikes per cycle, representing that the valve is pushed downward (negative velocity) for opening and is being pulled back (positive velocity) for closing, respectively. The valve stays closed with zero velocity.

Figure 7-16 reveals high accelerations when the valve is pushed and pulled. Note that such a high acceleration is due to high-speed rotation at the camshaft. This high acceleration could produce large inertial force on the valve, yielding high contact force between the top of the valve and the socket surface in the rocker. We would like to check the reaction force between the top of the valve and the rocker. The

graph of the reaction force can be created by expanding *Constraints* and *Joints* branches, right clicking *Concentric2* (between the valve and the rocker), and choosing *Plot > Reaction Force > Y Component.*

The reaction force graph (Figure 7-17) shows that the reaction force between the top of the valve and the socket face of the rocker is about *0.4* lb_f, which is insignificant. Note that this small reaction force can be attributed to the small mass of the valve. If you open the valve part and acquire its mass (from pull-down menu, choose *Tools > Mass Properties),* the mass of the valve is *0.03* lb_m. Therefore, the inertia for the valve at the peak accelerations is about *0.03* $\backslash b_m x 5{,}600$ in/sec^2 = *168* lb_m in/sec^2 = *168/386* lb_f = *0.44* lbf, which is consistent to peaks found in Figure 7-17. If you are not quite sure about why this *386* is factored in for force calculation, please refer to Appendix B for mass and force unit conversions. Save your model.

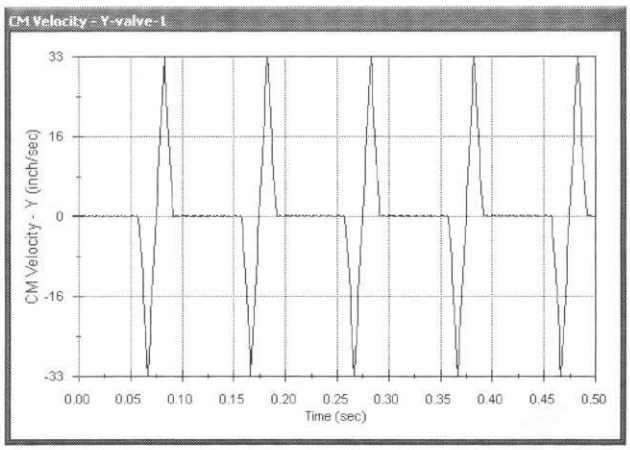


Figure 7-15 Graph of Valve Velocity

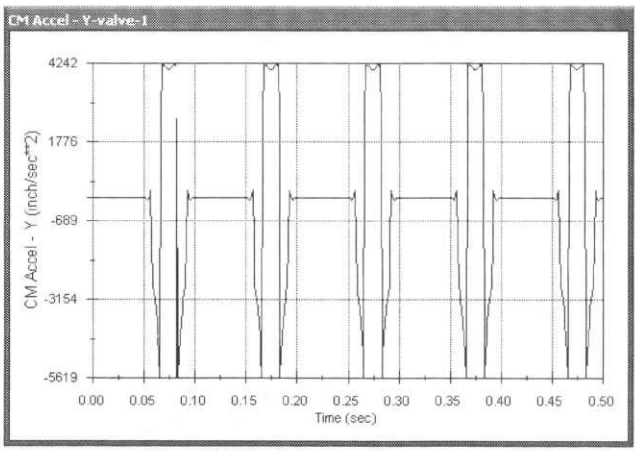


Figure 7-16 Graph of Valve Acceleration

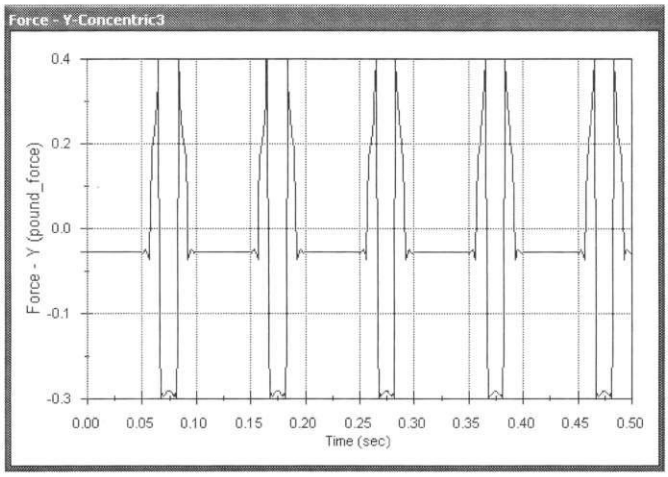


Figure 7-17 Graph of the Reaction Force

Exercises:

1. Redesign the cam by reducing the small arc radius from *0.25* to *0.2* and reducing the center distance of the small arc from *0.52* to *0.40,* as shown in Figure E7-1. Repeat the dynamic analysis and check reaction force between the valve and the rocker. Does this redesigned cam alter the reaction force?

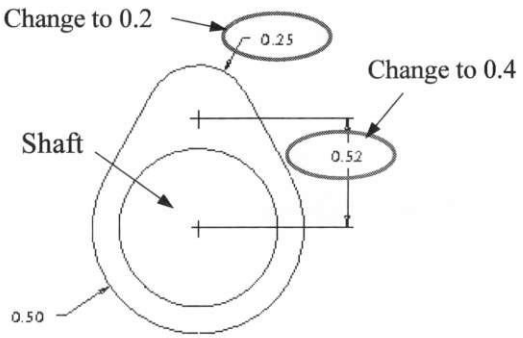
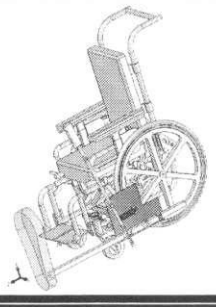


Figure E7-1 The Cam Profile

2. If we change the *Parallel!* mate between the *Right* plane of the valve and the *Right* plane of the first bushing to a distance mate, will the mechanism move? What other changes must be made in order to create a valid and movable mechanism similar to that was presented in this lesson?

Lesson 8: Assistive Device for Wheelchair Soccer Game



8.1 Overview of the Lesson

This is an application lesson. We will apply what we learned in previous lessons to a real-world application. This application involves designing a device that can be mounted on a wheelchair to mimic soccer ball-kicking action while being operated by a child sitting on the wheelchair with limited mobility and hand strength. Such a device will provide more incentive and realistic experience for children with physical disabilities to participate in soccer games. This example was extracted from an undergraduate student design project that was carried out in conjunction with a local children hospital. This device was intended primarily to be used in the summer camp sponsored by the children hospital.

The focus of this lesson is slightly different from previous ones. Instead of focusing on discussing how to use *COSMOSMotion* to create motion entities, we will focus on how to use *COSMOSMotion* to support design. In order to narrow down the design options to be more manageable, we will assume all major components are designed with dimensions determined. More specifically, we will use *COSMOSMotion* to help choose a spring, as well as determine if the required operating force is acceptable. Since the users of this device are children with limited physical hand strength, the operating force must be minimized in order to make the device useful.

The examples we have discussed in previous lessons are simple enough so that some of the simulation results can be verified by hand calculations (for example, using a spreadsheet). However, most of the real-world applications, including the example of this lesson, are too complicate to verify by hand calculations. When we are dealing with such applications, two principles are helpful in leading to successful simulations. First, the simulation model you created has to be physically meaningful and is as consistent to the physical conditions as possible. Second, very often you will have to make assumptions in order to simplify the problems so that the simulations can be carried out. This is because that the simulation models must comply with the ability of the software you are using. In order to effectively use the simulations to support design, you will have to understand the physical problems very well; in the mean time, be familiar with the capabilities and limitations of the software you are using.

Since most of you will be often learning the software as you are tackling simulation and/or design problems, it is strongly recommended that you employ the principle of spiral development to incrementally build up your simulation model. In another word, you may want to start from a simplified model with simple scenarios by making adequate assumptions to your simulation model. Make the simplified model works first, then relax the assumptions and add motion entities to make your model closer to the real situation. Repeat the process until you reach a simulation model and simulation scenarios that answer you questions and help you make design decisions. In each step, make sure that the simulation model does what you expect it to do before bringing it to the next level.

In this lesson, we will employ the spiral development principle. We will assume that all components and their physical dimensions are determined. The design is essentially narrowed down to the selection of

a spring, including both spring constant and free length, and to ensure that the required force is small enough for a child to easily operate the device. We will try our best to check and hopefully verify the simulation model in each step. Note that *COSMOSMotion* is very sensitive to some of the conditions and parameters, such as the initial condition (that is, the handle bar orientation), spring constant, etc. Some of the conditions simulated are physically meaningful and yet *COSMOSMotion* gives unrealistic simulation results due to its limitations. Again, *COSMOSMotion* is not foolproof. One cannot blindly accept the simulation results. When a result is determined unrealistic after reviewing animation, graphs, etc., the best way to proceed is to compose a simpler simulation model and/or try a different (more idealized) scenario until the simulation result is physically meaningful based on your educated judgment. You will see trials-and-errors in this lesson and many other real-world applications in the future.

8.2 The Assistive Device

Physical Model

This assistive device for soccer games consists of five major components: the clamper, handle bar, plate, kicking-rod, and spring, as illustrated in Figure 8-1. In reality these five components will be assembled first and clamped to the lower frame of the wheelchair for use.

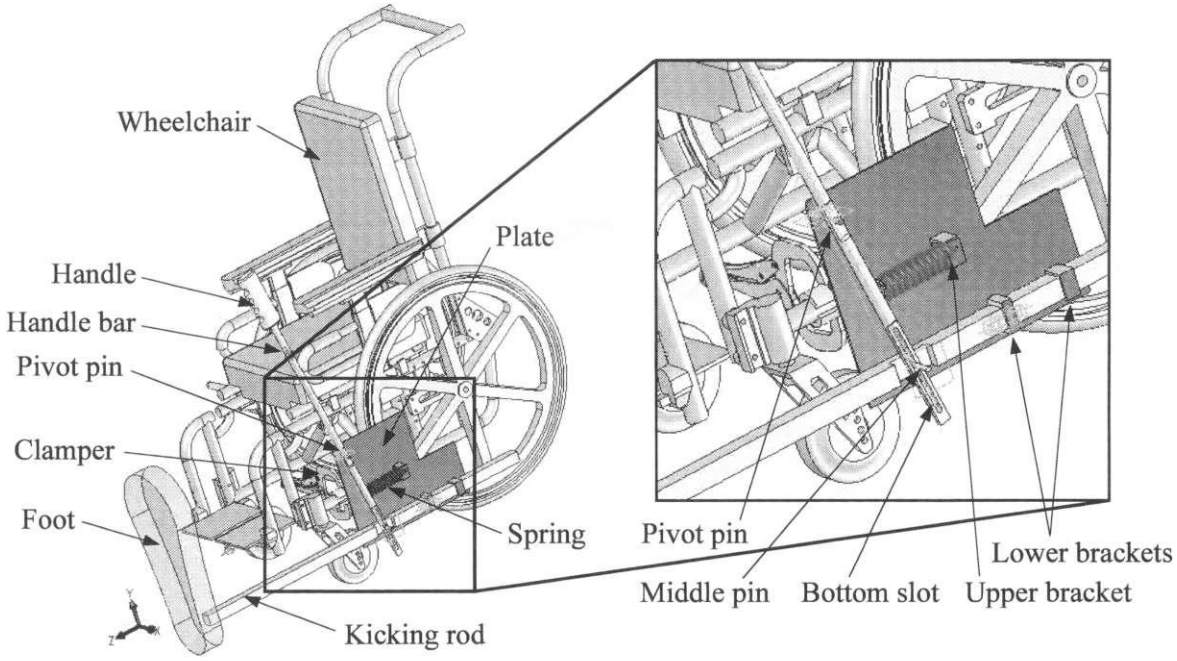


Figure 8-1 Assistive Device for Soccer Game

The handle bar is mounted to the plate at the pivot pin of the plate and linked to the middle pin of the kicking rod. The kicking rod is inserted into the two lower brackets mounted on the plate. When the handle (on top of the handle bar) is pulled backward, the handle bar rotates about the pivot pin; therefore, drives the kicking rod to move forward along the longitudinal direction through the link between the bottom slot of the handle bar and the middle pin of the kicking rod. The forward movement of the kicking rod produces momentum to "kick" the soccer ball. A spring is added between the upper bracket and the handle bar to restore the handle bar to its neutral position after pulling. The spring also helps the user to pull the handle bar with a lesser force.

The focus of this lesson is to use *COSMOSMotion* to simulate the position and velocity of the kicking rod for a given force that can be comfortably provided by a child with limited physical strength. Lots of factors contribute to the operating force of the mechanism. For example, one of the critical parameters is the location of the pivot pin. The lower the pivot pin is located the lesser force is required to operate the mechanism. However, the purpose of this lesson is not necessarily to determine the final design of the device, but illustrate the process of using *COSMOSMotion* to assist the design. Therefore, as mentioned earlier, the scope of the design has been narrowed down to the selection of the spring and to determine if the force is small enough for a child to operate the device.

SolidWorks Parts and Assembly

The assembly of the mechanism consists of eight parts and one subassembly. These parts are *handle.SLDPRT, plate. SLDPRT, rod.SLDPRT, foot.SLDPRT, clamper. SLDPRT, joint.SLDPRT, collar. SLDPRT,* and *wheelchair.SLDPRT.* The subassembly is *kickingjrod.SLDASM* which consists of *rod.SLDPRT m& foot.SLDPRT.*

In addition, there are six assembly files, *Lesson8. SLDASM, Lesson8TaskOne.SLDASM, Lesson8TaskTwo.SLDASM, Lesson8TaskThreeNoFriction.SLDASM, Lesson8TaskThreeSmallFriction. SLDASM, mdLesson8TaskThreeLargeFriction. SLDASM.* You can download these files from publisher's web site.

Same as before, the assembly files, with the exception of *Lesson8.SLDASM,* consist of complete simulation models with simulation results under respective simulation scenarios. You may want to open these files to see the motion animations of the mechanism. In these assembly files, a mate *(Anglel)* has been suppressed. You can also see how the parts move by right clicking in the graphics screen, choosing *Move Component,* and dragging any movable parts; for example dragging the handle to drive the kicking rod.

There are fifteen assembly mates, including four coincident, three concentric, four distance, two parallel, and one angle, defined in the assembly, as listed in the browser (see Figure 8-2). You may want to expand the *Mates* branch in the browser to see these assembly mates. Move your cursor over any of the mates; you should see the entities chosen for the assembly mate highlighted in the graphics screen.

The first nine mates assemble the clamper to the wheelchair, the joint to the clamper and the wheelchair, and then the plate to the joint part. Note that the joint is a *SolidWork* part that connects the plate and the clamper rigidly. These nine mates are pretty standard. All these parts are fully constrained and are fixed to the wheelchair.

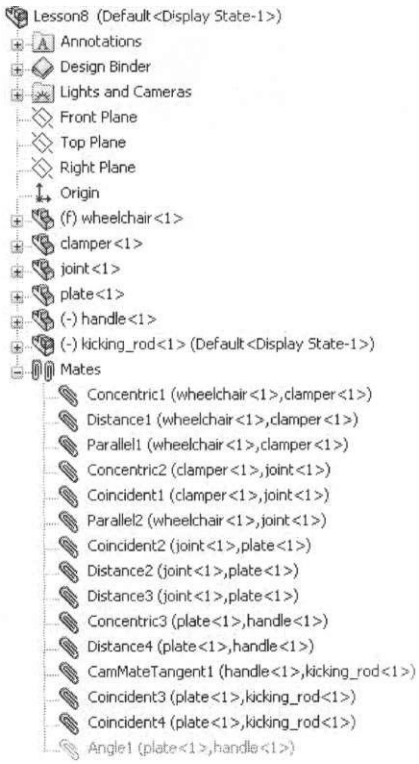
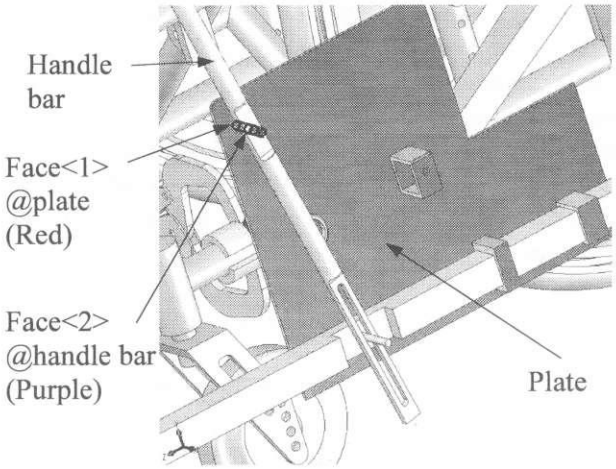
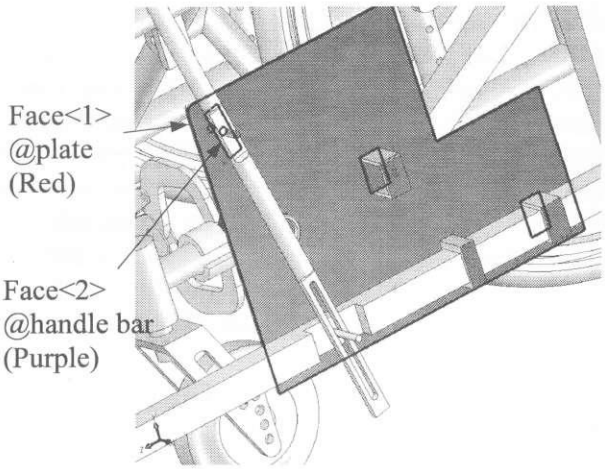


Figure 8-2 Assembly Browser

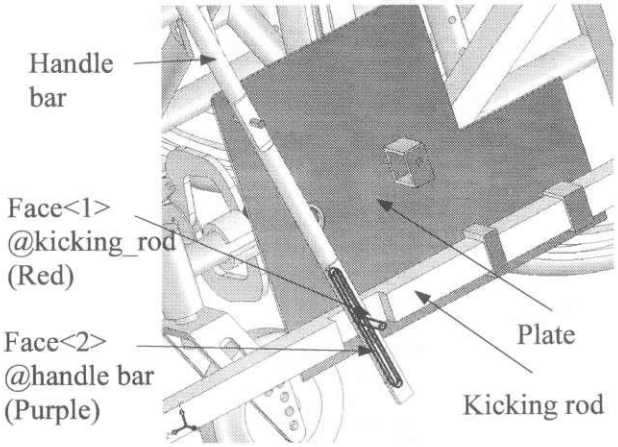
The next two mates, *Concentric3* and *Distance4,* assemble the handle bar to the plate at the pivot pin, allowing the handle bar to rotate at the pivot pin, as shown in Figure 8-3a. *COSMOSMotion* will convert these mates to a revolute joint. The distance mate provides an adequate clearance between the handle bar and the plate, as shown in Figure 8-3b.



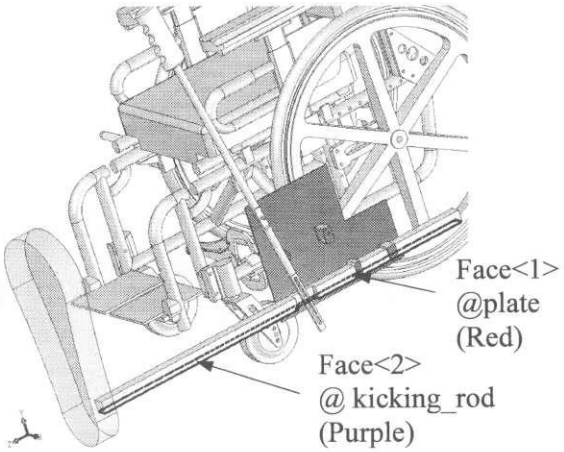
(a) Handle bar: Concentric3



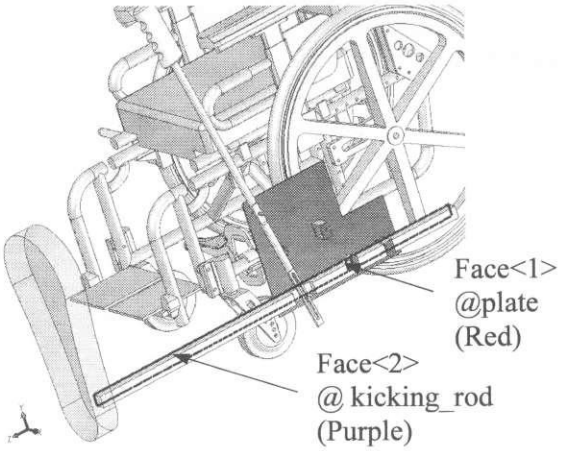
(b) Handle bar: Distance4



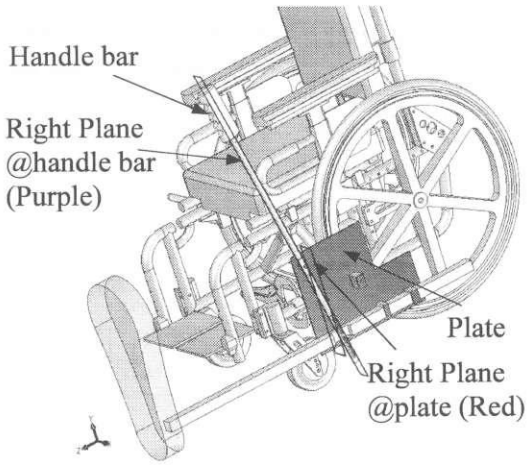
(c) Kicking rod: CamMateTangent1



(d) Kicking rod: Coincident3



(e) Kicking rod: Coincident4

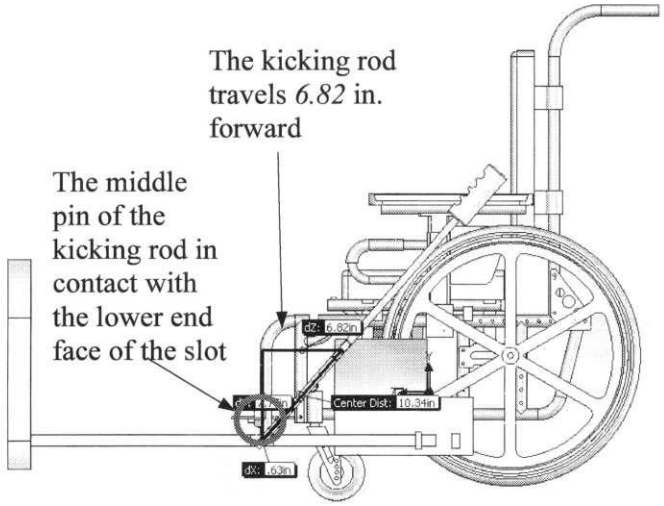


(f) Handle bar: Angle1

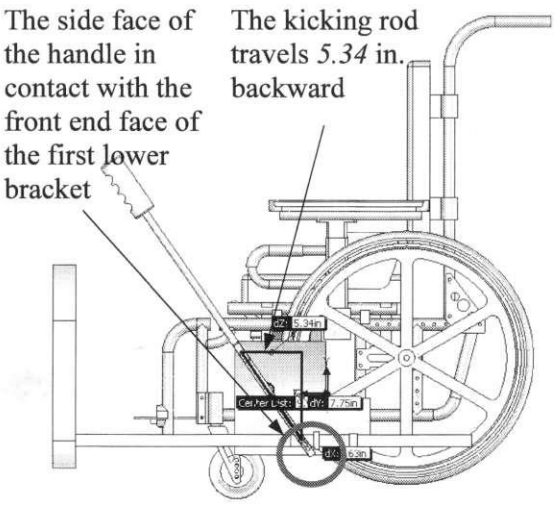
Figure 8-3 Assembly Mates Defined in *Lesson8.SLDASM*

The next three mates, *CamMateTangentl, Coincident3,* and *Coincident4,* assemble the kicking rod to the plate and the handle bar. First, the middle pin of the kicking rod is assembled to the inner surface of the bottom slot of the handle bar using *CamMateTangentl,* as shown in Figure 8-3c. As a result, the pin can only move within the slot, which is desirable. Second *Coincident3* mates the bottom face of the kicking rod to the inner bottom face of the first lower bracket of the plate, as shown in Figure 8-3d. Similarly *Coincident4* mates the rear face of the kicking rod to the inner side face of the first lower bracket of the plate, as shown in Figure 8-3e. These two mates restrict the kicking rod to slide along the longitudinal direction, which will be converted to a translational joint by *COSMOSMotion.*

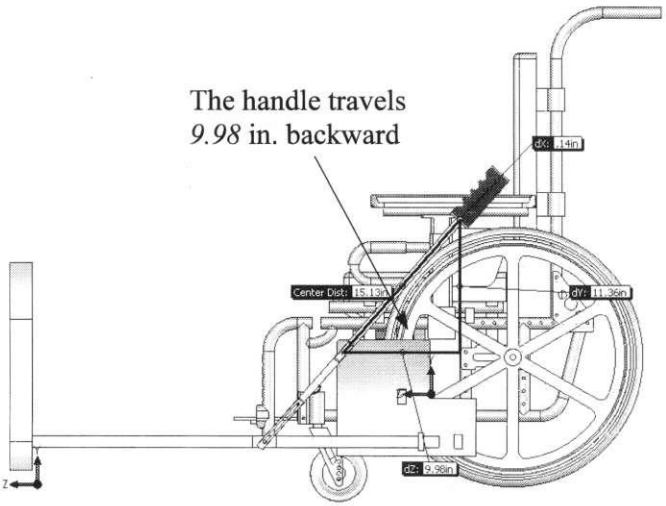
The final mate, *Anglel,* orients the vertical plane of the handle bar *{Right Plane)* respect to that of the plate *{Right Plane).* This mate will help determine an initial condition for motion simulations. Note that the *Anglel* mate to orient the handle, it will has to be suppressed to allow the handle bar to rotate



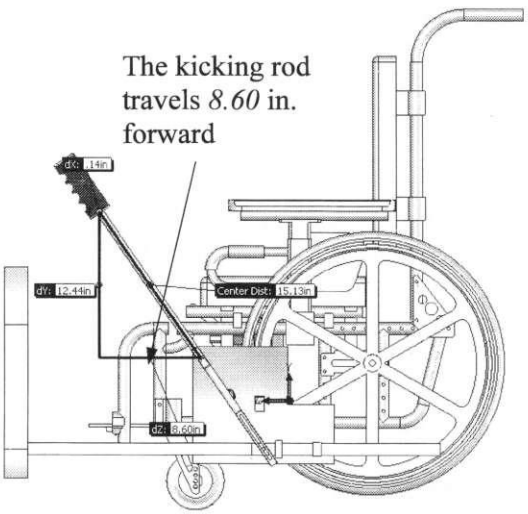
(a) Kicking rod moving forward *6.82* in.



(b) Kicking rod moving backward *5.34* in.



(c) Handle moving backward *9.98* in.



(d) Handle moving forward *8.60* in.

Figure 8-4 Travel Distances of the Kicking Rod and the Handle Bar

Based on the geometry of the mechanism and dimensions of its constituent components, the kicking rod is able to travel a total of *12.2* in. along the longitudinal direction (Z-direction). The kicking rod can move *6.82* in. forward (positive Z-direction with respect tot the middle pin) until its middle pin becomes in contact with the inner face at the lower end of the slot, as shown in Figure 8-4a. Similarly, the kicking rod moves *5.34* in. backward until the side face of the handle bar becomes in contact with the front end face of the first lower bracket, as shown in Figure 8-4b.

At the same time, the handle will travel a total of *18.6* in., *9.98* backward and *8.6* in. forward, as shown in Figures 8-4c and 8-4d, respectively. Note that these measurements shown in Figure 8-4 can be obtained by using the *Measure* option in *SolidWorks.* To access the *Measure* option, simply choose from the pull-down menu *Tools* > *Measure.*

Simulation Model

In this motion model, the only moving parts are the handle bar and the kicking rod. After suppressing the mate *Angle1, COSMOSMotion* will add a revolute joint to the motion model between the handle bar and the plate at the pivot pin, as shown in Figure 8-5. The handle bar is allowed to rotate about the X-axis of the global coordinate system at the pivot pin. In addition, a translational joint is created by *COSMOSMotion* between the kicking rod and the first lower bracket. This joint constrains the kicking rod to translate along the longitudinal direction; i.e., the Z-direction. The third and final joint is the *CamMateTangent* between the outer surface of the middle pin and the inner surface of the slot at the bottom of the handle bar. This joint restricts the pin to move inside the slot.

In addition to joints, a spring is added between the upper bracket of the plate and the hook on the side of the handle bar. This spring, as shown in Figure 8-5, is added to restore the handle bar to a neutral position after pulling. Since the free spring length is set to a slightly larger value so that the handle bar will be oriented at a negative 5~/0-degree angle (about the X-axis). Therefore, the user will have to push the handle forward before pulling it back to kick the ball. A larger spring free length will push the handle leaning further backward (toward the user), providing additional pulling force that helps the users to pull back the handle bar. Note that the pulling action will push the kicking rod forward (positive Z-direction) to "kick" the soccer ball.

There is one contact constraint added to the motion model. The contact constraint is defined between the outer side face of the handle bar and the front end face of the first lower bracket. This contact constraint will prevent the handle bar from moving further backward and penetrating through the brackets. In this contact constraint, a restitution coefficient of *0.5* is assumed.

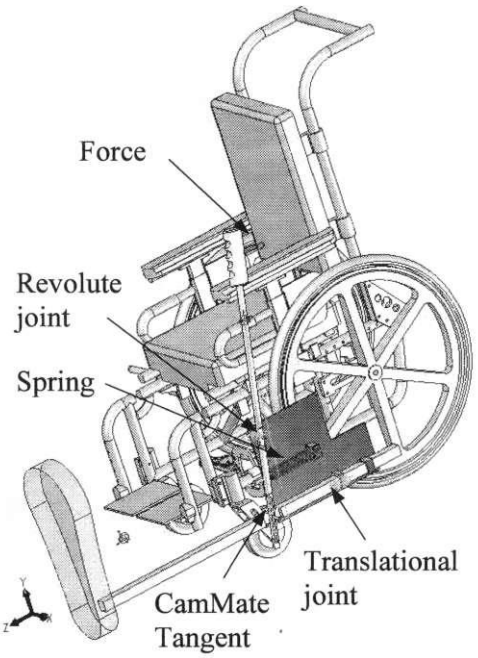
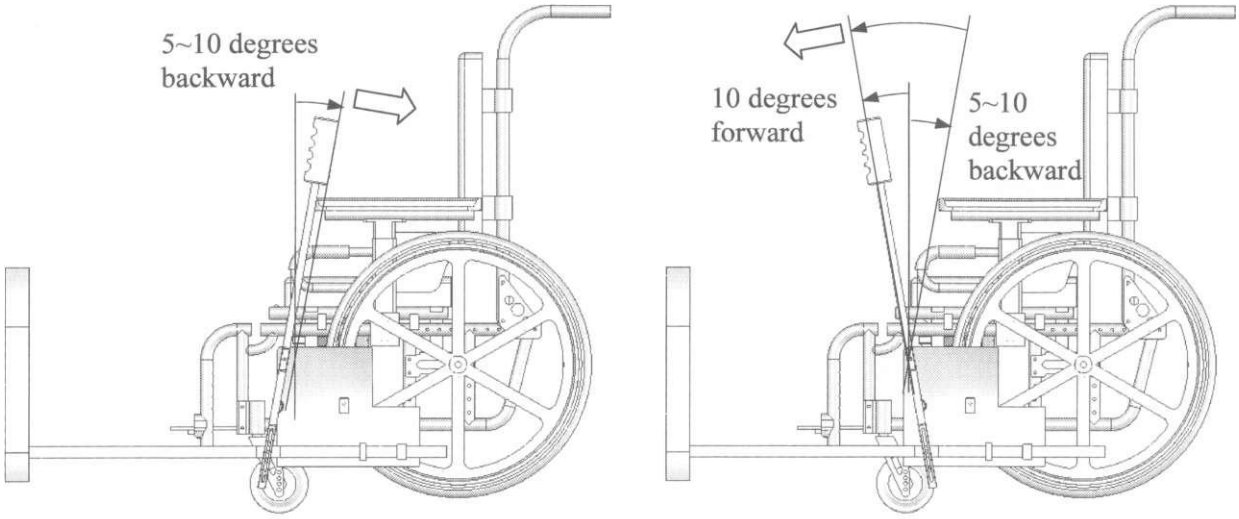


Figure 8-5 Motion Model of the Mechanism

Finally, an impulse force of *5-65* lb_f in a time span of *1.0* second will be added to the handle bar along the Z-direction, as shown in Figure 8-5, to simulate the operating force. Note that the impulse force must first push the handle bar about *10* degrees forward before pulling it back, as illustrated in Figure 8-6.



(a) Handle Bar Moved Backward Due to Spring (b) Handle Bar Pushed Forward Due to Force

Figure 8-6 Positions of the Handle Bar During Operation

In this example, we will turn on the gravity, which is acting in the negative 7-direction (vertically downward); i.e., the default setting. We will first assume no friction in any joints. This assumption will greatly simplify the simulation model and help set up the motion model correctly. We will use this model to choose a spring, including the selection of spring constant and free length. The simulation results of this non-friction model have been created in the assembly files *Lesson8TaskOne.SLDASM* and *Lesson8TaskTwo. SLDASM.*

The friction force will be turned on for both the revolute and translational joints to determine the required force for operating the mechanism. In addition, the operating force, modeled as an impulse force will be added to the mechanism. Note that the friction is not added to the *CamMateTangent* joint between the middle pin and the slot since such a capability is not currently supported by *COSMOSMotion.* Results of these simulations can be found in *Lesson8TaskThreeNoFriction.SLDASM, Lesson8TaskThreeSmallFriction.SLDASM,* and *Lesson8TaskThreeLargeFriction.SLDASM.*

8.3 Using *COSMOSMotion*

In this example, we will start with a valid simulation model defined in the assembly *Lesson8.SLDASM.* The unit system is *IPS.* When you open this assembly file, you should see a properly assembled model with fifteen mates, as shown in Figure 8-2, where the last constraint, *Angle1,* should have been suppressed. Note that the mate angle is set to positive *10* degrees for the time being, and the handle bar is leaning forward (toward the Z-direction), as shown in Figure 8-7. Note that this angle is what we assume for Task One simulations.

If you enter *COSMOSMotion* and expand the *Parts* and *Constraints* branches in the browser, you should see the existing motion entities, as shown in Figure 8-8. There are four parts under the *Ground Parts* branch. Only handle bar and kicking rod are movable. In addition, there are three joints defined, *CamMateTangentl, Revolute,* and *Translational,* as discussed earlier. In the *Contact* branch under the *Constraints,* there is one contact joint, *Contact 3D,* defined to prevent the kicking rod from penetrating into the brackets. Due to the contact constraint and the restitution coefficient defined, the handle bar will bounce back when it hits the front edge of the first lower bracket.

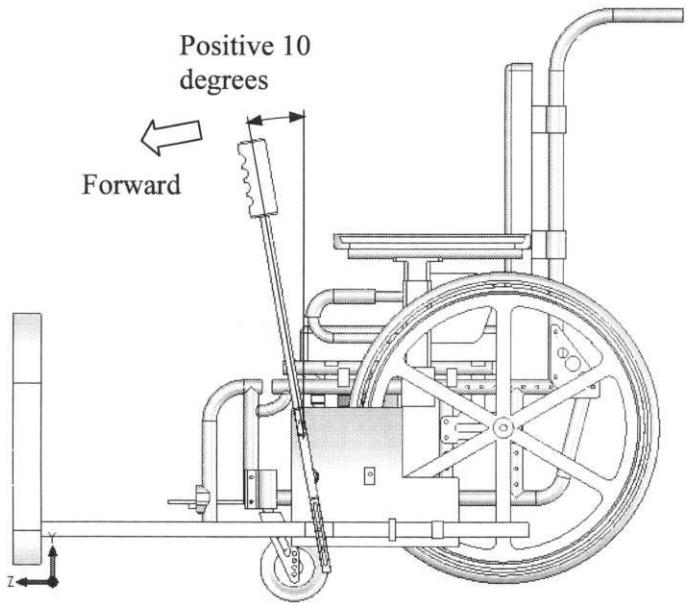


Figure 8-7 Initial Orientation of the Handle Bar For *TaskOne* Simulations

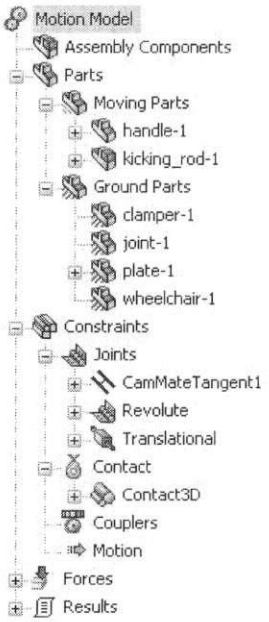


Figure 8-8

There are three major tasks we will carry out in this design. In Task One we will run simulation to check the function of the *3D contact* constraint. We hope to ensure that all joints are adequately defined, and the restitution coefficient we assumed in the contact constraint gives us reasonable results.

In Task Two, we will add the spring. We will adjust the spring constant and its free length until we reach an equilibrium configuration that we can work with. Note that a desired free length should bring the handle bar backward a negative *5-10* degree angle (about the X-axis). After the spring is determined, we will add an operating force at the handle in Task Three. Note that the force will first push the handle bar forward about *10* degrees (positive, about the X-axis) before pulling it backward in order to provide enough travel distance for the kicking rod, and hopefully, sufficient momentum to kick the ball.

In Task Three, we will start with a non-friction case, and then turn on friction in both the revolute and translational joints. From the friction cases, we will determine if the operating force is sufficient to push the handle forward about *10* degrees before pulling it backward. We will vary the friction coefficients to simulate different scenarios and then determine the magnitude of the operating forces accordingly. The magnitude of the force will answer the critical question: if the design is acceptable. We hope to keep the maximum force magnitude under *20* lbf.

Task One: Determining Contact Constraints

We will carry out simulations for the motion model defined in *Lesson8.SLDASM*. In the assembly, the handle is leaning forward *10* degrees (see Figure 8-7), the contact joint *Contact 3D* is defined, and the gravity is turned on. You may want to open the *Contact3D* constraint by right clicking *Contact3D* from the browser and choosing *Properties.* In the *Edit 3D Contact* dialog box, you should see that the plate and handle are included in the first and second containers, respectively, as shown in Figure 8-9. If you choose the *Contact* tab, you should see that the *Coefficient of Restitution* is 0.5, as shown in Figure 8-10. Note that no friction is imposed for this constraint. Close the dialog box by clicking the *Apply* button.

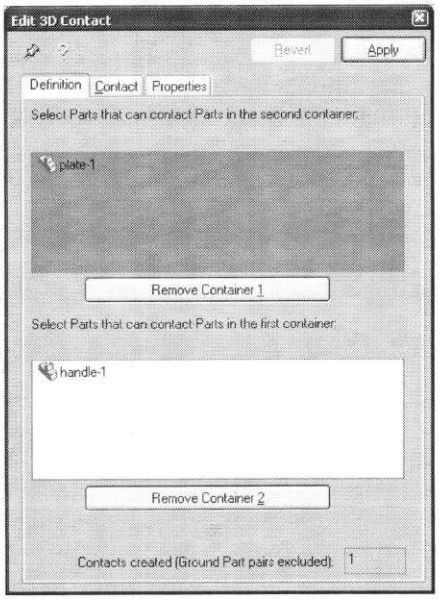


Figure 8-9 The *Edit 3D Contact* Dialog Box: *Definition* Tab

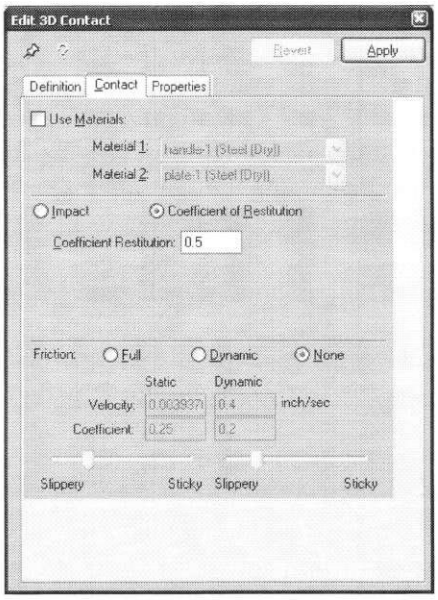


Figure 8-10 The *Edit 3D Contact* Dialog Box: *Contact* Tab

Also, we would like to make sure that the gravity is set up properly. From the browser, right click the *Motion Model* node and select *System Defaults.* In the *Options* dialog box (Figure 8-11), you should see the acceleration is *386.22* in/sec^2, and the *Direction* is set to *-1* for *Y*. Click *OK* to accept the gravity setting.

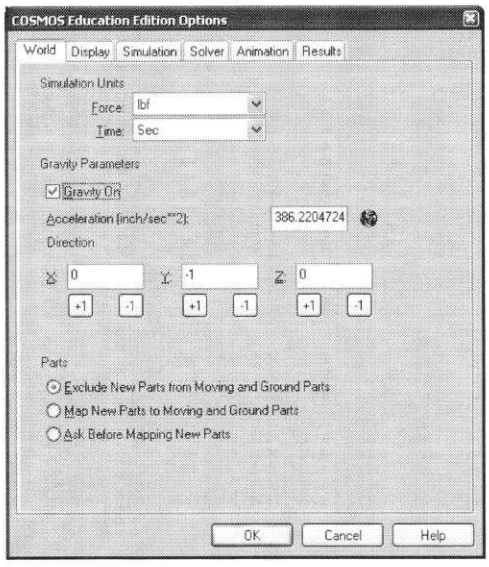


Figure 8-11 Turning Off Gravity

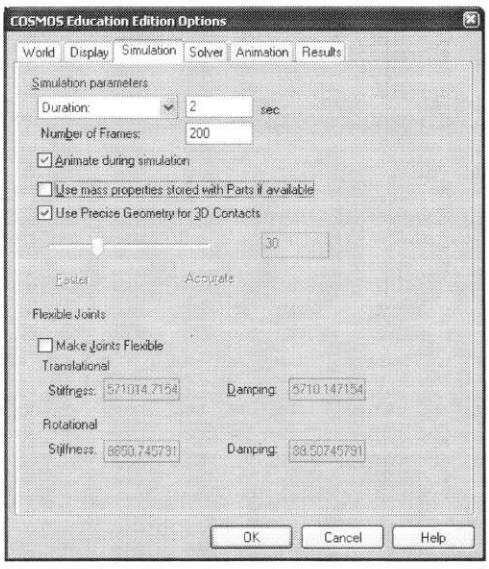


Figure 8-12 Choose Option to Detect Contact in Simulation

Click the *Motion Model* node from the browser, press the right mouse button and select *Simulation Parameters.* Enter *2* for simulation duration and the *200* for the number of frames, as shown in Figure 8-

12. Make sure the *Use Precise Geometry for 3D Contacts* is selected in order for *COSMOSMotion* to detect contact during simulation.

Click the *Motion Model* node again, press the right mouse button and select *Run Simulation.* You should see that the handle bar start moving forward and the kicking rod moving backward due to gravity. When the handle bar and the first lower bracket is in contact, the handle bar bounces back slightly due to the contact constraint we defined.

Next, we will graph the position and velocity of the kicking rod along the Z-direction.

From the browser, expand the *Parts* node and then the *Moving Parts* node. Right click *kickingjrod-7*, and choose *Plot > CM Position* > Z, and *Plot > CM Velocity > Z Component.*

Two graphs like those of Figures 8-13 and 8-14 should appear. Note that the vertical scales of the graphs have been adjusted for clarity. The position graph shows that the mass center of the kicking rod was located at Z = *21.6* in. initially. The mass center moves to the right to Z = *17.8* in., where the handle bar is in contact with the first lower bracket. The handle bar, therefore the kicking rod, bounces back, and the center mass of the kicking rod reaches to Z = *18.6* in. before it slides to the right again due to gravity. After about *1.5* seconds, the kicking rod rests and stays in contact with the bracket.

The velocity graph (Figure 8-14) shows that the bouncing velocity is half of the incoming velocity in the opposite direction. This is certainly due to the *0.5* restitution coefficient defined at the contact constraint.

From these two graphs, we conclude that the motion model has been defined correctly. Save your model. You may want to save the model under different name and use it for Task Two simulations.

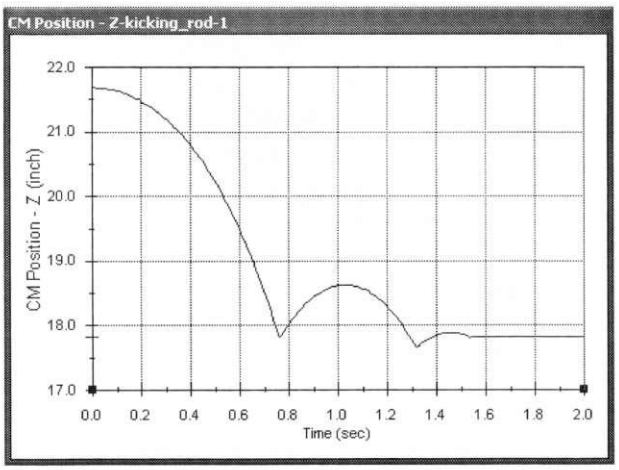


Figure 8-13 CM Position of the Kicking Rod

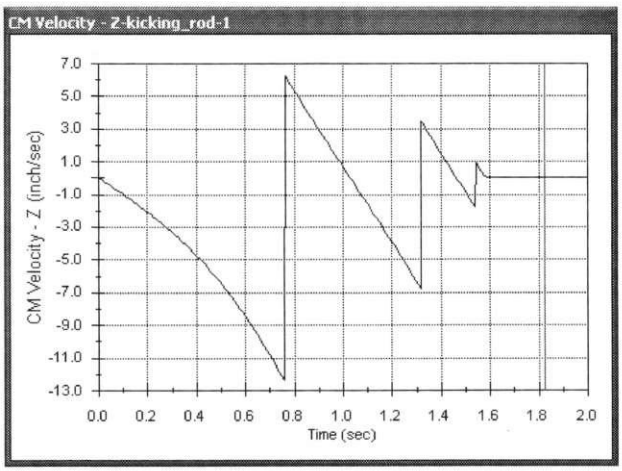


Figure 8-14 CM Velocity of the Kicking Rod

Task Two: Adding Spring

In Task Two, we will add a spring to the mechanism. We will adjust the spring constant and its free length until we reach an equilibrium configuration that we can work with. Note that the free length we specify will have to bring the handle bar backward about *5-10* degrees (that is, negative *5-10* degrees about the X-axis).

Delete the simulation result.

From the browser, right click the *Spring* node and choose *Add Translational Spring*, the *Insert Spring* dialog box will appear (Figure 8-15). Pick the center hole of the upper bracket and the hook of the handle bar, as shown in Figure 8-16. Enter the followings:

Stiffness: 30
Length: 5
Force: 0
Coil Diameter: 1
Number of coils: 10
Wire Diameter: 0.25

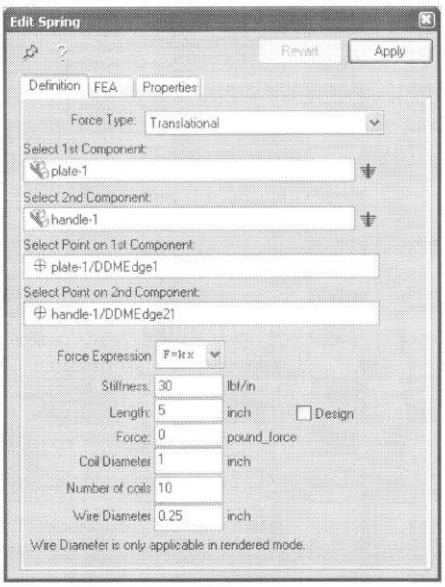


Figure 8-15 The *Edit Spring* Dialog Box

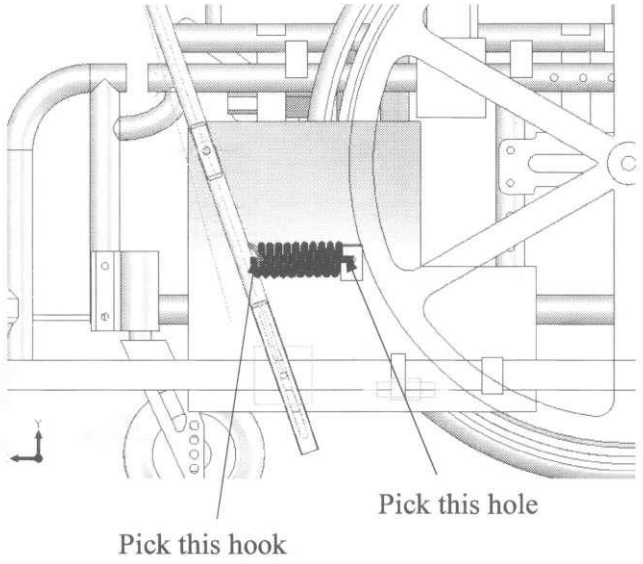


Figure 8-16 Pick Geometric Features to Define Spring

Note that the actual distance between the hole and the hook under the current configuration is about *4.34* in., which can be obtained by clicking the *Design* box to the right of the *Length* field. The number we enter, 5, is larger than the actual distance. Again, the purpose of entering a larger free spring length is to make the handle bar lean backward at equilibrium.

Click the *Apply* button to accept the spring. In the current configuration, the spring is compressed since the handle is leaning *10* degrees forward.

Run a simulation.

You should see that the handle bar start moving backward and the kicking rod moving forward due to the stretching of the spring.

When the simulation is completed, the position and velocity graphs of the kicking rod will appear, similar to Figures 8-17 and 8-18. Both the position and velocity graphs reveal a sinusoidal type curve. This is due to the fact that no friction has been applied to the joints. Also, the handle bar is not colliding with the bracket. The graphs show that the period of one vibration is just under *0.5* seconds.

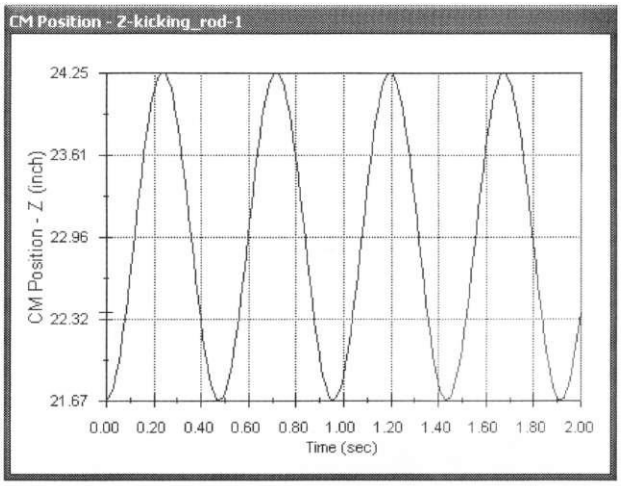


Figure 8-17 CM Position of the Kicking Rod

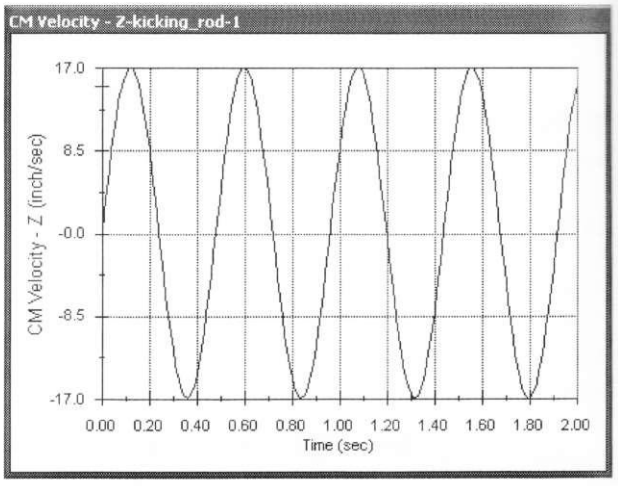


Figure 8-18 CM Velocity of the Kicking Rod

Now we will add a graph to show the rotation angle of the handle bar. Delete the simulation result. From the browser, right click the *handle-1* and choose *Plot* > *Bryant Angles* > *Angle7*; i.e., about the *X*-axis. Rerun the simulation, the angle graph should appear, similar to Figure 8-19. The graph shows that the handle bar is oscillating between *10* and *-9* degrees. The angle of the handle bar at equilibrium (assuming friction is added to dissipate energy) will be roughly the average of *10* and -9; i.e., less than *1* degree, which is far less than the desired angle (negative *5-10* degrees).

Delete the simulation result, change the free length to *5.5* in., and rerun the simulation. The angle graph shown in Figure 8-20 indicates that the handle bar oscillates between *10* and *-25* degrees, and the mean angle is about -7 degrees, which is desired.

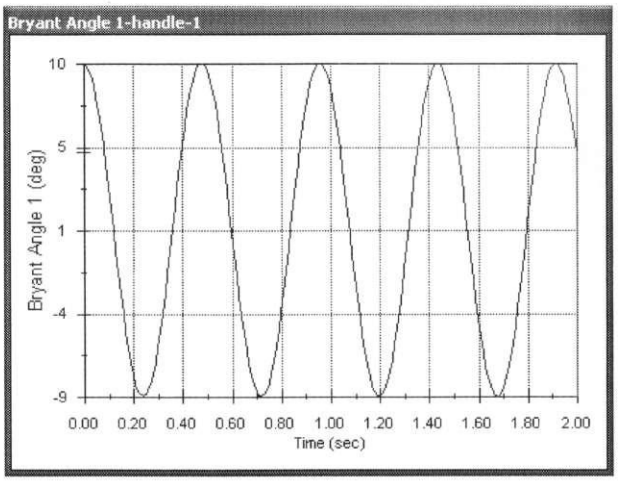


Figure 8-19 *X*-Rotation Angle of the Handle Bar: *U = 5″*

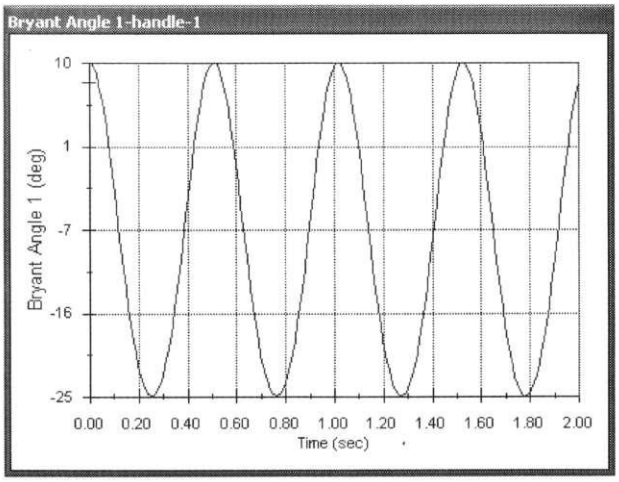


Figure 8-20 *X*-Rotation Angle of the Handle Bar: *U = 5.5″*

Note that a softer spring will increase the oscillation angle. If the spring constant is too small, say *2* lbf/in, the handle bar may reach a dead lock position with the middle pin of the kicking rod, similar to what was shown in Figure 8-4a, which is not desirable. On the other hand, if the spring is too stiff, the

force required to push the handle bar forward may be excessive. Currently, the spring constant is set to *30* lbf/in. This parameter will be revisited later in Task Three. Save you model before moving to Task Three.

Task Three: Determining the Operating Force

In Task Three, we will determine the operating force. The force must be small enough to allow children with limited physical strength to operate the mechanism. We will add a force at the handle, and use simulation results to determine the required operating force.

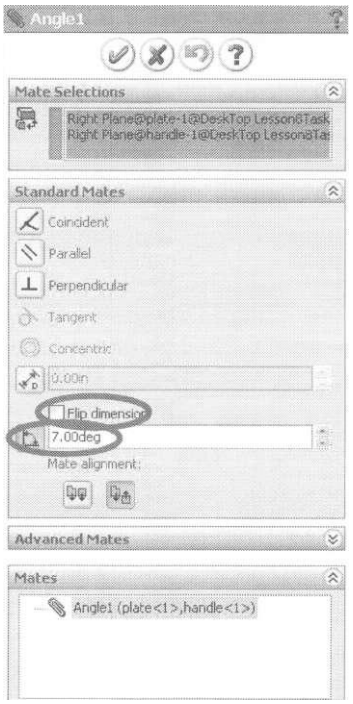


Figure 8-21

The force to be determined is supposed to push the handle forward about *10* degrees, as shown in Figure 8-6, and then pull it backward to push out the kicking rod. Then, we will turn on friction at both translational and revolute joints, and determine the operating force again. We will adjust the friction coefficient (assuming different physical conditions) in order to determine a range of the operating force. In the simulation, we will start with a configuration where the handle bar is set to -7 degrees; i.e., in its neutral position, as determined in Task Two.

Save the model under a different name, say *Task3.* Delete the simulation result, and go back to *SolidWorks* assembly mode by clicking the *Assembly* button on top of the browser.

First unsuppress the mate *AngleL* Then, right click the *Anglel* mate and choose *Edit Feature.* In the *Anglel* window (Figure 8-21), change the angle to 7, click the *Flip direction* button (to deselect it), and click the checkmark button on top to accept the definition. The handle bar should rotate to a 7-degree position backward in the graphics screen, as shown in Figure 8-22. This is the initial configuration for Task Three simulations.

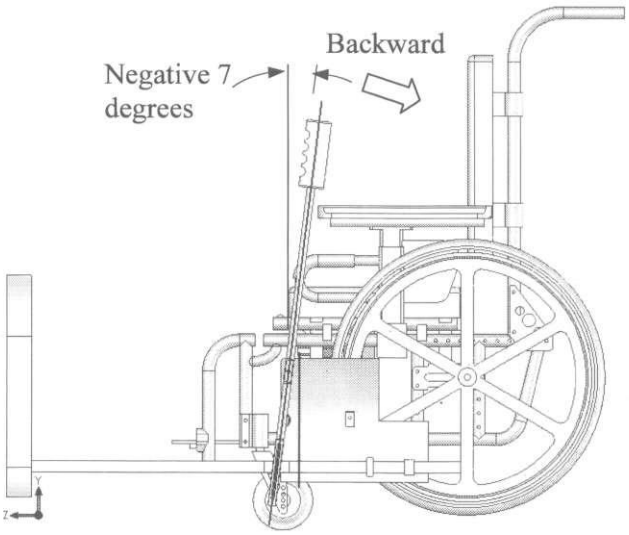


Figure 8-22 Set the Initial Angle to 7 Degrees Backward

Next we will create a force at the handle.

The force can be added from the browser by expanding the *Forces* branch, right clicking the *Action Only* node, and choosing *Add Action-Only Force.* In the *Insert Action-Only Force* dialog box, the *Select Component to which Force is Applied* field (see Figure 8-23) is active (highlighted in red) and ready for you to pick the entities.

Rotate the model and pick the face of the handle, as shown in Figure 8-24. The part *handle-1* is now listed in the *Select Component to which Force is Applied* field, and *handle- 1/DDMFace 10* is listed in both the *Select Location* and the *Select Direction* fields.

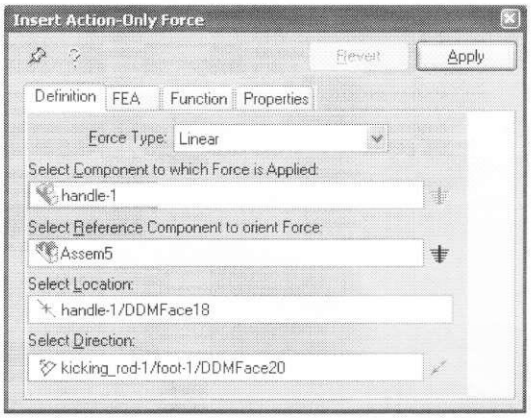


Figure 8-23 The *Insert Action-Only Force* Dialog Box

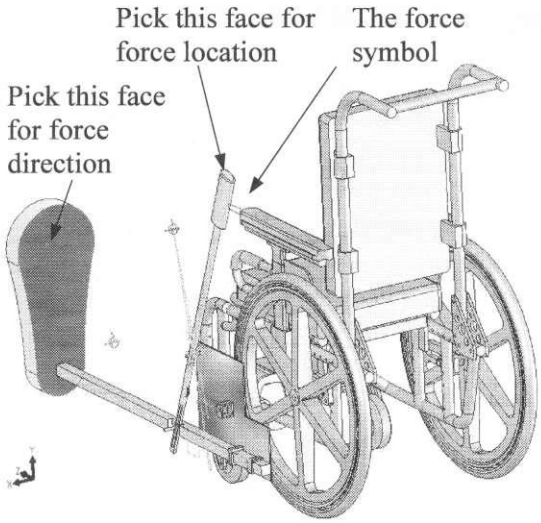


Figure 8-24 Defining the Operating Force

Now the Select *Reference Component to orient Force* field is active for selection. We will click the ground button to the right 01 the held. Alter that, you should see *Assem5* (representing the ground) appear in the text field. A force symbol will appear in the graphics screen pointing downward, which is not what we want. We will have to change the force direction.

Now the Select *Reference Component to orient Force* field is active for selection. We will click the ground button • to the right of the field. After that, you should see *Assem5* (representing the ground) appear in the text field. A force symbol will appear in the graphics screen pointing downward, which is not what we want. We will have to change the force direction.

Select all the text in the *Select Direction* field (press the let mouse button and drag to select all text), and press the *Delete key* to delete the current selection. Pick the end face of the foot, as shown in Figure 8-24, to *orient* the force along the negative Z-direction. The arrow of the force symbol should now point to the negative Z-direction, which is normal to the face we picked.

Click the *Function* tab (see Figure 8-25), choose *Spline* for function, and enter the following:

sec	pound_force
0	0
0.1	–3
0.2	–5
0.3	–5
0.4	–3
0.5	0
0.6	3
0.7	5
0.8	3
0.9	1
1.0	0
2.0	0

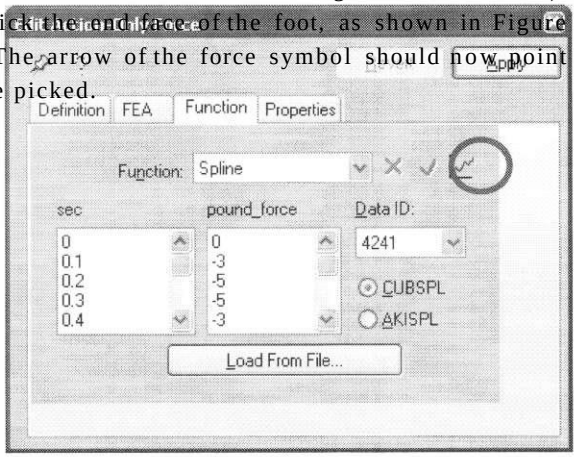


Figure 8-25 Entering Force Data

The maximum force is 5 lb_f. Note that the negative sign for the first few force data is to reverse the force direction in order to push the handle bar forward.

Click the graph button (right most, as circled in Figure 8-25), the function graph will appear like the one in Figure 8-26. Note that internally *COSMOSMotion* will create a smooth spline function using the data entered.

Close the graph and click *Apply* button to accept the force definition.

Run a simulation. The result graphs will appear at the end of the 2-second simulation.

The angle graph (Figure 8-27) shows that the handle bar starts at an orientation angle of -7 degrees as expected, and swing forward to a *13*-degree angle, which is desirable, due to the forward force. The handle then moves backward to about *26* degrees, and then oscillates.

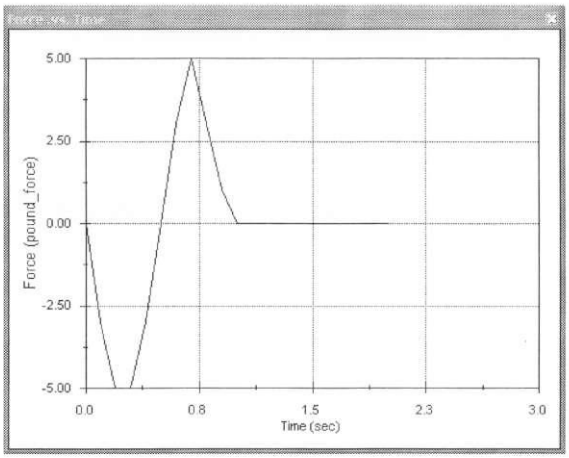


Figure 8-26 Force Polyline Graph

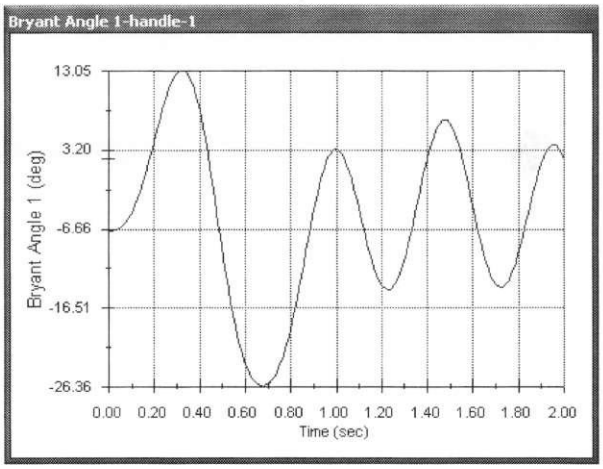


Figure 8-27 *X*-Rotation Angle of the Handle Bar: No Friction

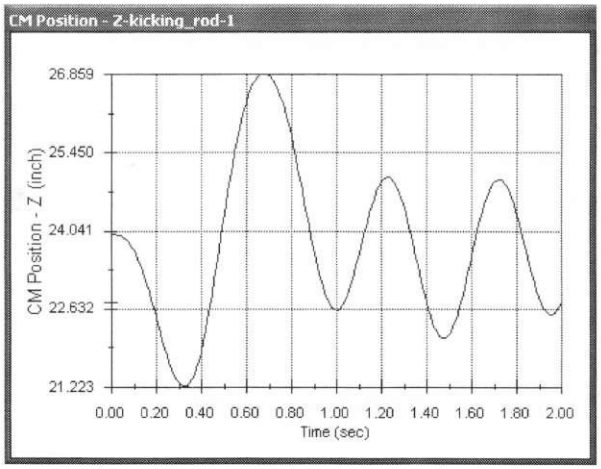


Figure 8-28 *Z*-Position the Kicking Rod: No Friction

The position graph in Figure 8-28 shows that the mass center of the kicking rod starts at about *24* in. It then travels to about *21.2* (backward) and then to about *26.8* in. forward due to the pulling force and stretch of the spring. The overall distance that the kicking rod travels is about *5.6* in., which seems to be sufficient to produce enough momentum to kick the ball.

Figure 8-29 shows that the velocity of the kicking rod reaches about *27* in/sec when the rod is pushed near the foremost position. This velocity will produce a momentum of about *310* lbf-sec at *0.5* seconds, as shown in Figure 8-30. To create a momentum graph you may simply right click *kicking_rod-l* from the browser and choose *Plot* > *Translational Momentum* > *Z Component.* Note that the velocity and momentum are proportional, with the mass of the kicking rod as the scaling factor.

The maximum force required to operate the device is determined to be *5* lb_f (maximum force value entered), assuming no friction at any joints. Even though the force data we entered show a polyline in Figure 8-26, the actual force employed for simulation in *COSMOSMotion* is a spline curve generated

using the data we entered. What does the force spline look like? We can graph a reaction force for the force *ForceAO* by right clicking the *ForceAO* node, and choosing *Plot > Reaction Force > Z Component*, as shown in Figure 8-31. The force will appear like that of Figure 8-32. Note that this spline curve is smooth and pass through all data points entered.

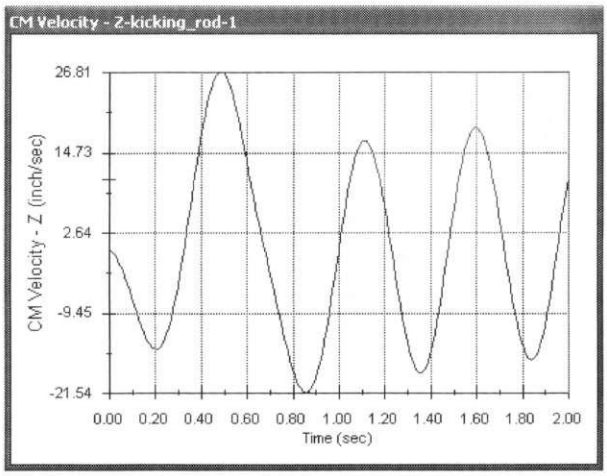


Figure 8-29 *Z*-Velocity the Kicking Rod: No Friction

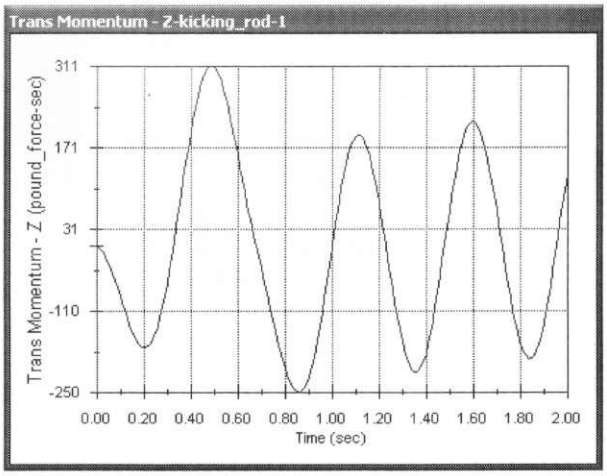


Figure 8-30 *Z*-Momentum the Kicking Rod: No Friction

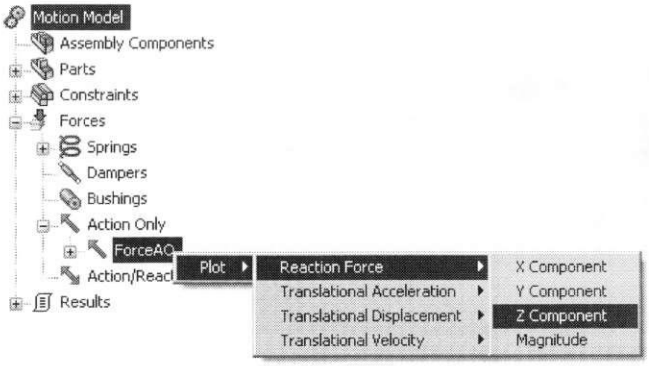


Figure 8-31 Graphing the Reaction Force

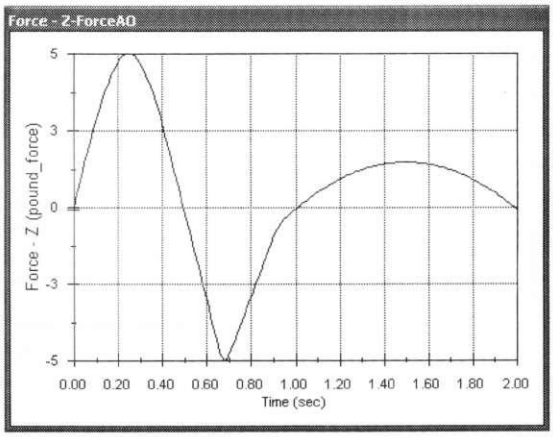


Figure 8-32 Operating Force in Spline Function

Save you model. We will turn on friction and continue determining the required operating force.

We will turn on friction at both the revolute and the translational joints. Friction for the *CamMateTangent* joint is currently unavailable in *COSMOSMotion.*

Delete the simulation result from the browser. Expand the *Constraints* and then the *Joints* branch. Right click *Revolute* and choose *Properties.* In the *Edit Mate-Defined Joint* dialog box (Figure 8-33), choose the *Friction* tab, click the *Use Friction,* and choose *Aluminum Greasy* for both *Material 1* and *Material 2.* The *Coefficient (mu)* will show *0.03.* Enter *Joint dimensions, Radius: 0.26* and *Length: 0.31.* Click *Apply* button to accept the definition. Note that the dimensions entered are the diameter of the pivot pin and the thickness of the handle bar where the joint is located.

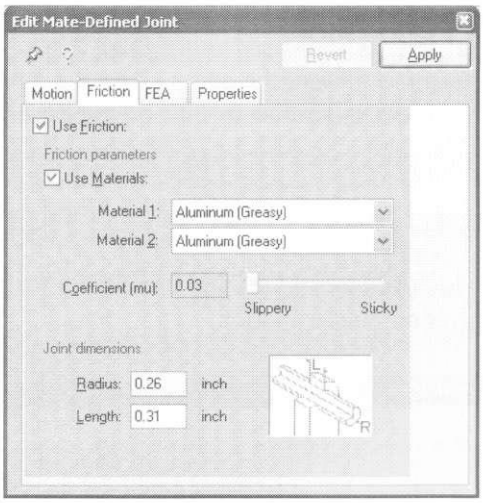


Figure 8-33 Defining Friction for the Revolute Joint

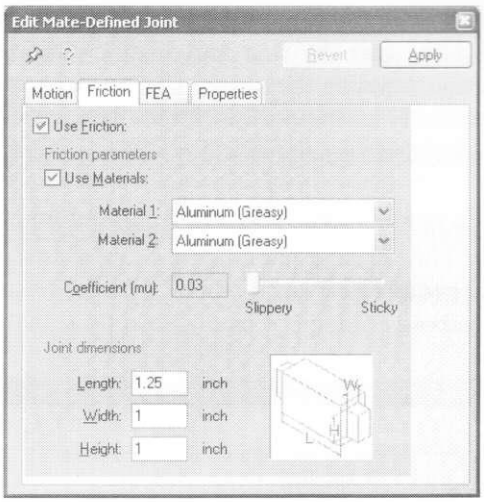


Figure 8-34 Defining Friction for the Translational Joint

Similarly, turn on the friction for the translational joint. Enter 7.25, 7, and 7, for *Length, Width,* and *Height,* respectively, as shown in Figure 8-34. Note that the length dimension entered is the sum of both the lower brackets; i.e., 7 and *0.25* in. for the first and second lower brackets, respectively. The width and height of the inner square of the brackets is 7 and 7 in., respectively.

Run a simulation and check the angle of the handle bar.

Even though the friction coefficient is small *(ju = 0.03)* for both joints, the handle bar hardly moves. Therefore, the force magnitude must be increased. We will follow in general the overall force pattern, and increase the force magnitude to push the handle bar forward.

Delete the simulation result. Expand the *Forces* and then the *Action Only* nodes from the browser. Right click the *ForceAO* node and choose *Properties.* In the *Edit Action-Only Force* dialog box, choose the *Function* tab (see Figure 8-25), and enter the followings:

sec	pound_force
0	*0*
0.1	*–8*
0.2	*–12*
0.3	*–12*
0.4	*–8*
0.5	*0*
0.6	*3*
0.7	*5*
0.8	*3*
0.9	*1*
1.0	*0*
2.0	*0*

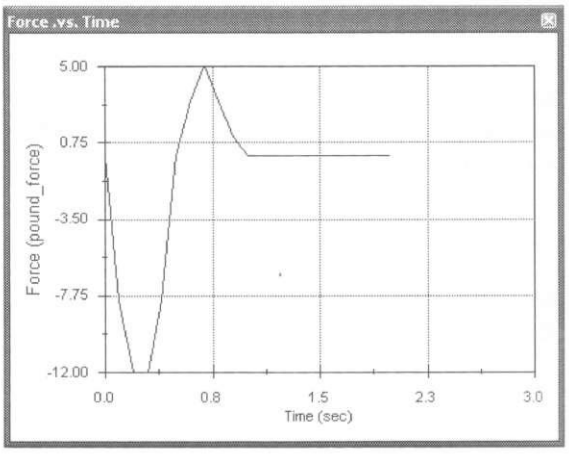


Figure 8-35 Operating Force: Small Friction

Click the graph button, the function will appear like the one in Figure 8-35. The maximum force is now *12* lbf. Note that the first half of the force (negative part) is increased more than twice, and the positive portion remains the same. The positive part of the force is kept the same since the spring will contribute partially to the pulling force. The overall force pattern is similar to that of Figure 8-26. The force data entered are results of a few trials-and-errors.

Rerun a simulation. The angle and momentum graphs appear as in Figures 8-36 and 8-37, respectively. The angle graph (Figure 8-36) shows that the handle bar starts at an orientation angle of *–7* degrees, and swings forward to about a *12*-degree angle, which is acceptable. The handle bar then moves backward to about *10* degrees, and then rests at that configuration due to friction.

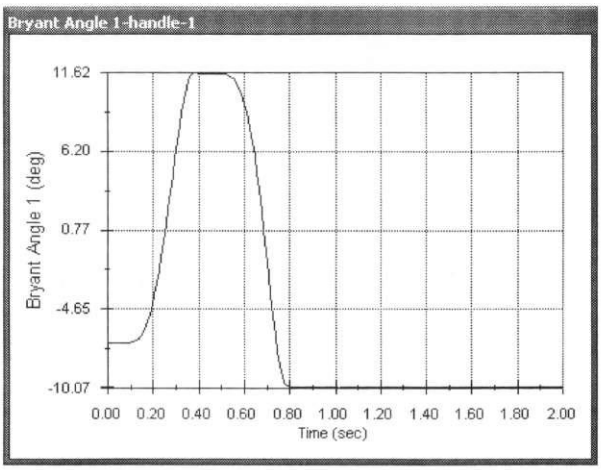


Figure 8-36 *X*-Rotation Angle of the Handle Bar: Small Friction

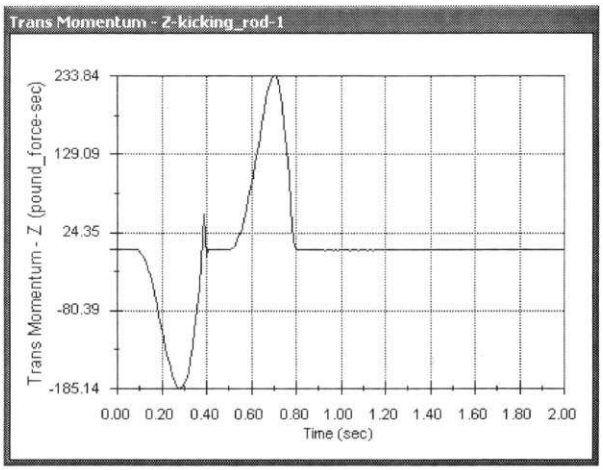


Figure 8-37 *Z*-Momentum of the Kicking Rod: Small Friction

The momentum graph (Figure 8-37) shows that when the rod is pushed near the foremost position the momentum of the kicking rod is about *230* lb,sec, which is less than the previous non-friction case. After reviewing the graphs, the 72-lb, force seems to be acceptable. However, this small operating force, *12* lbf, is due to a very small friction force; i.e., friction coefficient ***jU*** = *0.03.* This result also indicates that the spring constant *30* lbf/in seems to be adequate. Next, we will increase the friction coefficient by changing the material from *Aluminum Greasy* to *Aluminum Dry.*

Delete the simulation result. Expand the *Constraints* and then the *Joints* branch. Right click *Revolute* and choose *Properties.* In the *Edit Mate-Defined Joint* dialog box, choose the *Friction tab,* and choose *Aluminum Dry* for both *Material 1* and *Material 2.* The *Coefficient (mu)* will show *0.20.* Repeat the same for the translational joint. Run a simulation. The handle bar is hardly moved. That is, the force will have to be increased again.

Delete the simulation results. Right click the *ForceAO* node to enter the force data. Note that after several attempts, a force that is large enough to move the handle bar, therefore the kicking rod, is about *65* lbf. More specifically, the force data entered are:

sec	pound_force
0	0
0.1	–30
0.2	–65
0.3	–65
0.4	–30
0.5	0
0.6	3
0.7	5
0.8	3
0.9	1
1.0	0
2.0	0

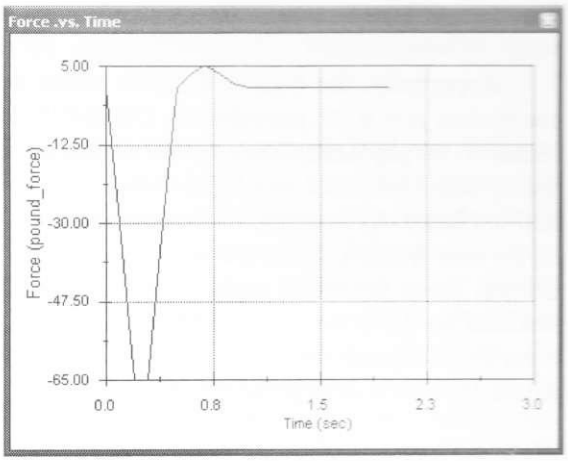


Figure 8-38 Operating Force: Large Friction

Click the graph button, the function will appear like the one in Figure 8-38. Even though the maximum pushing force is now *65* lb_f. the pulling force (positive portion) remains the same, Are: running a simulation, the angle and momentum graphs appear as in Figures 8-39 and 8-40, respectively Both seem to be reasonable. However, the large operating force, *65* lb_f, raises a flag. The mechanism requires too large a force for a child to operate.

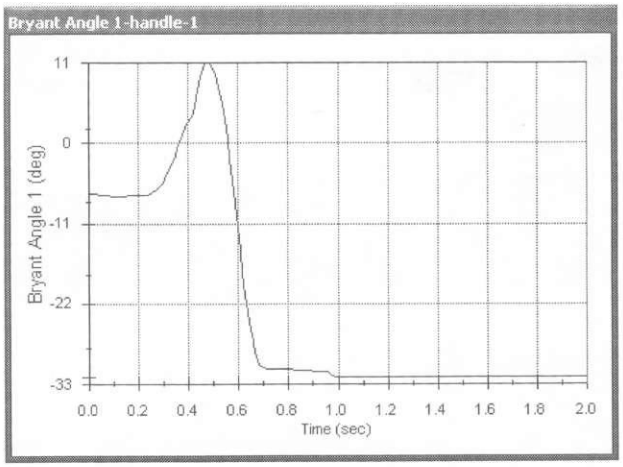


Figure 8-39 *X*-Rotation Angle of the Handle: Large Friction

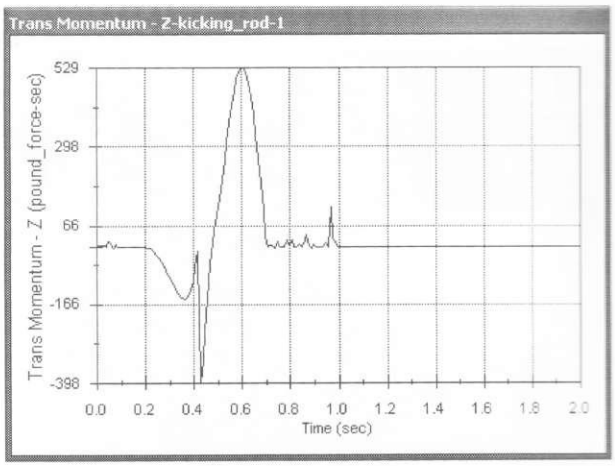


Figure 8-40 *Z*-Momentum of the Kicking Rod: Large Friction

Note that the simulation engine, *ADAMS/Solver,* is very sensitive to the force data entered. You may encounter problems while carrying out some of the simulations in Task Three. When this happens, simply change the maximum force data, e.g., *65,* to a slightly different value, e.g., *63,* until a simulation can be completed.

Save your model.

8.4 Result Discussion

Apparently, the biggest concern raised in the simulations is the large operating force. The friction coefficient *ju = 0.20,* provided by *COSMOSMotion* for *Aluminum-Aluminum* contact without lubrication, seems to be physically reasonable. Although this friction coefficient represents an extreme case since in reality some lubricant would be added to the joints to reduce friction resistance. Nevertheless, the force is still too large. As revealed in simulations; i.e., Task Three, the maximum operating force increases from 5 lbf for non-friction, *12* lbf for small friction *(ju = 0.03)* to *65* lbf for large friction *(ju = 0.20).* Reducing the friction force at joints, especially the translational joint between the kicking rod and the two lower brackets on the plate, is critical for a successful device. You can easily confirm that the translational joint contributes significantly to the friction encountered in the mechanism by conducting separate simulations where friction is only present in one of the two joints.

The flag raised by the simulation has been observed in the physical device, as shown in Figure 8-41, built by students following the design created in *SolidWorks* and *COSMOSMotion.* The physical device confirms that the contact between the kicking rod and the two brackets produces a large friction force, resulting in a large operating force to operate the device. For children with limited physical strength, such a device is unattractive.

In order to reduce the friction, four bearings are added to the device, as shown in Figure 8-42. Two are added to the top surface of the kicking rod, and two are underneath the kicking rod. With the bearings, the friction is significantly reduced. Therefore, a smaller force is required to operate the device. The actual operating force is less than *20* lbf.

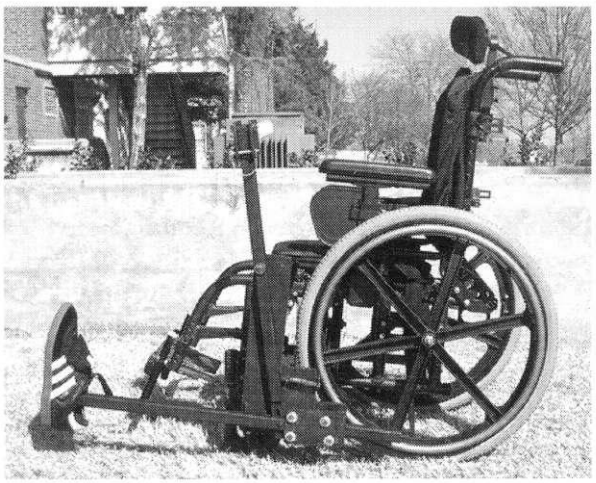


Figure 8-41 Device Assembled to the Wheelchair

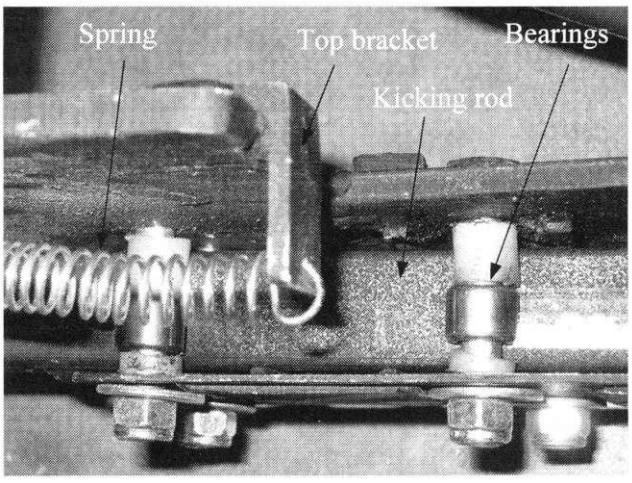


Figure 8-42 Bearings Added (Top View)

8.5 Comments on *COSMOSMotion* Capabilities and Limitations

Using *COSMOSMotion* does answer critical questions and help the design process, as demonstrated in this example. However, from this example, a number of limitations in *COSMOSMotion* have also been encountered. Knowing these limitations will help you use *COSMOSMotion* more effectively.

First, the simulation engine, *ADAMS/Solver,* which solves the equations of motion for the mechanism, is not stable. Sometimes when you rerun the same simulation, you could see slightly different results. This problem is more vivid when we ran the same simulation using different computers.

Moreover, the simulation engine produced results, such as the velocity or momentum (see Figures 8-37 and 8-40), with spikes, which is not quite realistic. Some of the spikes disappear or become smaller when we rerun the same simulation.

Second, friction is not supported for a *CamMateTangent* joint. Therefore, an attempt was made to add a 3D contact joint between the middle pin and the slot of the handle bar. A number of restitution coefficients were used, including *0.5, 0.75,* etc. Yet, adding a 3D contact joint significantly slow down the motion of the handle bar and the kicking rod since lots of contact are encountered between the outer surface of the middle pin and the inner surface of the slot when the mechanism is in motion. More contact causes more energy dissipation. However, the slow-down is way too much to grant a physically reasonable simulation. In addition, more spikes appear in graphs, such as in velocity. However, the biggest issue with adding the 3D contact is that the simulation engine becomes extremely sensitive to the initial orientation angle of the handle bar and the magnitude of the operating force. For some simulations, the solution engine encountered problems, for example, force imbalance at certain time steps during the simulation, and terminated the simulation prematurely.

Overall *COSMOSMotion* is an excellent tool with lots of nice features and capabilities for support of mechanism design and analysis. However, as mentioned in this book numerous times, no software is foolproof. Before creating a simulation model, you always want to formulate your design questions and set up your simulation model and scenarios gearing toward answering these specific questions. You will have to exam the simulation results very carefully and challenge yourself about the validity of the results since if you don't somebody else will do, usually in a less friendly way.

Notes:

APPENDIX A: DEFINING JOINTS

Degrees of Freedom

Understanding degrees of freedom is critical in creating successful motion model. The free degrees of freedom of the mechanism represent the number of independent parameters required to specify the position, velocity, and acceleration of each rigid body in the system for any given time. A completely unconstrained body in space has six degrees of freedom, three translational and three rotational. If you add a joint; e.g., a revolute joint to the body, you restrict its movement to rotation about an axis, and the free degrees of freedom of the body are reduced from six to one.

For a given motion model, you can determine its number of degrees of freedom using the Gruebler's count. The mechanism's Gruebler count is calculated using the mechanism's total number of bodies. As mentioned above, each movable body introduces six degrees of freedom. Joints added to the mechanism constrain the system, or remove dofs. Motion inputs, i.e., motion drivers, remove additional dofs.

COSMOSMotion uses the following equation to calculate the Gruebler's count:

$$D = 6M - N - O \tag{A.1}$$

where D is the Gruebler count representing the total free degrees of freedom of the mechanism, M is the number of bodies excluding the ground body, TV is the number of dofs restricted by all joints, and O is the number of the motion inputs in the system.

For kinematic analysis, the Gruebler's count must be equal to or less than *0*. The *ADAMS/Solver* recognizes and deactivates redundant constraints during motion simulation. For a kinematic analysis, if you create a model with a Gruebler's count greater than *0* and try to simulate it, the simulation will not run and an error message will appear.

If the Gruebler's count is less than zero, the solver will automatically remove redundancies, if possible. For example, you may apply this formula to a door model that is supported by two hinges modeled as revolute joints. Since a revolute joint removes five dofs, the Gruebler's count becomes:

$$D = (6 \times 1) - (2 \times 5) = -4.$$

The calculated degrees of freedom result is -4, which include five redundant dofs.

Redundancy

Redundancies are excessive dofs. When a joint constrains the model in exactly the same way as another joint (like the door example), the model contains excessive dofs, also known as redundancies. A joint becomes excessive when it does not introduce any further restriction on a body's motion.

It is important that you eliminate redundancies from your motion model while carrying out dynamic analyses. If you do not remove redundancies, you may not get accurate values when you check joint reactions or load reactions.

For example, if you model a door using two revolute joints for the hinges, the second revolute joint does not contribute to constraining the door's motion. *COSMOSMotion* detects the redundancies and ignores one of the resolute joints in its analysis. The outcome may be incorrect in reaction results, yet the

motion is correct. For complete and accurate reaction forces, it is critical that you eliminate redundancies from your mechanism.

For a kinematic simulation where you are interested in displacement, velocity, and acceleration, redundancies in your model do not alter the performance of the mechanism.

You can eliminate or reduce the redundancies in your model by carefully choosing joints. These joints must be able to restrict the same dofs, but not duplicate each other, introducing redundancies. After you decide which joints you want to use, you can use the Gruebler's count to calculate the dofs and check redundancies.

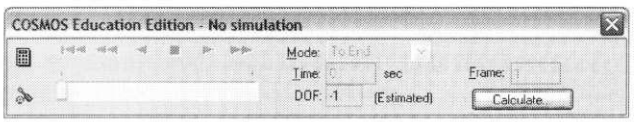


Figure A-1 The *Simulation Control* Dialog Box

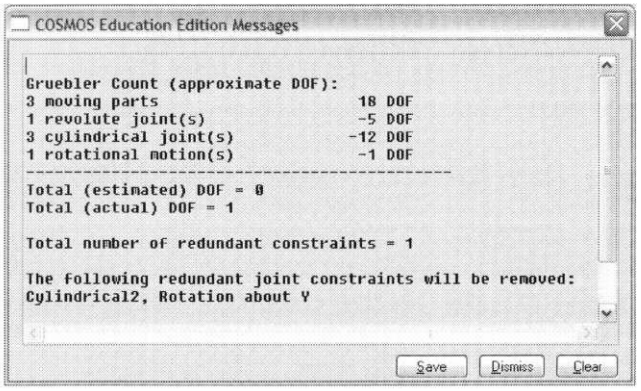


Figure A-2 The *Message* Window

You may also ask *COSMOSMotion* to calculate the Gruebler's count for you. You can simply click the *Show simulation control* button on top of the graphics screen to bring up the dialog box, as shown in Figure A-1. Note that the *DOF* field in the dialog box will show the Gruebler's count if a simulation has been completed. If not, You may click the *Calculate* button to ask *COSMOSMotion* to calculate the actual dof. In the message window appearing next (Figure A-2), *COSMOSMotion* identifies redundant dofs and recalculates the dof for the simulation model.

Joint Types in *COSMOSMotion*

Before you select a joint to add to your model, you should know what movement you want to restrain for the body and what movement you want to allow. The following table describes the commonly employed joint types in *COSMOSMotion* and the degrees of freedom they remove.

Joint Type	DOF Removed			Remarks
	Translation	Rotation	Total	
Revolute	3	2	5	Rotates about an axis
Translational	2	3	5	Translates along an axis
Cylindrical	2	2	4	Translates along and rotates about an axis
Spherical	3	0	3	Rotates in any direction
Universal	3	1	4	Rotates about two axes
Screw	0.5	0.5	1	Coupled rotation and translation along one axis
Planar	2	1	3	Bodies connected by a planar joint move in a plane with respect to each other. Rotation is about an axis perpendicular to the plane.
Fixed	3	3	6	Glues two parts together Parts constrained by a rigid connection constitute a single body.

The following provides more details about the joints listed in the table above.

Revolute Joint

A revolute joint, as depicted in Figure A-3, allows the rotation of one rigid body with respect to another rigid body about a common axis. The origin of the revolute joint can be located anywhere along the axis about which the bodies can rotate with respect to each other. The joint origin is assigned by *COSMOSMotion* when you enter *COSMOSMotion* from *SolidWorks.*

Orientation of the revolute joint defines the direction of the axis about which the bodies can rotate with respect to each other. The rotational axis of the revolute joint is parallel to the orientation vector and passes through the origin.

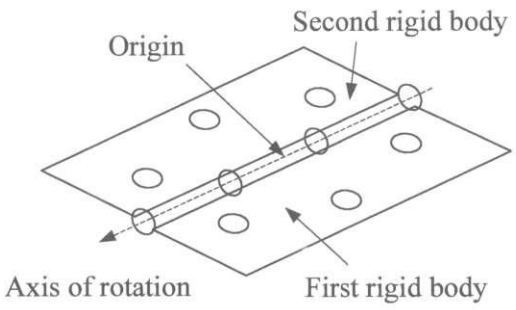


Figure A-3 Revolute Joint Symbol

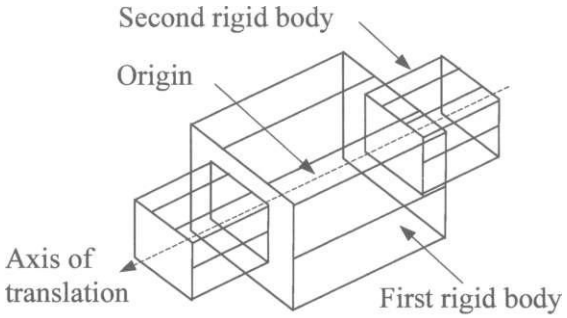


Figure A-4 Translational Joint Symbol

Translational Joint

A translational joint allows one rigid body to translate along a vector with respect to a second rigid body, as illustrated in Figure A-4. The rigid bodies may only translate, not rotate, with respect to each other.

The location of the origin of a translational joint with respect to its rigid bodies does not affect the motion of the joint but does affect the reaction loads on the joint. The location of the joint origin determines where the joint symbol is located.

The orientation of the translational joint determines the direction of the axis along which the bodies can slide with respect to each other (axis of translation). The direction of the motion of the translational joint is parallel to the orientation vector and passes through the origin.

Cylindrical Joint

A cylindrical joint allows both relative rotation and relative translation of one body with respect to another body, as shown in Figure A-5. The origin of the cylindrical joint can be located anywhere along the axis about which the bodies rotate or slide with respect to each other.

Orientation of the cylindrical joint defines the direction of the axis about which the bodies rotate or slide along with respect to each other. The rotational/translational axis of the cylindrical joint is parallel to the orientation vector and passes through the origin.

Spherical Joint

A spherical joint allows free rotation about a common point of one body with respect to another body, as depicted in Figure A-6. The origin location of the spherical joint determines the point about which the bodies pivot freely with respect to each other.

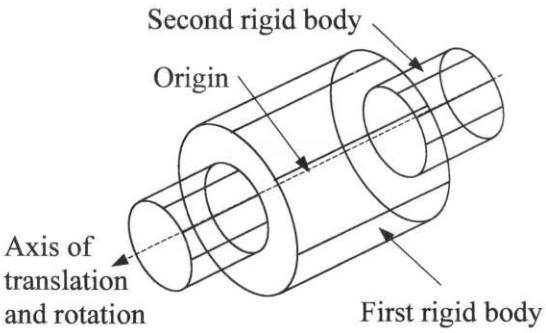


Figure A-5 Cylindrical Joint Symbol

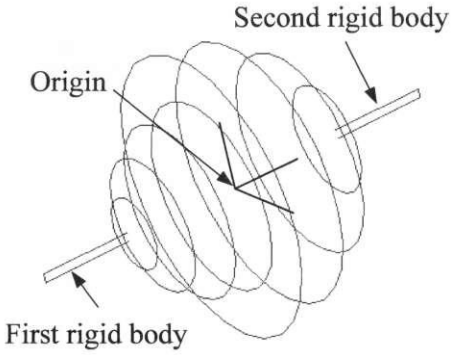


Figure A-6 Spherical Joint Symbol

Universal Joint

A universal joint allows the rotation of one body to be transferred to the rotation of another body, as shown in Figure A-7. This joint is particularly useful to transfer rotational motion around corners, or to transfer rotational motion between two connected shafts that are permitted to bend at the connection point (such as the drive shaft on an automobile).

The origin location of the universal joint represents the connection point of the two bodies. The two shaft axes identify the center lines of the two bodies connected by the universal joint. Note that *COSMOSMotion* uses rotational axes parallel to the rotational axes you identify but passing through the origin of the universal joint.

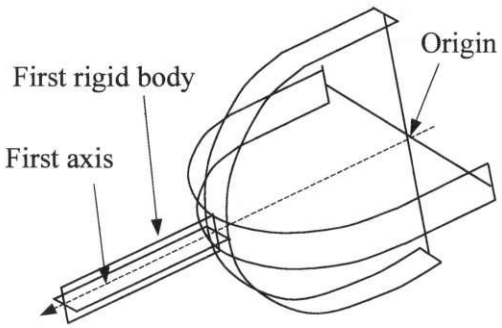


Figure A-7 Universal Joint Symbol

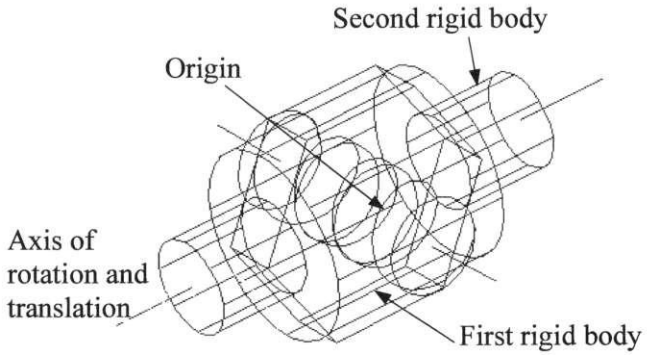


Figure A-8 Screw Joint Symbol

Screw Joint

A screw joint removes one degree of freedom. It constrains one body to rotate as it translates with respect to another body, as shown in Figure A-8.

When defining a screw joint, you can define the pitch. The pitch is the amount of translational displacement of the two bodies for each full rotation of the first body. The displacement of the first body

relative to the second body is a function of the body's rotation about the axis of rotation. For every full rotation, the displacement of the first body along the translation axis with respect to the second body is equal to the value of the pitch.

Very often, the screw joint is used with a cylindrical joint. The cylindrical joint removes two translational and two rotational degrees of freedom. The screw joint removes one more degree of freedom by constraining the translational motion to be proportional to the rotational motion.

Planar Joint

A planar joint allows a plane on one body to slide and rotate in the plane of another body, as shown in Figure A-9.

The orientation vector of the planar joint is perpendicular to the joint's plane of motion. The rotational axis of the planar joint, which is normal to the joint's plane of motion, is parallel to the orientation vector.

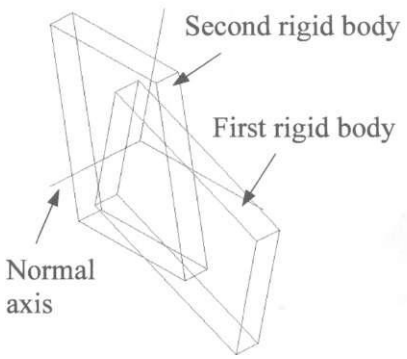


Figure A-9 Planar Joint Symbol

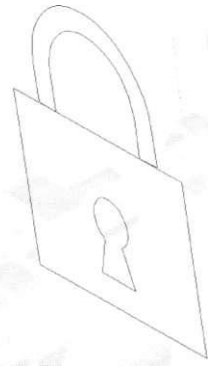


Figure A-10 Fixed Joint Symbol

Fixed Joint

A fixed joint locks two bodies together so they cannot move with respect to each other. The fixed joint symbol is shown in Figure A-10.

Mapped *SolidWorks* Mates

After you bring an assembly from *SolidWorks* to *COSMOSMotion* and assign components to ground part and moving parts, *COSMOSMotion* automatically maps assembly mates to joints. In most cases, these joints work well for the motion simulation. A selected set of the mapped *SolidWorks* mates, which is commonly employed in assembly, is given in the next table for your reference.

Mate Type	Feature 1	Feature 2	*COSMOSMotion* Joint
Coincident	Cone	Circular/Arc Edge	Cylindrical
Coincident	Cone	Cone	Revolute
Coincident	Cylinder	Circular/Arc Edge	Cylindrical
Coincident	Line	Line	Cylindrical
Coincident	Plane	Plane	Planar
Coincident	Point	Point	Spherical
Coincident	Point	Sphere	Spherical
Coincident	Circular/Arc Edge	Circular/Arc Edge	Revolute
Concentric	Cone	Circular/Arc Edge	Cylindrical
Concentric	Cone	Cone	Cylindrical
Concentric	Cone	Cylinder	Cylindrical
Concentric	Cone	Line	Cylindrical
Concentric	Cylinder	Cylinder	Cylindrical
Concentric	Cylinder	Line	Cylindrical
Concentric	Cylinder	Circular/Arc Edge	Cylindrical
Concentric	Line	Circular/Arc Edge	Cylindrical
Concentric	Point	Sphere	Spherical
Concentric	Sphere	Sphere	Spherical
Concentric	Circular/Arc Edge	Circular/Arc Edge	Cylindrical
Distance	Cone	Cone	Revolute
Distance	Plane	Plane	Planar

Note that a complete list of mapped mates can be found in *COSMOSMotion* help. *COSMOSMotion* help system can be viewed by choosing from the pull-down menu, *Help > COSMOS > Help on COSMOS Education Edition*.

APPENDIX B: THE UNITS SYSTEM

The mass unit lb_m is not quite common to many engineers. The basic physical quantities involved in determining a units system are length, time, mass, and force. These four basic quantities are related through Newton's second law,

$$F = ma \tag{B.1}$$

where F, m, and a are force, mass, and acceleration (length per second square), respectively.

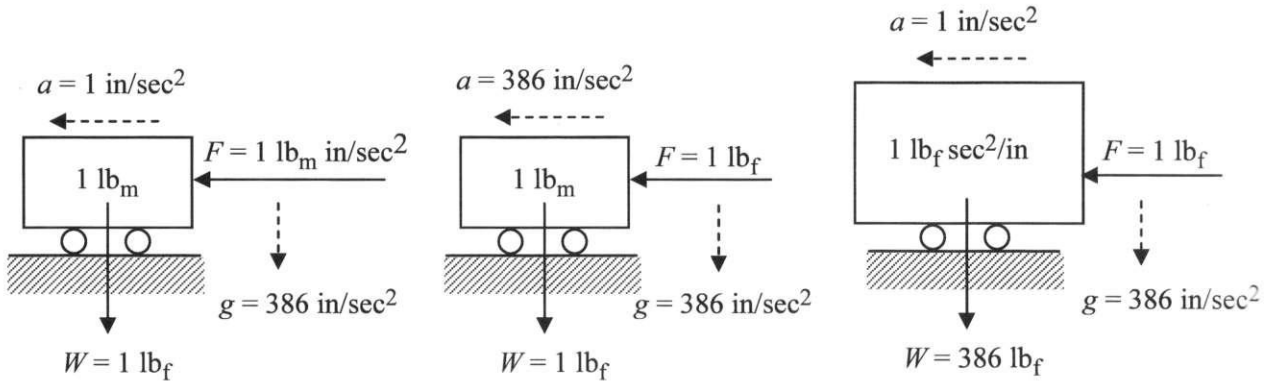
When the unit lb_m for mass is employed, the force unit will be determined by length (in.), mass (lb_m), and second (sec) through Eq. B.1; i.e.,

$$1\ lb_m\ in/sec^2\ (force) = 1\ lb_m\ (mass) \times 1\ in/sec^2\ (acceleration) \tag{B.2}$$

where the force unit, lb_m in/sec^2, is a derived unit.

From Eq. B.2, a *1* lb_m in/sec^2 force will generate a *1* in/sec^2 acceleration when applied to a *1* lb_m mass block, as shown in Figure B-1a. The same block will weigh *1* lb_f on earth (see Figure B-1b), where the gravitational acceleration is assumed *386* in/sec^2; i.e.

$$1\ lb_f\ (force) = 1\ lb_m\ (mass) \times 386\ in/sec^2\ (acceleration) \tag{B.3}$$



$F = 1\ lb_m\ in/s^2 = 1\ lb_m \times 1\ in/s^2$ $W = 1\ lb_f = 1\ lb_m \times 386\ in/s^2$	$F = 1\ lb_f = 1\ lb_m \times 386\ in/s^2$ $W = 1\ lb_f = 1\ lb_m \times 386\ in/s^2$	$F = 1\ lb_f = 1\ lb_f\ s^2/in \times 1\ in/s^2$ $W = 1\ lb_f\ s^2/in \times 386\ in/s^2 = 386\ lb_f$
(a) A 1 lb_m in/sec^2 Force Applied to a 1 lb_m Mass Block	(b) A 1 lb_f Force Applied to a 1 lb_m Mass Block	(c) A 1 lb_f Force Applied to a 1 lb_f sec^2/in Mass Block

Figure B-1 Forces Applied on Blocks of Different Masses

Therefore, from Eqs. B.2 and B.3, we have 7 lb_f = *386* lb_m in/sec^2, and the force in lb_m in/sec^2 unit is 386 times smaller than that of lb_f that we are more used to. When you apply a 7 lb_f force to the same mass block, it will accelerate *386* in/sec^2, as shown in Figure B-lb.

On the other hand, we have the mass unit, 7 lb_m *1/386* lb_f sec^2/in. It means that a *1* lb_m mass block is *386* times smaller than that of a 7 lb_f sec^2/in block. Therefore, a *1* lb_f sec^2/in block will weigh *386* lb_f on earth. When applying a *1* lbf force to the mass block, it will accelerate at a 7 in/sec^2 rate, as illustrated in Figure B-lc.

The mass unit slug that we are more familiar with is defined as

$$1\ lb_f\ (force) = 1\ slug\ (mass) \times 1\ ft/sec^2\ (acceleration) = 386\ lb_m\ in/sec^2 = 32.2\ lb_m\ ft/sec^2 \quad \text{(B.4)}$$

Therefore, we have *1* slug = *32.2* lb_m.

APPENDIX C: IMPORTING *Pro/ENGINEER* PARTS AND ASSEMBLIES

From time to time when you use *COSMOSMotion* for simulations, you may encounter the need for importing solid models from other CAD software, such as *Pro/ENGINEER*. *SolidWorks* provides an excellent capability that support importing solid models from a broad range of software and formats, including Parasolid, ACIS, IGES (Initial Graphics Exchange Standards), STEP (STandard for Exchange of Product data), *SolidEdge, Pro/ENGINEER,* etc. For a complete list of supported software and formats in *SolidWorks,* please refer to Figure C-1. You may access this list by choosing *File* > *Open* from the pull-down menu, and pull down the *Files of type* in the *File Open* dialog box.

In this appendix, we will focus on importing *Pro/ENGINEER* parts and assemblies. Hopefully, methods and principles you learn from this appendix will be applicable to importing solid models from other software and formats.

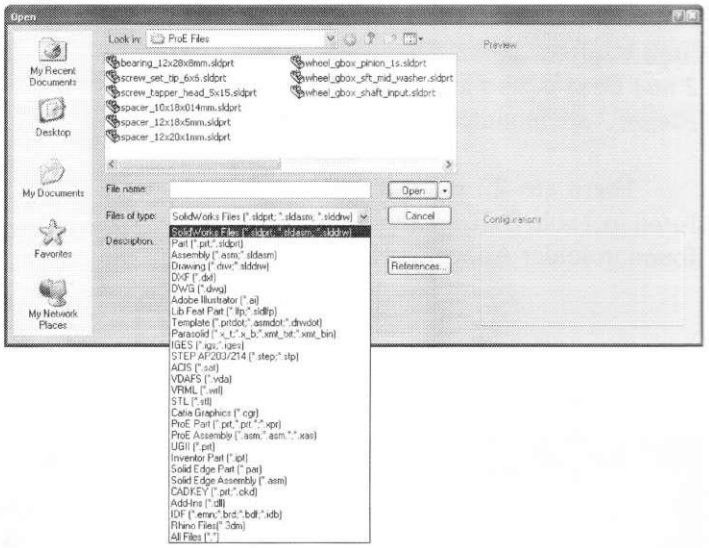


Figure C-1 *File Open* Dialog Box

SolidWorks provides capabilities for importing both part and assembly. Users can choose two options in importing solid model. They are Option 1: importing solid features and Option 2: importing just geometry. Importing solid features may bring you a parametric solid model that just like a *SolidWorks* part that you will be able to modify. On the other hand, if you choose to import geometry only, you will end up with an imported feature that you cannot change since all solid features are lumped into a single imported geometry without any solid features nor dimensions.

Importing geometry is relatively straightforward. In general, *SolidWorks* does a good job in bringing in *Pro/ENGINEER* part as a single imported geometry. In fact, several other translators, such as IGES and STEP, support such geometric translations well. IGES and STEP are especially useful when there is no direct translation from one CAD to another.

Importing solid models with solid features is a lot more challenging, in which solid features embedded in the part geometry, such as holes, chamfers, etc., must be identified first. In addition, sketches that were employed for generating the solid features must be recovered and the feature types, for instance revolve, extrude, sweep, etc., must be identified. With virtually infinite number of possibilities in creating solid features, it is almost certain that you will encounter problems while importing solid models with feature conversion. Therefore, if you do not anticipate making design changes in *SolidWorks.* it is highly recommend that you import parts as a single geometric feature.

We will discuss the approaches of importing parts, and then importing assemblies. In each case, e will try both options; i.e., importing solid features vs. importing geometry. We will use the gear train example employed in *Lesson 6* as the test case and as an example for illustrations.

The Gear Train Example in *Pro/ENGINEER*

The gear train assembly consists of one part and three subassemblies. If you have access to *Pro/ENGINEER*, you may want to open the final assembly, *gear_train_final.asm*, to check the assembled gear train shown in Figure C-2. There are four components in this assembly: *gbox_housing.prt*, *gbox_input.asm*, *gbox_middle.asm*, and *gbox_output.asm*. The input and output gear assemblies consist of one gear each, *Pinion 1* and *Gear 2*, respectively. The middle gear assembly has two gears, *Gear 1* and *Pinion 2*. The four spur gears form two gear pairs: *Pinion 1* and *Gear 1*, and *Pinion 2* and *Gear 2*, as illustrated in Figure C-2. *Gear 1* and *Pinion 2* are mounted on the same shaft.

There are 22 distinct parts in this assembly, as listed in Table C-1. All the parts and assemblies can be found in folder Appendix C.

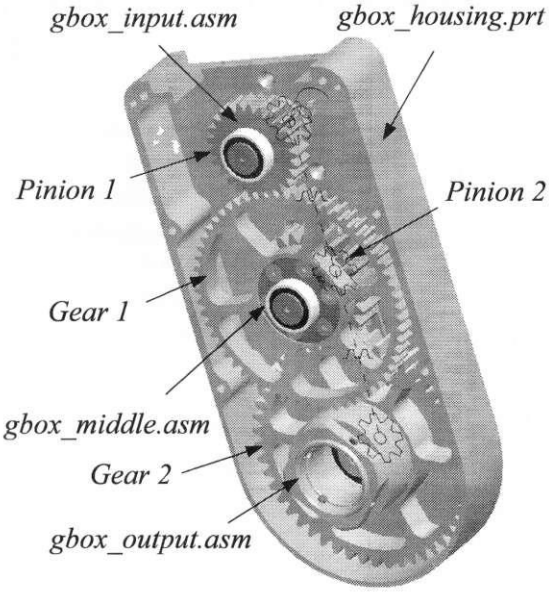


Figure C-2 The Gear Train Assembly

Table C-1 List of Parts and Assemblies in Appendix C Folder

Part/Subassemblies	Part Names	Remarks
gbox_housing.prt		
gbox_input.asm	wheel_gbox_shaft_input.prt	
	wheel_gbox_pinion_1s.prt	Pinion 1
	spacer_12×18×5mm.prt	
	spacer_12×20×1mm.prt	
	bearing_12×18×8mm.prt (2)	
	spacer_10×18×014mm.prt	
	wheel_gbox_sft_mid_washer.prt	
	screw_tapper_head_5×15.prt	
	screw_set_tip_6×6.prt (2)	
gbox_middle.asm	wheel_gbox_pinion_2s.prt	Pinion 2
	wheel_gbox_gear_1s.prt	Gear 1
	wheel_gbox_shaft_mid_pinion.prt	
	wheel_gbox_shaft_mid_gear.prt	
	bearing_12×18×8mm.prt (2)	
	screw_tapper_head_5×28.prt (6)	
	wheel_gbox_sft_mid_washer.prt (2)	
	screw_tapper_head_5×15.prt (2)	
	align_pin_4×27mm.prt (2)	
gbox_output.asm	wheel_gbox_gear_2s.prt	Gear 2
	wheel_gbox_connect_wheel.prt	
	bear_tap_roller25×47×15mm.prt	
	screw_straight_head_4×15.prt (10)	
	align_pin_4×20mm.prt (2)	
	wheel_gbox_connect_wh_setscrew.prt (4)	

sketches in the graphics screen by clicking their names listed in the browser. In addition, the back plate *(Extrudel* in the browser) is recognized incorrectly. Certainly, *SolidWorks* is capable of importing some parts correctly and completely, especially, when the solid features are relatively simple (but not this gear housing part).

If you take a closer look at any of the successful solid features, for example *ExtrudeS,* you will see that the sketches (for example *Sketch3* of *ExtrudeS)* of the solid features do not have complete dimensions. Usually a (—) symbol is placed in front of the sketch, indicating that the sketch is not fully defined.

Apparently, this translation is not satisfactory. Unfortunately, this translation represents a typical scenario you will encounter for the majority of the parts. In many cases, it may take only a small effort to repair or re-create wrong or unrecognized solid features. However, when you translate an assembly with many parts, the effort could be substantial.

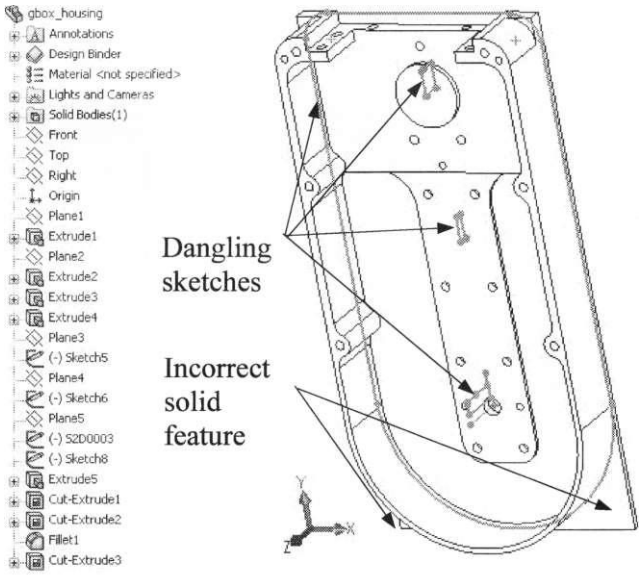


Figure C-7

Option 2: Importing Geometry

Importing geometry is more straightforward and has a higher successful rate than that of importing solid features.

Repeat the same steps to open the gear housing part, *gboxjiousingprt.* In the *Pro/ENGINEER to SolidWorks Converter* dialog box (Figure C-8), choose *Import geometry directly* (default), and then *Kniting* (default) in order to import solid models instead of just surface models. Note that if you choose *BREP* (Boundary Representation), only boundary surfaces will be imported. Click *OK.*

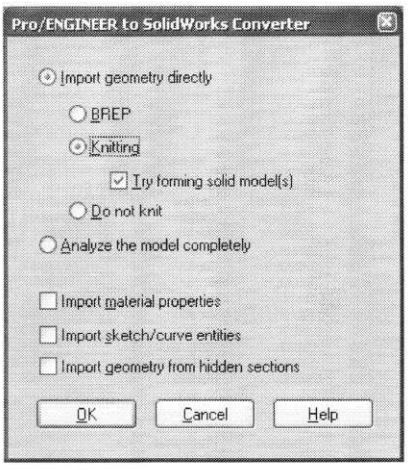


Figure C-8

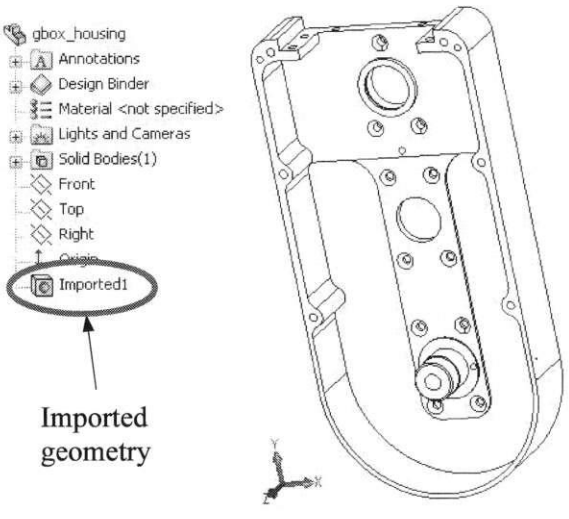


Figure C-9

The conversion process will start. After about a minute or two, the converted model will appear in the graphics screen, as shown in Figure C-9. In addition, an entity *Importedl* will appear in the browser (Figure C-9). As mentioned earlier, there will be no parametric solid feature with dimensions and sketch converted if you choose *Option 2*. However, the geometry converted seems to be accurate. All the geometric features in *Pro/ENGINEER* shown in Figure C-3 were included in this imported feature. This translation is successful. Since we do not anticipate making any change to the gear housing, this imported part is satisfactory. The gear housing part, *gbox housing, sldprt,* employed in *Lesson 6* was created by using *Option 2.*

Importing *Pro/ENGINEER* Assembly

We will import the input gear assembly *(gbox_input.asm)* shown in Figure C-10 using both options. We will try *Option 1* first; i.e.. importing solid features. As shown in Figure C-10 *{Pro/ENGINEER Model Tree* window), there are 11 parts (and several datum features) in this assembly. *SolidWorks* will try to import this assembly as well as the 11 parts from *Pro/ENGINEER.*

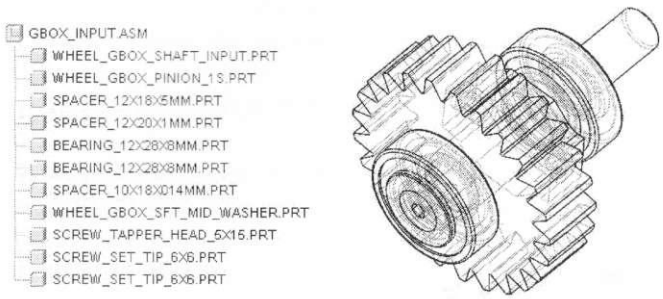


Figure C-10 Input Gear Assembly in *Pro/ENGINEER*

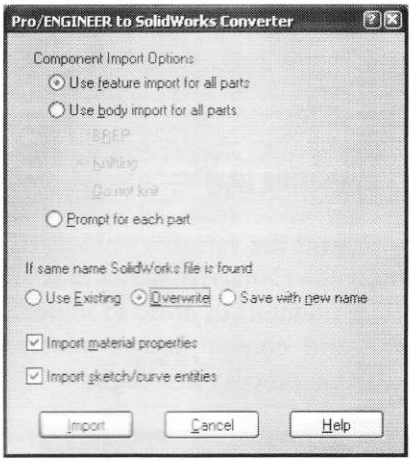


Figure C-11

Option 1: Importing Solid Features

Repeat the same steps to open the input gear assembly, *gbox_input.asm*. In the *Pro/ENGINEER to SolidWorks Converter* dialog box (Figure C-11), choose *Use feature import for all parts*, and choose *Overwrite* for *If same name SolidWorks file is found* (just in case you have *SolidWorks* files with the same file names in the same folder). Choose *Import material properties* and *Import sketch/curve entities*. Click *Import*. The conversion process will start.

You will see sketches, solid features, and solid models appear in the graphics screen. After about a minute, the translation process is completed. The converted assembly and the browser with parts listed are shown in Figure C-12.

As shown in Figure C-12, parts are not completely converted. Major solid features are missing, for example, the pinion 1 (*wheel_gbox_pinion_1s<1>*), where most solid features are not converted. If you expand the part, you will see that only one extrude and one cut features were converted, the remaining entities are mostly sketches. There are only three simple parts converted successfully, *spacer_12×18×5mm<1>*, *spacer_12×20×1mm<1>*, and *spacer_10×18×014mm<1>*. In addition, the *Mates* branch in the browser is empty, implying that no assembly mates have been imported.

Apparently, this translation is not satisfactory. A non-trivial effort will be have to be spent in order to reconstruct the solid features (therefore, solid models) as well as the final assembly.

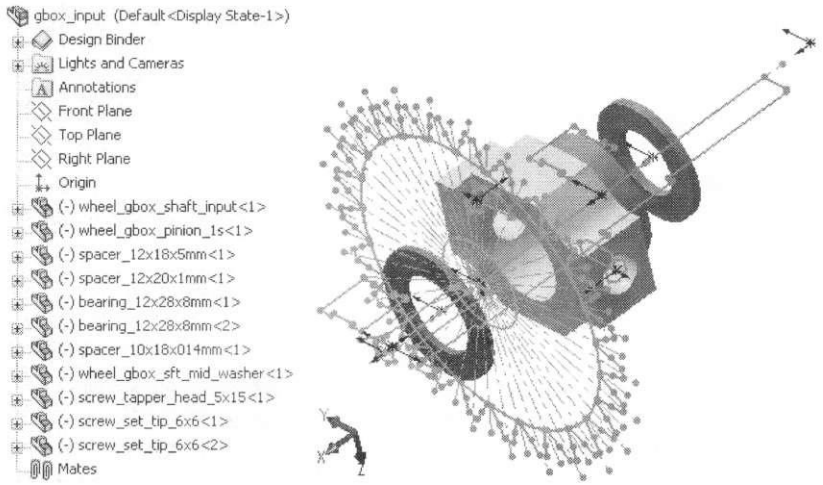


Figure C-12

Option 2: Importing Geometry

Importing geometry is also more straightforward for assembly and has a higher rate of success.

Repeat the same steps to open the input gear assembly, *gbox input.asm.* In the *Pro/ENGINEER to SolidWorks Converter* dialog box (Figure C-l 3), choose *Use body import for all parts* (default), and then *Kniting* (default) in order to import solid models. Choose *Overwrite* for *If same name SolidWorks file is found,* and choose *Import material properties* and *Import sketch/curve entities.* Click *Import.* The conversion process will begin.

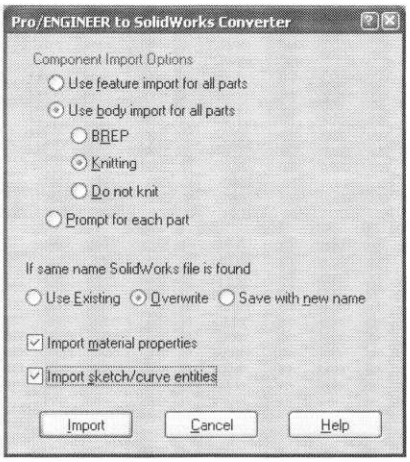


Figure C-13

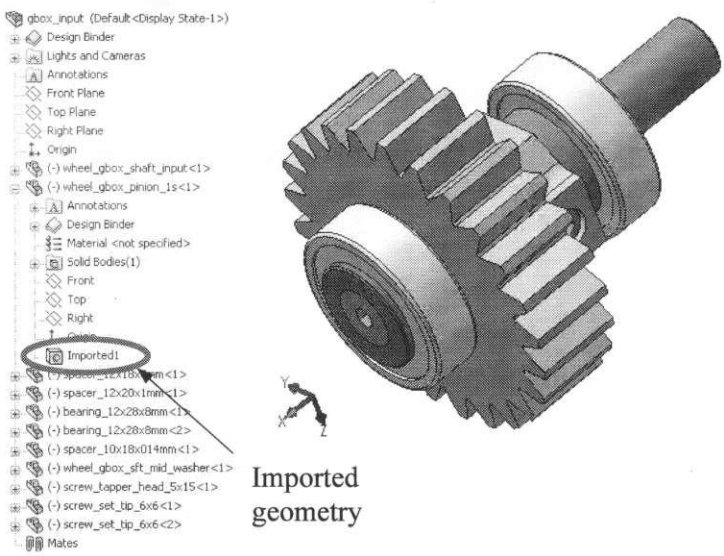


Figure C-14

After about a minute or two, the converted assembly will appear in the graphics screen, as shown in Figure C-l4. The assembly and all 11 parts seem to be correctly imported. If you expand any of the part branch, for example, the gear *(wheel_gbox_pinion_ls<l>),* you will see an imported feature listed, as

depicted in Figure C-14. Again, there is no solid feature converted in any of the parts. In addition, the *Mates* branch is empty.

Since we do not anticipate making any change to this input gear assembly, this imported assembly is satisfactory, except it does not have any assembly mates. Assemble all 11 parts (may be more for some cases) will take a non-trivial effort. Since we do not anticipate making change in how these parts are assembled, we will merge all 11 parts into a single part.

In *SolidWorks,* you can join two or more parts to create a new part in an assembly. The merge operation removes surfaces that intrude into each other's space, and merges the parts into a single solid volume. We will insert a new part into the assembly and merge all 11 parts into that new part.

Choose from the pull-down menu *Insert* > *Component* > *New Part.* In the *Save As* dialog box, enter *gbox Jnput* for part name, and click *Save. SolidWorks* is expecting you to select a plane or a flat face to place a sketch for the new part. Click a plane or planar face on a component, for example pick the assembly *Front* plane from the browser, the *Front* plane will appear in the graphics screen (Figure C-15). The new part *gboxinput<l>* will appear in the browser. In the new part, a sketch opens on the selected plane. Close the sketch. Because we are creating a joined part, we do not need a sketch.

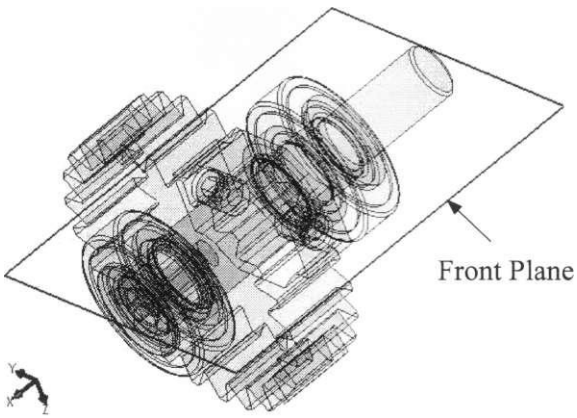


Figure C-15

Next, we will select all 11 parts and merge them into the new part.

From the browser, click the first part *wheel box_shaftinput<l>,* press the *Shift* key, and then click the last part, *screw_setjip_6*6<2>.* All 11 parts will be selected.

From the pull-down menu, choose *Insert* > *Features* > *Join.* The *Join* window will appear (overlapping with the browser) as shown in Figure C-16. In the *Join* window, all 11 parts are listed. All you have to do is to click the checkmark on top to accept the parts. Save the assembly (and the part), and then close the whole assembly.

Now open the part *gbox Jnput.* Make sure you open *gbox input, sldprt* instead of *gboxJnput. sldasm.* The part *gbox input* will appear in the graphics screen. In addition, all entities belong to this part will be listed in the browser, as shown in Figure C-17. Note that there is an arrow symbol -> to the right of the root entity, *gbox Jnput.* This symbol indicates that these entities enclosed in this part refer to other parts or assembly. Note that the *Joinl* branch has the same symbol. Expand the *Joinl* branch, you will see 11 parts listed, all with arrows, pointing to the actual parts currently in the same folder. When the link is broken; i.e., when the referring parts are removed from the folder, a question mark symbol will be added to the arrow.

The three gear parts, *gbox Jnput. sldprt, gboxjniddle.sldprt,* and *gbox output, sldprt,* employed for *Lesson 6* were created following the approach discussed. One axis in each part that passes through the center hole of the gear was created simply by intersecting two planes, for example, *Top* and *Right* planes for the axis in *gbox Jnput. sldprt,* as shown in Figure C-18. These axes are necessary for creating gear pairs, as discussed in *Lesson 6.*

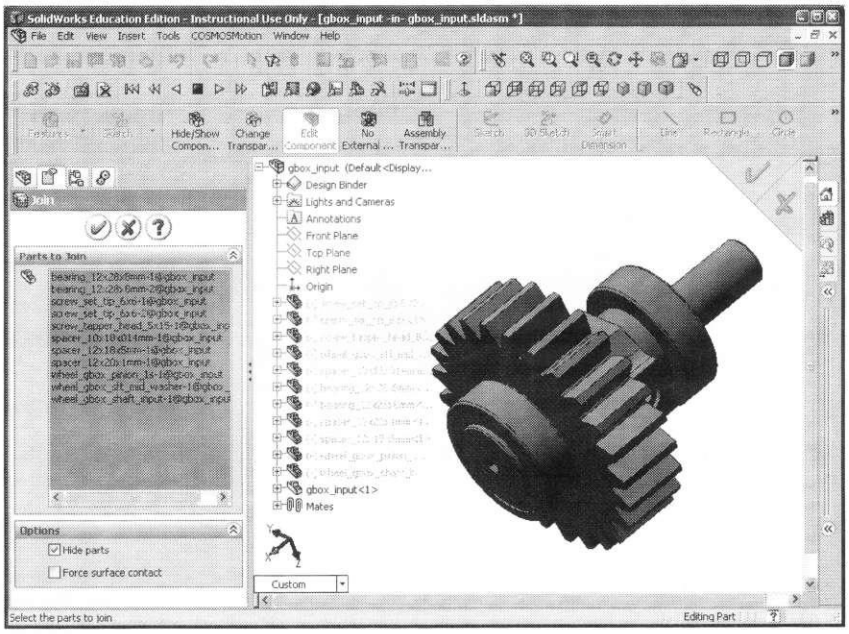


Figure C-16

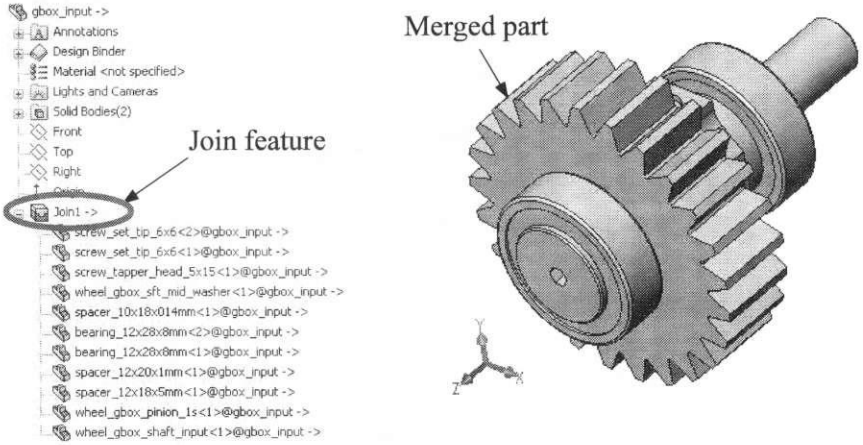


Figure C-17

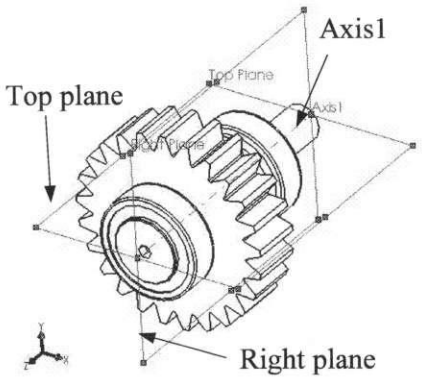
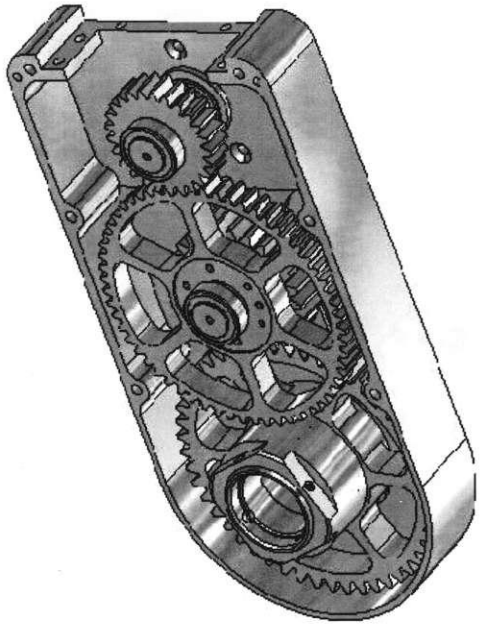
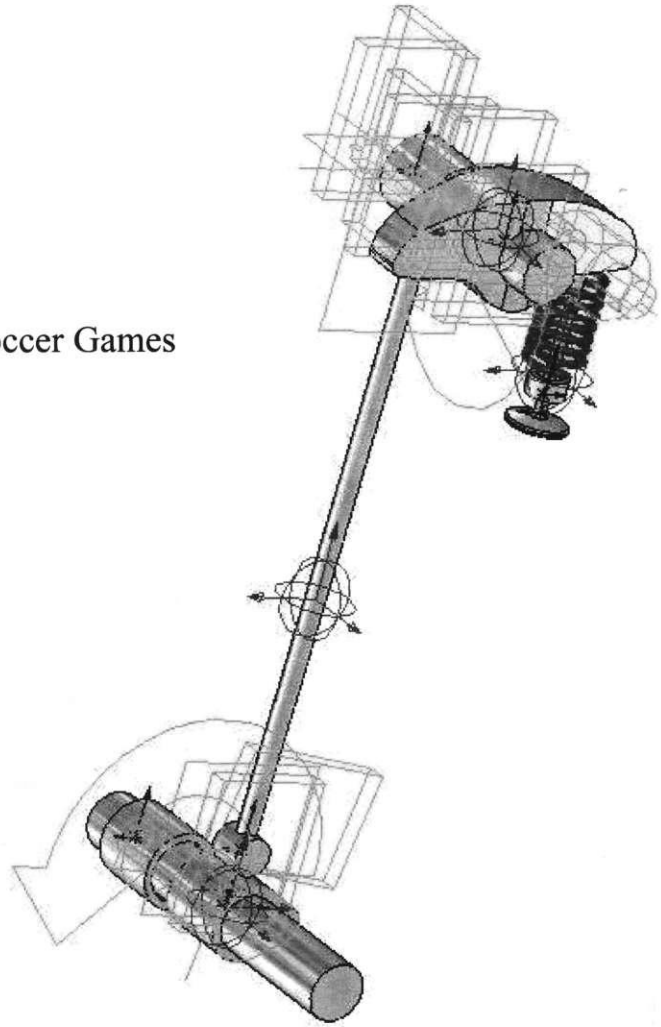


Figure C-18

Motion Simulation and Mechanism Design
with COSMOSMotion 2007

Table of Contents





SDC
PUBLICATIONS

Offering a wide variety of
CAD, FEA and Engineering Graphics Books

Schroff Development Corporation
www.schroff.com

ISBN 978-1-58503-482-6

Better Textbooks. Lower Prices.