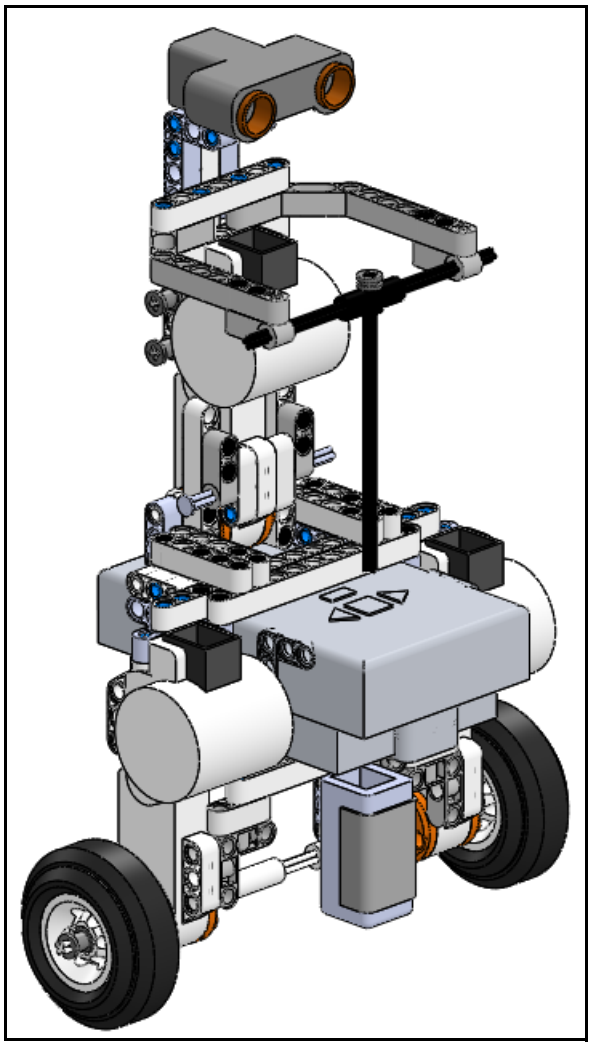
SolidWorks Education
Learn. Create. Succeed.

Engineering Design and Technology Series

SolidWorks Lego Mindstorm



Dassault Systèmes SolidWorks Corporation,
175 Wyman Street
Waltham, Massachusetts 02451 USA
Phone: +1-800-693-9000

Document Number:

Contents

1

NXT SegWay Robot

When you complete this lesson, you will be able to:

- Setup a motion simulation.
- Use functions and expressions.
- Create a basic control system.

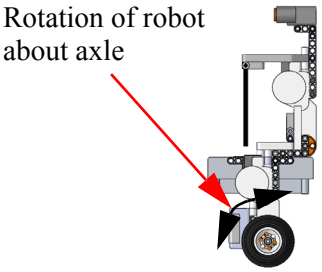
SegWay Transportation

SegWay Personal Transporter (PT) is the world's first self balancing transportation device with two wheels. The mechanism is based on the principle of the inverted pendulum with advanced control to maintain balance at all times. To move, the rider leans forwards or backwards and the transporter accelerates in the proper direction to balance the system. Similarly, to take turns the rider leans sideways and the transporter adjusts the speed of both wheels. Handling feels natural because the amount of lean, measured by the gyroscope, controls the acceleration and the curvature of turns.

Project Description

In this project, you will simulate the SegWay robot built using the Lego Mindstorms NXT 2.0 kit. In the first part of the lesson, you will create a system control in SolidWorks Motion and compare it to the behavior of the robot. You will consider the following scenario.

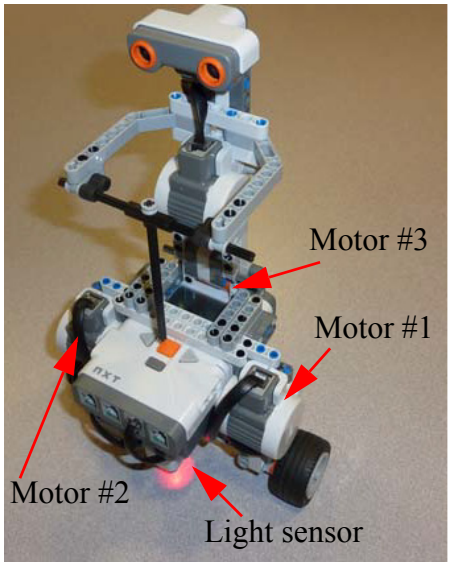
- **Single Degree of Freedom (DOF) system**
  The rider and transporter are rigidly connected to form a single mass. The only degree of freedom is then the rotation about the axle.



The behavior of the robot under various circumstances can be viewed in video files supplied with the project and at http://www.nxtprograms.com/NXT2/segway/.

Balance Control

The real SegWay transporter uses input from a gyroscope to control the stability and balance. Because a gyroscope is not part of the Lego kit, SegWay robot makes use of the light sensor. The sensor registers the light intensity when the robot is started. As the robot tilts, the light intensity changes and the robot reacts by adjusting power to motors #1 and #2. As a consequence, the robot cannot maintain vertical position; it tries to maintain the position at which it is started. It is therefore necessary to start the robot at near vertical orientation.



SolidWorks Model

Follow the instructions below to build your SegWay robot assembly. All SolidWorks part files are supplied as part of this project, so you only need to mate them to create a final assembly. Because the basic assembly building skill is a prerequisite to this lesson, the steps below do not detail how to mate parts.

It is a good habit to think about the assembly building procedure ahead of time if we want to follow with a simulation in SolidWorks Motion. Proper methodology will result in a smooth and faster computation, while improperly built assemblies may cause the solver to lock and terminate the solution. First, review how the mechanism moves and how many degrees of freedom it features. Then decide on how to split the top level assembly into subassemblies. This is important because SolidWorks Motion typically considers subassemblies as rigid, resulting in a less complex model to solve.

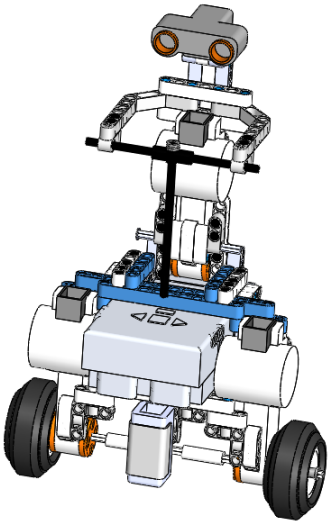
Tip: The subassembly's status can be changed from rigid to flexible. It is, however, recommended that you keep sub-assemblies set to rigid.

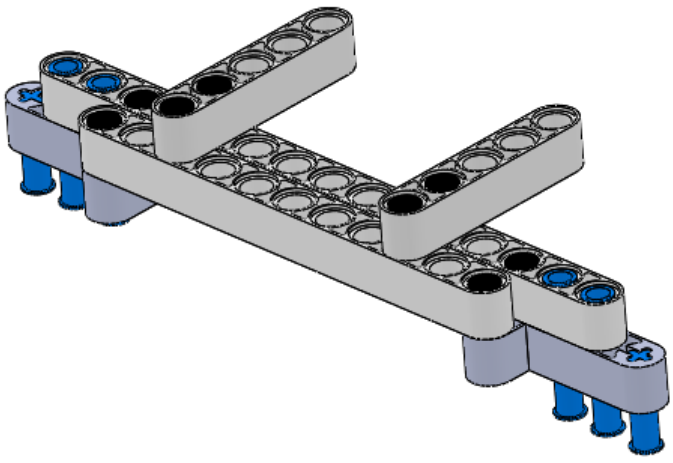
The procedure below recommends one way how to split the robot into individual subassemblies, which are then mated to create the full SegWay robot. You do not have to follow these steps rigorously. Rather, experiment and find your own way to build the SegWay robot.

Steps **1** to **8** suggest how to split the robot into the individual subassemblies. Steps **9** to **23** continue with the instructions on how to assemble the robot and orient it in the global coordinate system.

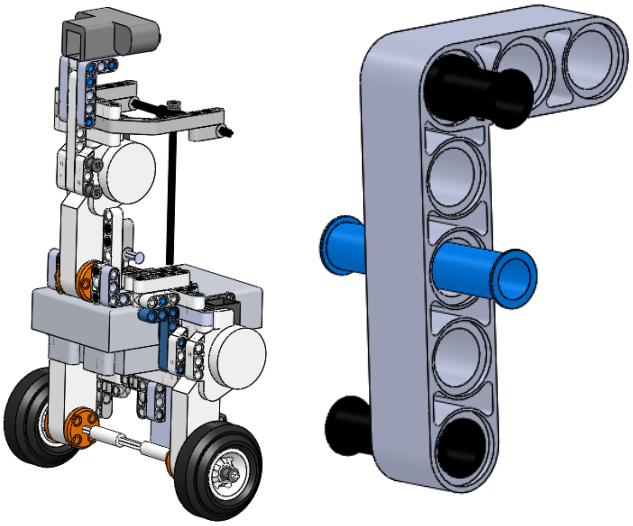
Build the following subassemblies.

1 **Horizontal connector.**

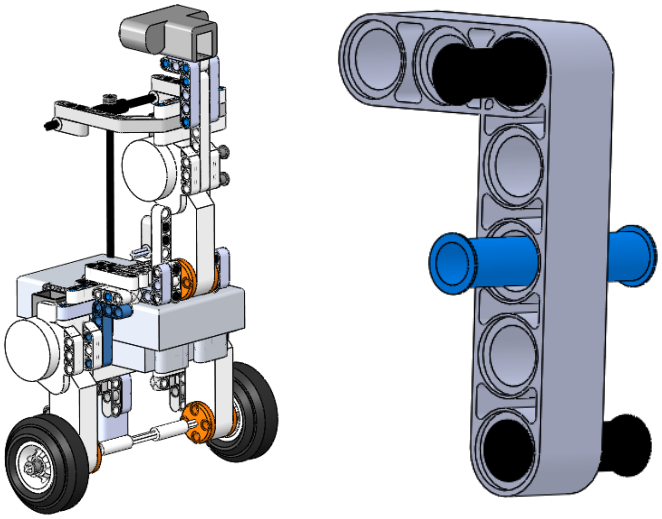




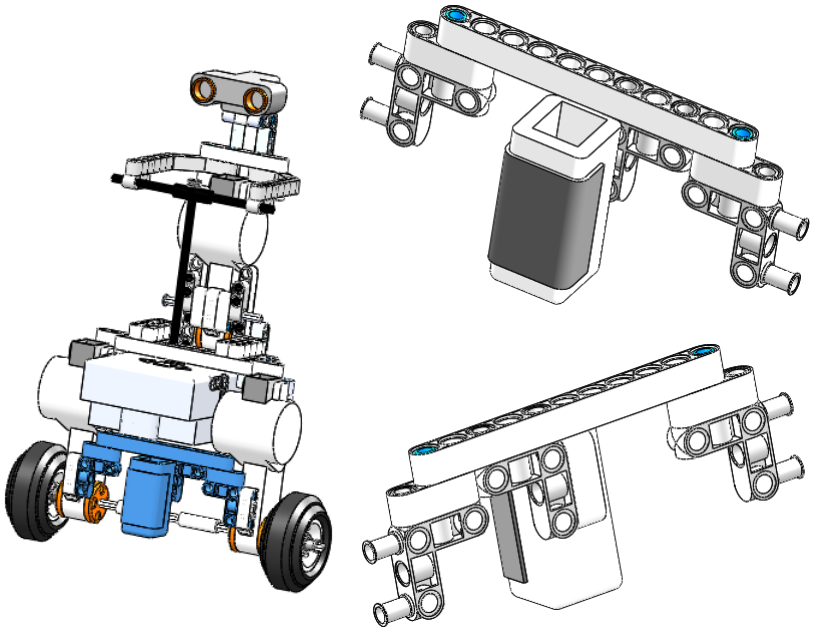
2 **Left connector.**



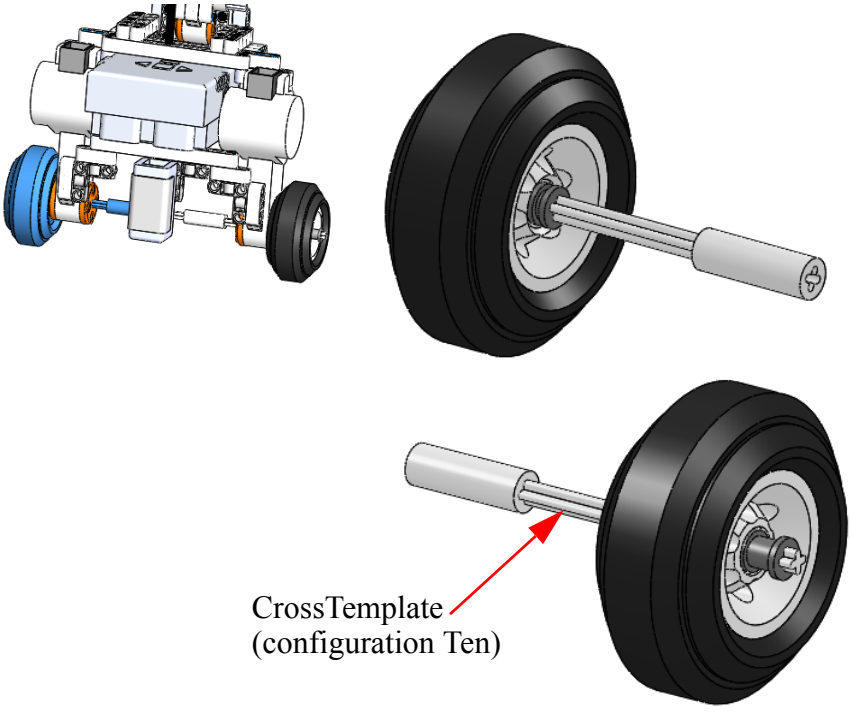
3 **Right connector.**



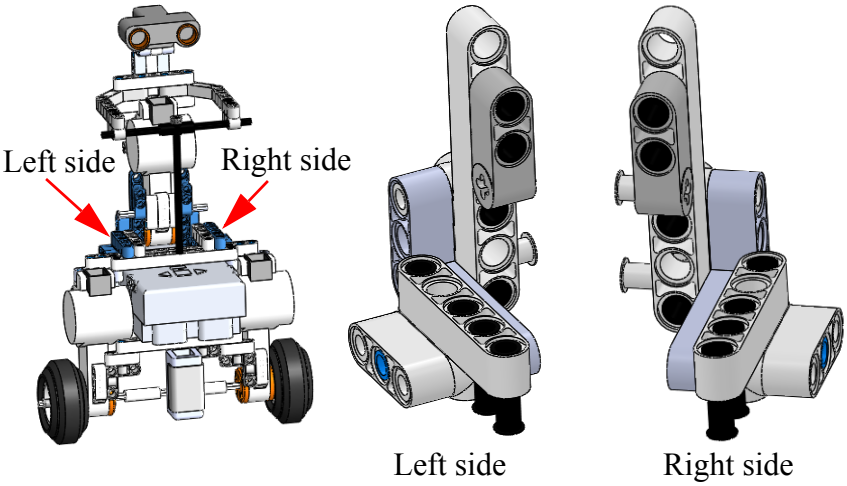
4 Lower horizontal connector.



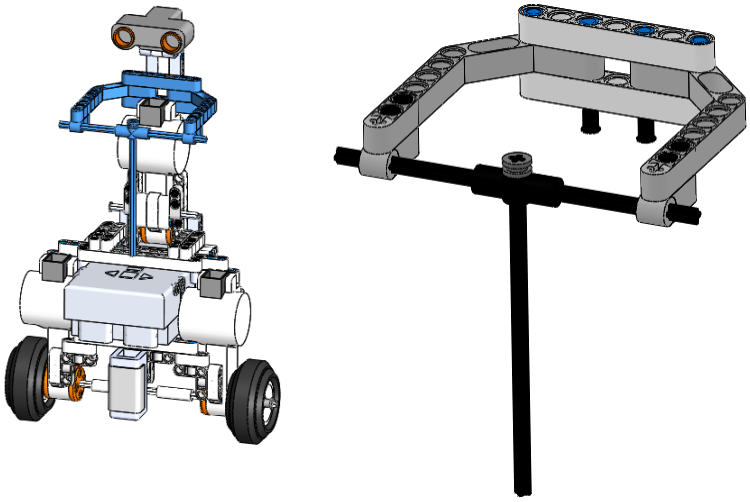
5 Wheel and axle.



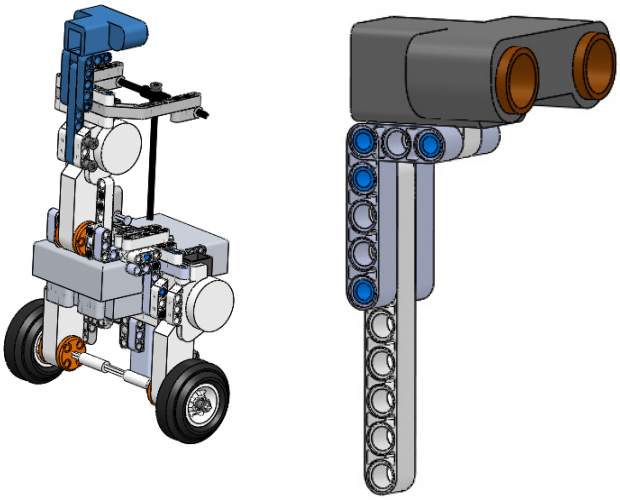
6 Left and right side of the rider bottom.

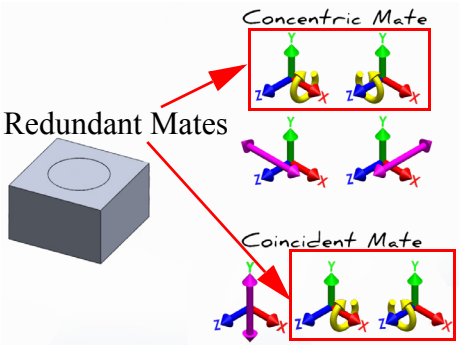


7 Arms and chest.



8 Rider’s head.





The following steps contain instructions to build the SegWay robot using the subassemblies and additional parts. Some suggested ways of mating the components are given in certain steps in order to eliminate redundant mates in the assembly. Redundant mates occur when two mates remove the same degree of freedom from a component. For example, when using a coincident and a concentric mate to mate a cylinder to a cylindrical hole in a block, both mates remove the rotation about the X and Y axis. This example can be seen in the image to the right. Reducing the redundant mates in an assembly helps to increase the accuracy of the motion simulation and reduce the potential for problems.

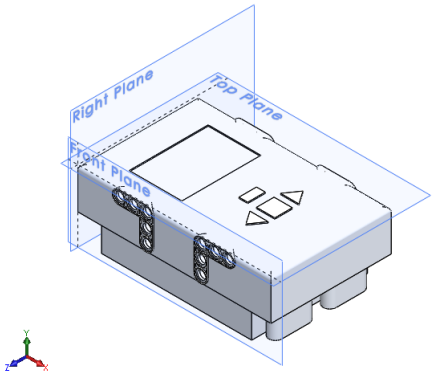
9 **SegWay assembly.**

Open a new assembly document.

Insert the `CPU` part, align it with the coordinate system and the planes as shown and **Fix** it.

Set the units to MMGS.

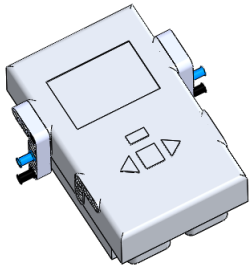
Save the assembly as `Segway.SLDASM`.



Note: Different orientations of the `CPU` is fine and would not have any effect on the accuracy of the simulation. However, to maintain consistency with the results published in this document, you must orient the `CPU` part as shown above.

10 **Left and right connectors**

Insert both the left and right connectors and mate them with the `CPU`.

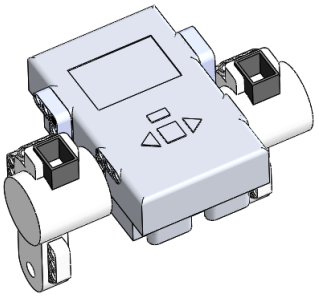


Tip: While we recommend using hinge and concentric mates, other mate types are acceptable as well.

11 **Motors.**

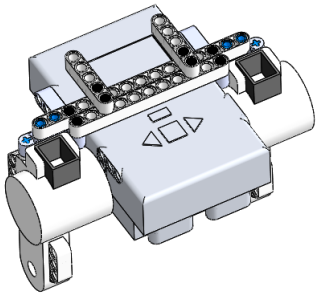
Insert the `Motor` part for motors #1 and #2.

Mate them to the left and right connectors.



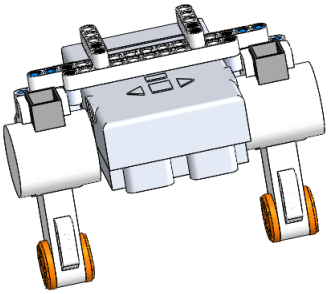
12 Horizontal connector.

Insert the `horizontal connector` and mate it to `Motor` parts #1 and #2.

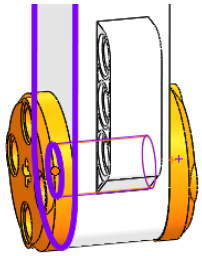


13 Rotors for motors #1 and #2.

Add two `Rotor` parts and mate them to `Motor` #1 and #2.



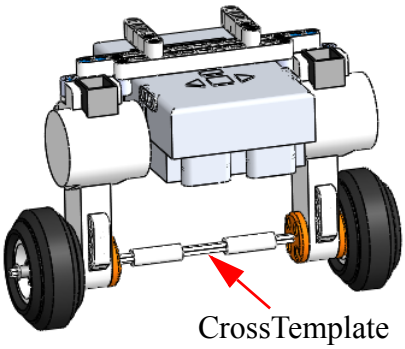
Tip: We recommend that you use a concentric mate and a coincident mate between the `Rotor` origin and the side face of the `Motor` to connect the rotors with both motors. Doing this will remove any redundancies in the mates between these two parts.



14 Axle and wheels.

Insert the `wheel and axle` subassembly and mate it to the `Rotor` parts.

Insert the `CrossTemplate` part (configuration Five) and center it. Mate the `CrossTemplate` part to either of the two wheels and axle subassemblies (but not both).

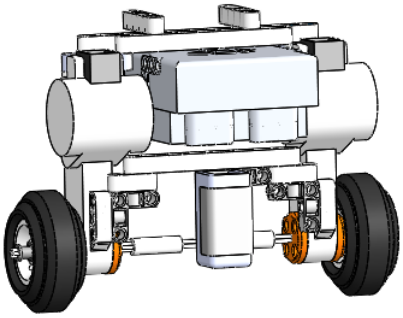


Important!

Because the robot wheels are driven by two independent motors, the wheel and axle subassemblies on both sides must be disconnected. This way you could make the robot turn by adjusting power at either motor #1 and #2.

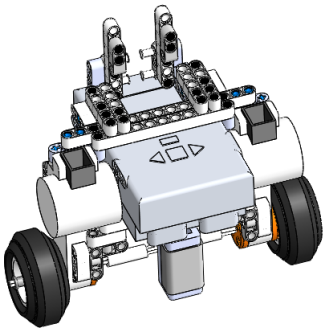
15 Lower horizontal connector.

Insert the `lower horizontal connector` subassembly and mate it to `Motor` #1 and #2.



16 Left and right bottom sides of the rider.

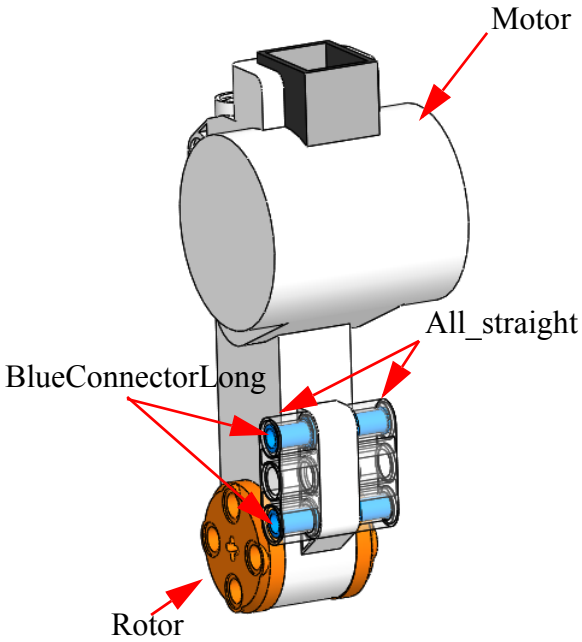
Mate both subassemblies to the horizontal connector.



17 Motor #3 and its rotor.

Insert new `Motor` and `Rotor` parts and mate them.

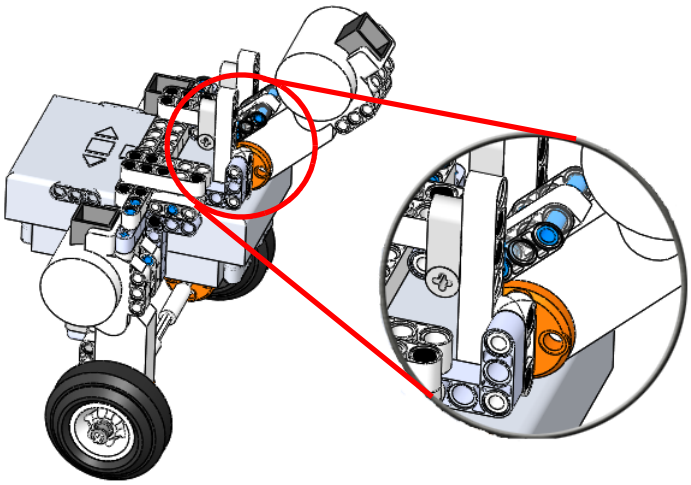
Insert two instances of `All_Straight` (configuration three) and two instances of `BlueConnectorLong` parts. Mate them to the new `Motor` #3 part, as shown in the figure.



18 Connect Motor #3.

Mate the `Rotor` from the previous step to the left and right bottom parts of the rider.

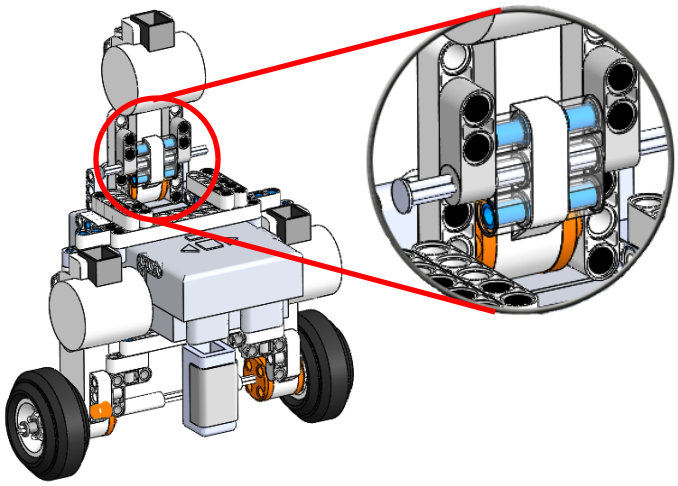
Notice, that `Motor` #3 can rotate about the `Rotor` axis. This degree of freedom will be eliminated in the following step.



19 Orient Motor #3.

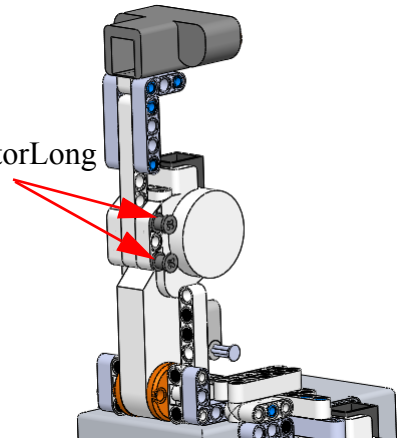
Using the `CrossTemplateWithCap` (configuration `Eight`), align `Motor` #3 with the rest of the assembly so that it assumes a vertical position.

Make sure to eliminate the horizontal motion of the `CrossTemplateWithCap` part by adding an extra mate.

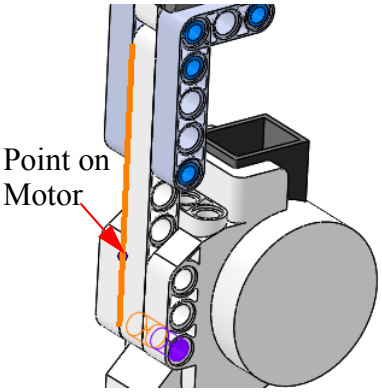


20 Rider's head

Insert the head subassembly. Using two instances of the `CrossConnectorLong` part attach it to `Motor` #3.

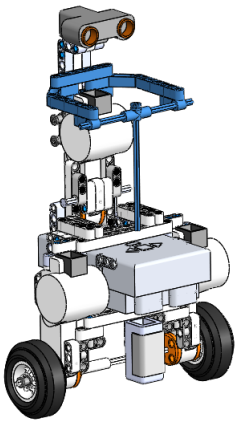


Tip: We recommend that you use a concentric mate and a coincident mate between a point on the back edge of the `Motor` and edge of the `All_Straight` part in the `rider's head` subassembly as seen in the image on the right. Doing this will remove any redundancies in the mates between these two parts.



21 Arms and chest.

Complete the SegWay robot assembly by attaching the arms and chest subassembly.

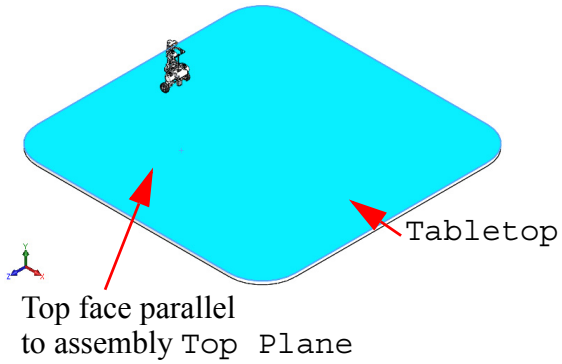


The next two steps insert the platform (Tabletop) and place the robot to its initial position and orientation.

22 Tabletop.

Insert the Tabletop part.

Position the Tabletop so that the robot is centered and closer to the back side. Orient it as shown by the triad in the figure and make it parallel to the assembly Top Plane.



Use **Tangent** mates between the tires and the top face of the Tabletop so that the tires touch the surface.

Fix Tabletop.

Note: Utilize **Use for positioning only** mates to align Tabletop with the Top Plane and for the **Tangent** mates. These mates are only used to position and orient the Tabletop part.

23 Float robot.

Change the status of the CPU part from **Fixed** to **Float**.

Note: Floating the CPU makes the robot free and ready to move in SolidWorks Motion. Also, because the CPU was aligned with the assembly plane (step **9**), the robot assumes a vertical position.

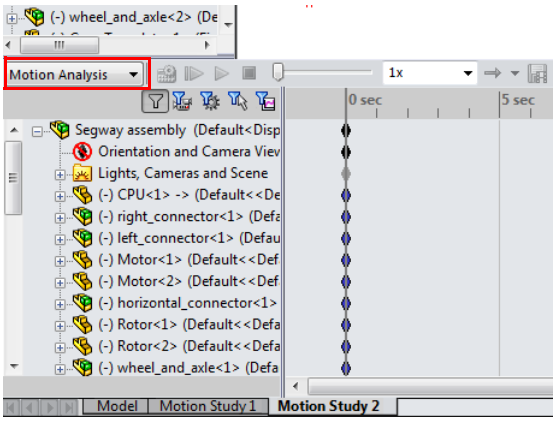
SolidWorks Motion Model

The following section will guide you through the setup of the simulation and basic controls in SolidWorks Motion. As you work through this part, you will gradually improve the control algorithm so that, in its final stage, it correlates with what you observe in reality.

It is assumed that you are familiar with the basics of the SolidWorks Motion software. If that is not the case, please ask your instructor to advise you on how to obtain more elementary training materials to make yourself comfortable with the software.

24 Motion study.

Add in SolidWorks Motion and insert a **New Motion Study** and change the **Type of Study** to **Motion Analysis.**



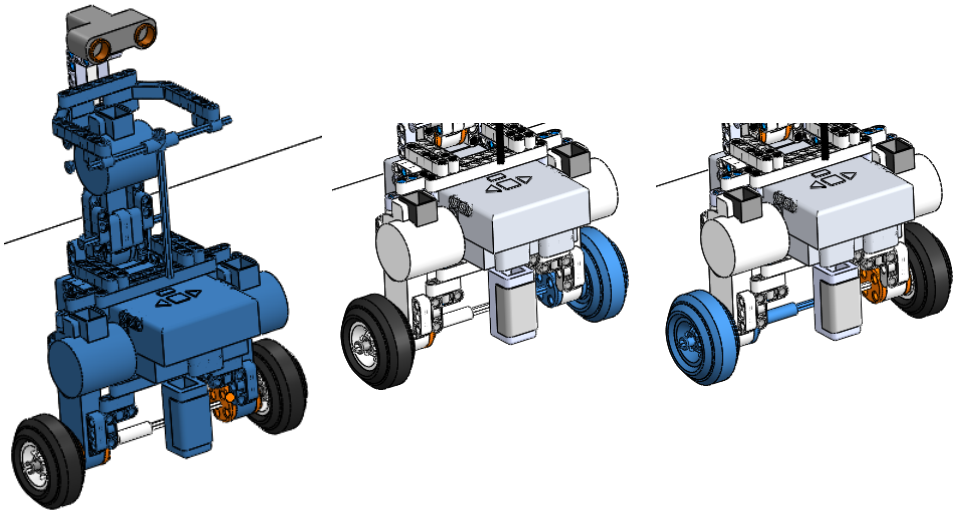
Rigid Groups

Review the top assembly mates; the total number of mates can range from 30 to 50 depending on how efficiently you mated the components. Mates are responsible for the description of the relative motion between the mated parts, i.e. hinge mates only allow two connected parts to rotate relative to each other about the hinge axis. If solving the current model, SolidWorks Motion must consider all mates as locations where motion may occur (even if you know that the only resulting motion of the robot is rotation about the wheel axle). We can effectively use rigid groups to simplify the calculation.

A rigid group combines parts and subassemblies in a group and treats them as one mass, significantly reducing the number of mates that need to be considered by SolidWorks Motion. Three rigid groups can be created in our case:

- Entire body of the robot, excluding the `rider's head` subassembly,
- Left wheel and axle subassembly and the corresponding `Rotor`,
- Right wheel and axle subassembly, corresponding `Rotor`, and the `CrossTemplate` part.

The image below shows all three rigid groups.

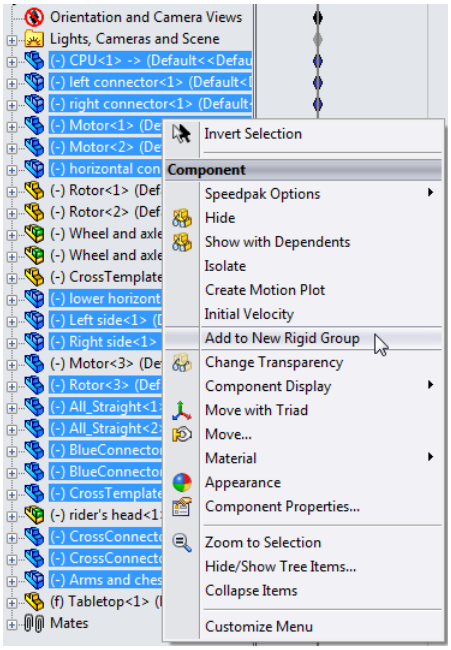


Note: The `rider's head` subassembly was not included in the rigid group because we will use this to measure the robot's tilt, angular velocity and angular acceleration.

25 Rigid groups.

Using the CTRL button, multi-select all parts but the `rider's head`, two `Rotor` parts for motors #1 and #2, two wheel and axle subassemblies, `CrossTemplate` and the `Tabletop` part.

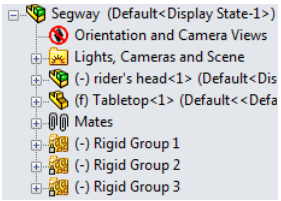
Right-click any selected part and click **Add to New Rigid Group**.



Continue with the definition of the two remaining rigid groups:

- Left wheel and axle subassembly and the corresponding `Rotor` part, and
- Right wheel and axle with its corresponding `Rotor` and `CrossTemplate` part.

After you complete definitions of the three rigid groups, SolidWorks Motion feature tree should show only two free parts: `rider's head` and `Tabletop`.



This step significantly simplifies the SolidWorks Motion model which needs to be solved.

Note: The number in the names of the rigid groups in your case may differ from the numbers shown in the figure.

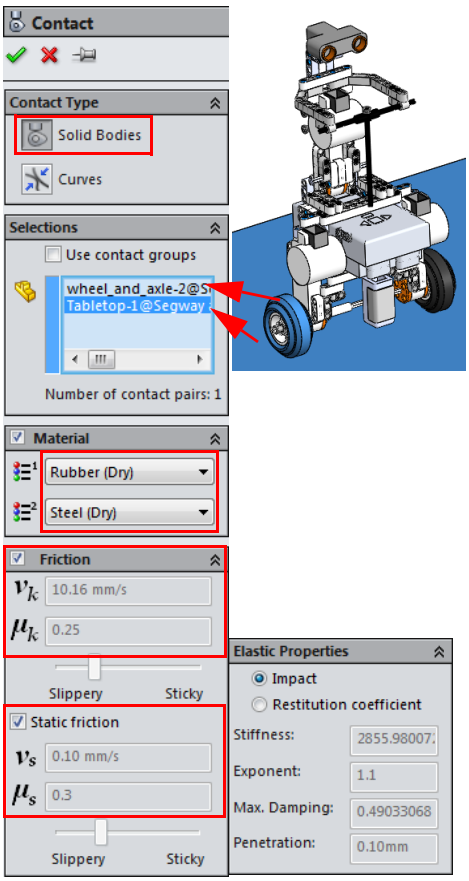
26 Contacts.

Define **Solid Bodies** contact condition between the left tire and `Tabletop`.

Specify `Rubber (Dry)` and `Steel (Dry)` for **Material**. This selection also determines the default friction coefficients.

Make sure that not only default kinematic friction but also **Static friction** is activated.

Repeat the definition for an identical contact between the right tire and `Tabletop`.



27 Gravity.

Activate **Gravity** in the negative Y direction.

28 Study properties.

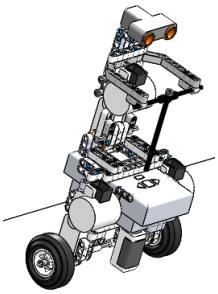
To save sufficient data on the disk in order to plot smooth graphs, set **Frames per second** to **400**.

29 Calculation.

Set the duration of the simulation to **0.5s** and **Calculate.**

30 Review motion.

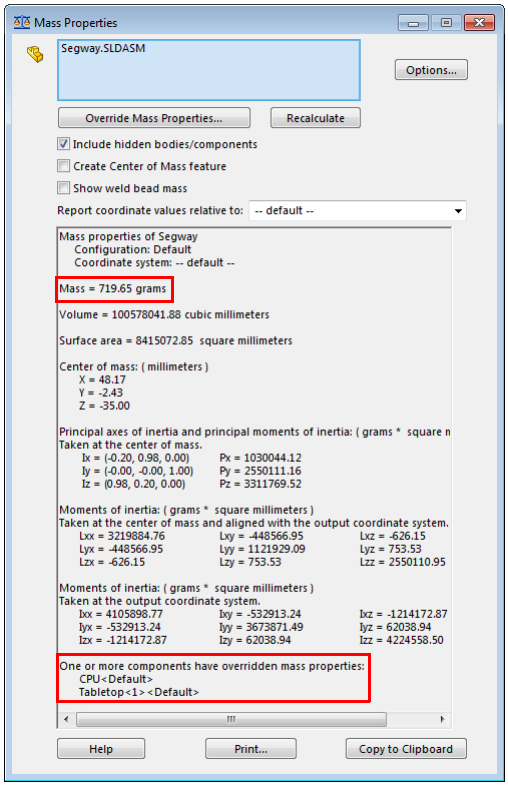
Play the motion at low speed. Observe the robot tilt forward as the wheels move backward. This is a sign that the robot is not balanced even though it is initially oriented vertically. The center of mass is shifted somewhat forward.



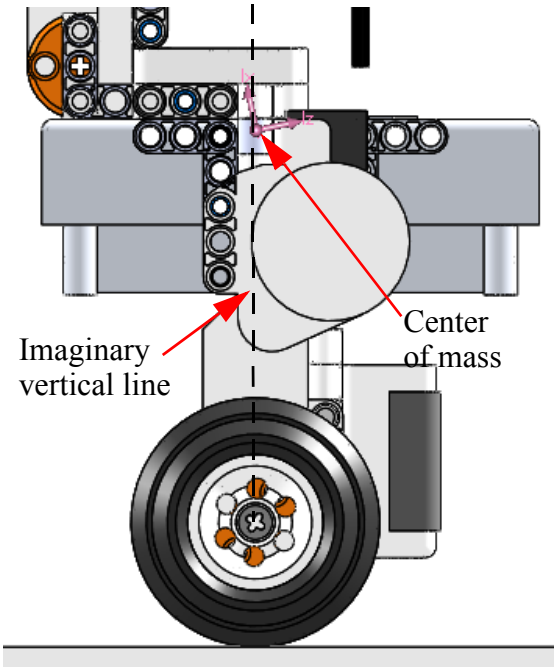
31 Mass properties.

Using the **Mass Properties** command (under **Tools**) show the mass of the robot.

The total mass of the robot is 719.65 grams. Also notice that the `CPU` and `Tabletop` parts have assigned mass properties. The `CPU` mass is increased to account for six AAA batteries, and the mass of the `Tabletop` is reduced to 0.001 grams to minimize its effect on the total mass calculation.

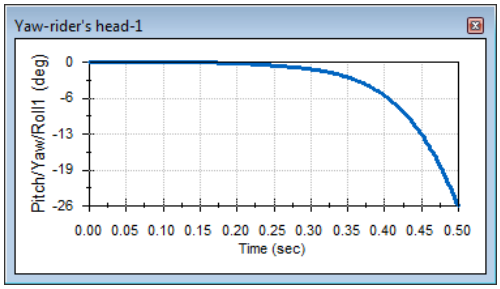


Locate the center of mass. Notice, that it is just a little off the imaginary vertical line passing though the center of the wheels. This agrees with the observed motion where, without any external inputs but the gravity, the robot tilts forward.



32 Tilt.

To measure the tilt, create a plot for the **Yaw** (under **Other quantities**) of the `rider's head`.



Note: The `rider's head` above is requested in order to maintain consistency between the result plots presented in this document and those generated in your simulation. If using parts other than `rider's head`, yaw may no longer be the desired quantity to monitor. Notice, that when the robot tilts forwards, the yaw magnitude assumes negative values.

Tip: Go to Wikipedia.org and study the definition of yaw, pitch, and roll. Can you explain why we monitor yaw of the rider's head to measure the tilt?

We will also have to monitor how fast the tilt changes. This can be measured by angular velocity (rad/sec) and angular acceleration (rad/sec^2). You will generate these plots at a later time.

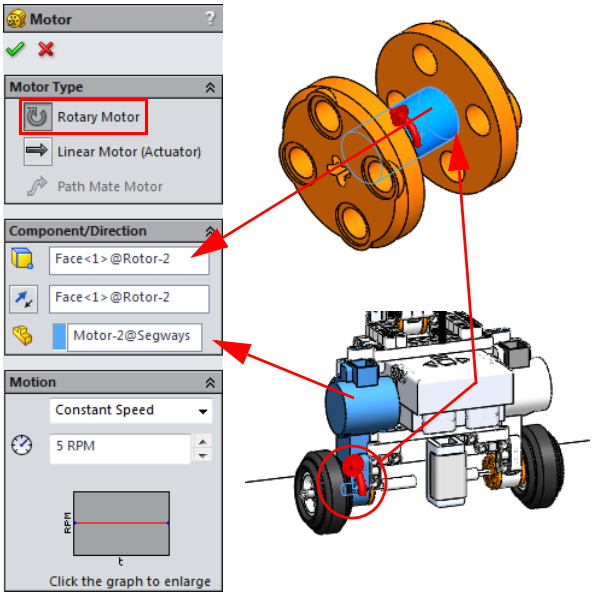
33 Motors.

Isolate the `Rotor` part connected to `Motor` #2.

Apply **Rotary Motor** to the cylindrical face of the `Rotor` as shown in the figure.

Select the `Motor` part as **Component to move relative to**.

Specify **Constant speed** of **5 RPM**. Orient the rotary motor feature as shown in the figure. (Positive rotation of the rotary motor should make the robot move along the negative X axis.)



Apply an identical **Rotary Motor** to the `Rotor` connected to Motor #1.

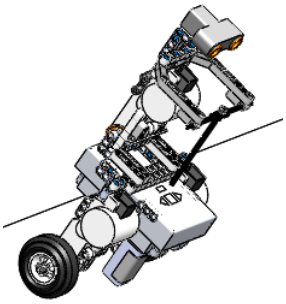
34 Calculation.

Calculate the simulation.

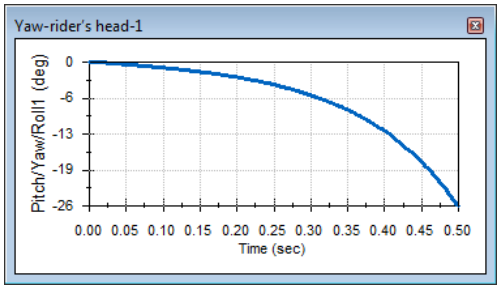
35 Review motion.

Play the motion at low speed. This time, observe the robot tilt forward as the rotary motors make the wheels move backward.

It is clear that to keep the robot balanced, we have to monitor the tilt of the robot, regard it as an error, and adjust the value of the rotary motors powering the wheels. Somewhat similar methodology is also used in the case of the real robot.



36 Tilt



Review the **Yaw** plot for the `rider's head`. Notice, that the yaw magnitudes are negative when the robot tilts forward.

The RPM of the rotary motor features must be adjusted accordingly based on the magnitude of the tilt. Let us try to alternate the magnitude of the rotary motors from -5 RPM to +5 RPM, depending on the sign of the yaw. To implement this elementary control algorithm, an IF logical statement must be used.

IF Statement

The IF statement is used to define an output based on the sign of an input variable.

It is in the form of:

IF (Input variable: A,B,C)

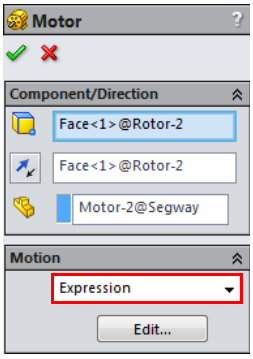
When the value of the Input variable is negative, output the value A.

When the value of the Input variable is zero, output the value B.

When the value of the Input variable is positive, output the value C.

The Input variable, A, B and C can all be either fixed values or expressions.

37 Controlled velocity in rotary motors.



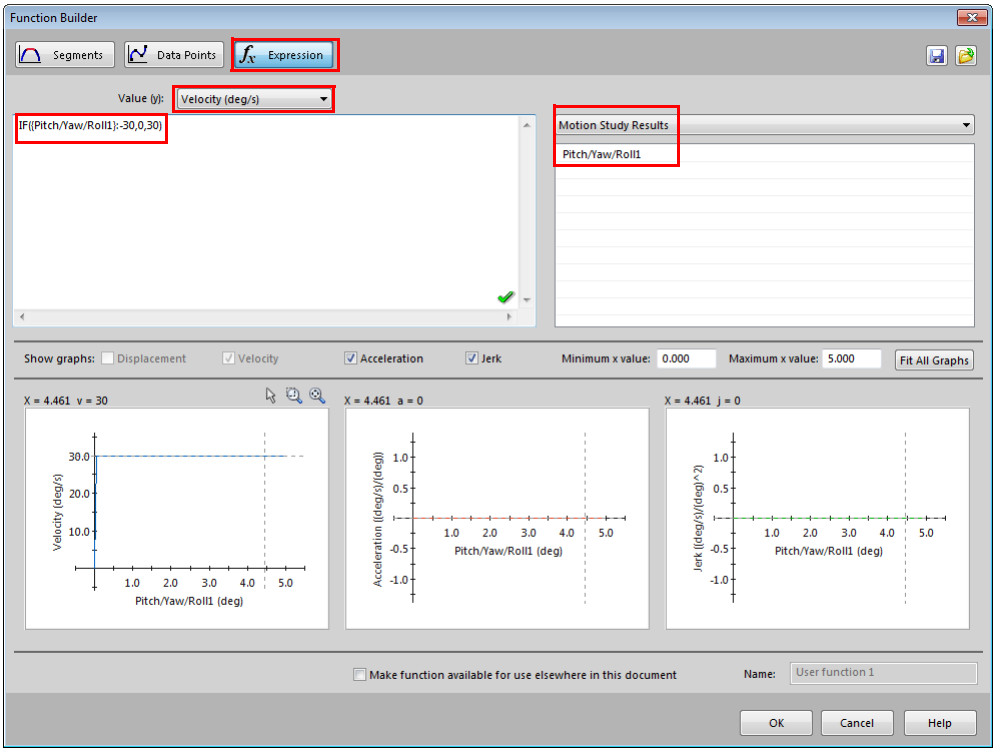
Edit both rotary motor features.

Under **Motion** select **Expression** for **Velocity**.

All results plots defined in the motion simulation are listed in the **Motion Study Results** field. As of now, you should see the identifier for the yaw plot **{Pitch/Yaw/Roll1}**, generated in step **32**.

Note: The plot identifier in your simulation may have a different number attached to it, i.e. {Pitch/Yaw/Roll3}.

In the **Expression** field, enter **IF({Pitch/Yaw/Roll1}:-30,0,30).**



Note: To copy the plot identifier **{Pitch/Yaw/Roll1}** into the expression field, double-click this identifier in the **Motion Study Results** field. Also, when using expressions to assign the velocity to rotary motors, the units of deg/sec are used (1 RPM = 6 deg/sec, so 5 RPM = 30 deg/sec).

This expression assigns 5 RPM to the motor if the yaw value is positive, and -5 RPM if the yaw value is negative.

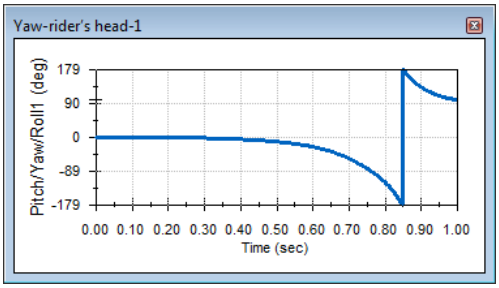
38 **Calculation.**

Set the time of the simulation to **1s** and **Calculate**.

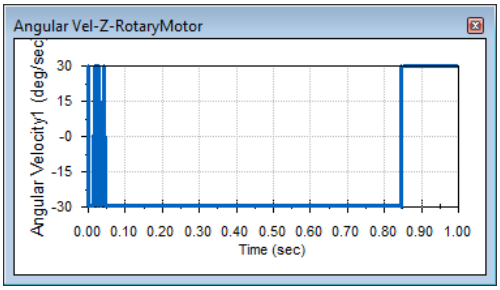
39 **Tilt and motor velocity.**

Review the **Yaw** plot for the `rider's head`.

The robot undergoes a continuous forward tilt, an indication that the control algorithm is not yet effective.



Generate a result plot of the applied motor velocity (under **Results and Plots, Displacement/Velocity/Acceleration, Angular Velocity, Z Component**).



Initially, as the robot tilts backwards (positive pitch) the applied rotary motor angular velocity is -30 deg/sec. Shortly after, the robot gains balance and tilts forward. This is a sign to switch the motor velocity to +30 deg/sec. From that point onwards, the robot increases its forward tilt until it looses balance entirely, i.e. the rotary motor velocity is not able to counter the robot's forward tilt.

A possible solution could be to increase the magnitude of the rotary motor velocity to, let us suggest, 60 RPM.

40 Increase velocity.

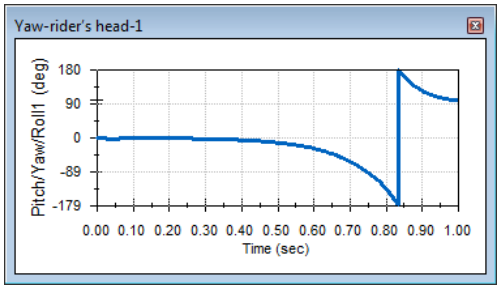
Edit both rotary motor features and increase their velocities to range from -60 to 60 RPM (-360 to 360 deg/sec). The modified expression is **IF({Pitch/Yaw/Roll1}:-360,0,360).**

41 Calculation.

Calculate the simulation for 1 second.

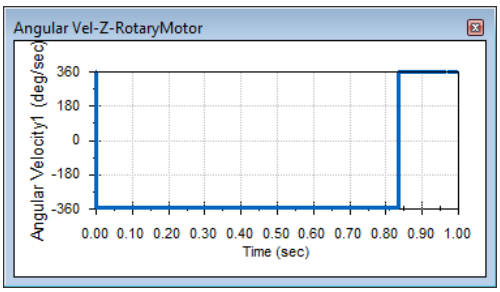
42 Tilt and motor velocity.

Review the Yaw plot for the `rider's head`.

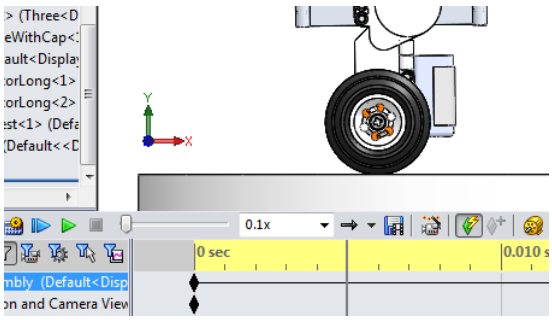


The robot still tilts forward. In fact, notice that it tilts forward even faster than when the rotation velocity of 5 RPM was used.

This result indicates that something is wrong with the control logic.



Zoom onto the beginning of the timeline and replay the animation from 0s to 0.01s by dragging the timebar.



Notice, that as the robot begins with the backwards rotation, the wheels move backward to counter this motion.

The reaction is very swift and the robot continues to tilt forward quickly. In fact, the larger the velocity of the wheels to counter the backward tilt, the faster the robot tilts forward.

Also notice, that when the rotational velocity changes sign, the wheels intermittently skid; this is not observed in the real situation. In reality, the wheels need to slow down to zero velocity and then accelerate to the maximum possible velocity in the positive direction.

The robot's control system does not control the velocity directly. Rather, it controls the power transmitted to the wheels. We will therefore improve the algorithm by controlling the applied torque, which is directly related to power.

43 Reaction torque.

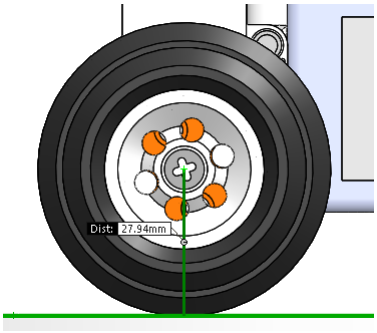
Plot the reaction torque required by the rotary motors (under **Results and Plots, Forces, Motor Torque, Z Component**).

Note: It is enough to plot torque at one rotary motor feature only.

The maximum torque is nearly 330 Newton-mm.

Maximum torque

The maximum torque generated by the motor can be computed from the observation, that the wheels do not slip on the `Tabletop` surface.



The maximum torque T_{MAX} transmitted to the ground can be computed from the following equation:

$$T_{MAX} = F_N \cdot \mu \cdot 27.94 = \frac{1}{2}(719.65x10^{-3} \cdot 9.81) \cdot 0.3 \cdot 27.94 = 29.58 Newton - mm$$

where F_N is the maximum normal force between one wheel and the ground (`Tabletop`), μ is the static friction coefficient and 27.94 is the distance between the ground (`Tabletop`) and the axle in millimeters (see the figure above). A value of 29.58 Newton-mm is the largest torque that the wheels are able to transmit to the ground under ideal conditions.

Let us try a similar control equation using the IF statement, with the maximum absolute value of torque equal to 29 Newton-mm.

44 Suppress rotary motors.

Suppress both rotary motor features.

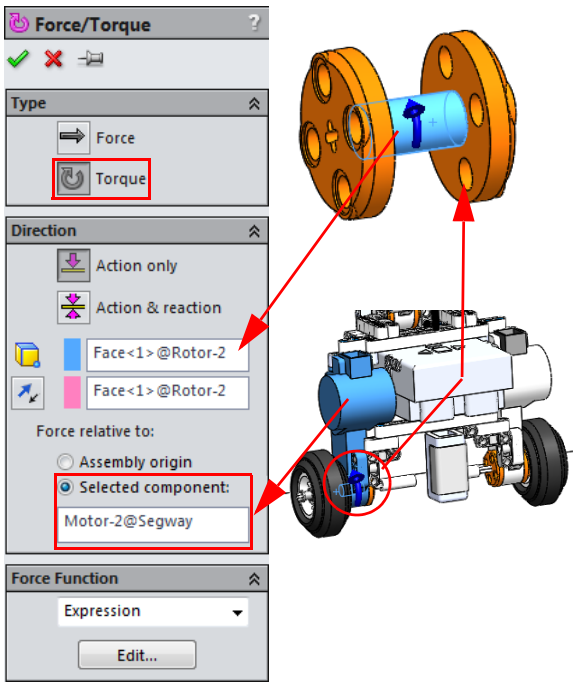
45 Input torque.

Isolate the `Rotor` part connected to `Motor` #2.

Apply **Action only Torque** to the cylindrical face of the `Rotor` part, as shown in the figure.

Under **Force Function**, select **Expression** and enter the following expression: **IF({Pitch/Yaw/Roll1}:-29,0,29)**.

Click the **Selected component** and select `Motor` #2 part as the component relative to which the torque is applied.



Apply an identical **Torque** to the `Motor` #1.

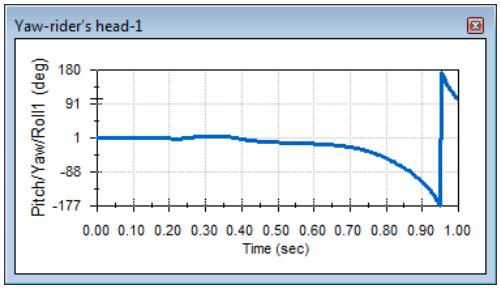
46 Calculation.

Calculate the simulation again for 1 second.

47 Tilt and wheel velocity.

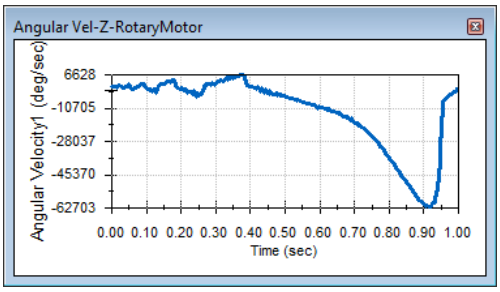
Review the Yaw plot for the `rider's head` and the wheel velocity.

The robot wobbles back and forth. Because the amplitudes are growing it eventually looses stability again.

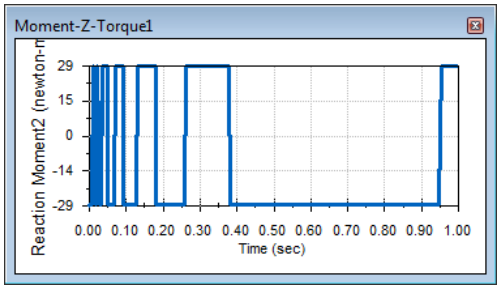


It can be observed that the rotational velocity now changes gradually, an important improvement when compared to the previous attempts.

Similarly to step **42**, visually analyze the motion of the wheels. It can be seen that as the robot tilts the wheels begin to slip again.



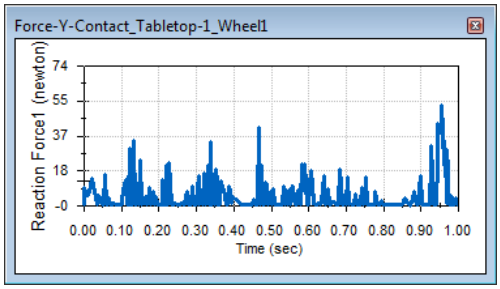
48 **Reaction torque.**



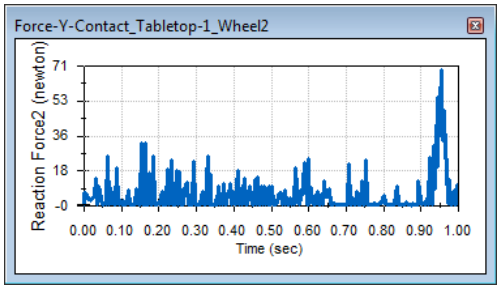
Under **Results and Plots, Forces, Reaction Moment, Z Component**, select the `Torque 1`.

Notice that as prescribed, the torque oscillates between positive and negative 29 Newton-mm. However, because the wheels are slipping, we need to improve the model to better reflect the real conditions. To understand what is happening at the tire/ground interface, plot the contact force.

49 **Contact force.**



Under **Results and Plots, Forces, Contact Force, Y Component**, select the contact faces of one of the tires and the `Tabletop` surface.



Plot the same contact force for the second tire/ground interface.

Notice the sharp peaks reaching as high as 70 N in magnitude. This goes against our assumption (Maximum torque on page 20), that the normal force exerted on the ground is equal to the weight of the robot.

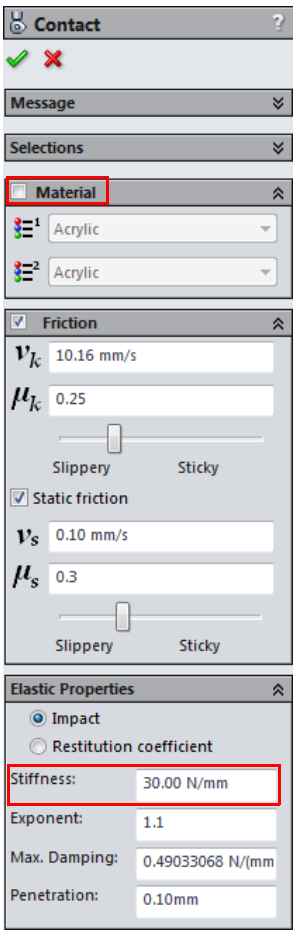
The sharp peaks signal that the robot's travel is uneven, exhibiting numerous intermittent jumps. Nothing similar is observed in the real situation. The reason for this behavior is inaccurate characterization of the contact between the tires and the ground, i.e. the contact stiffness is too high.

50 Contact modification.

Edit both contact conditions.

Uncheck the **Material** checkbox to open the contact properties for editing.

Under **Elastic Properties**, enter **30 N/mm** for the **Stiffness**.



Note: Unchecking the **Material** checkbox sets both fields to **Acrylic**, which may suggest that acrylic material is used instead of the original materials (rubber and steel). Notice, however, that these fields are grayed out and not used. The original numerical specifications for the Rubber (Dry) and Steel (Dry) contact are retained.

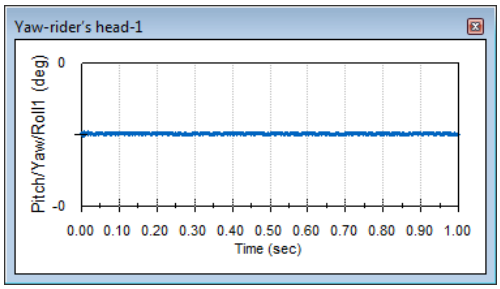
Note: The original contact stiffness of 2855.98 N/mm corresponds to the contact between solid steel and rubber objects. In our case, the contact is between the stiff `Tabletop` and the inflated tire. Because the robot tires are significantly less stiff than the surface of the `Tabletop`, we can assume that all contact stiffness arises from the tire part. It is then easy to estimate (by pressing the tire) that the stiffness is in the range of tens, rather than thousands of Newtons per millimeter. The new value of the contact stiffness, 30 N/mm, is our best guess which we feel is close to the real condition.

51 Calculation.

Calculate the simulation again for 1 second.

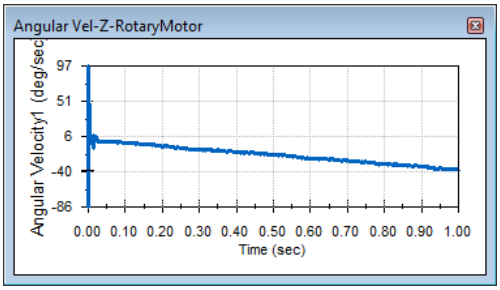
52 Tilt and wheel velocity.

Review the **Yaw** plot for the `rider's head`.



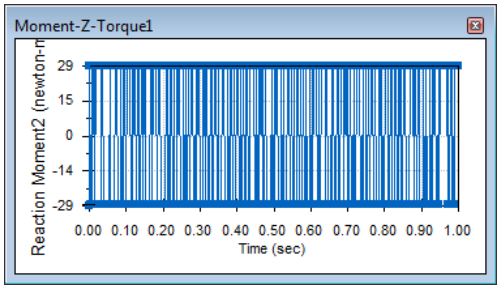
Notice how dramatically the results changed and how much closer they correspond to reality. The robot tilts forward and backward while trying to achieve and maintain its balance for significantly longer than in previous runs. Similar behavior was observed in the videos as well.

The rotational velocity trend still indicates that it increases, suggesting that the robot accelerates in the horizontal direction.



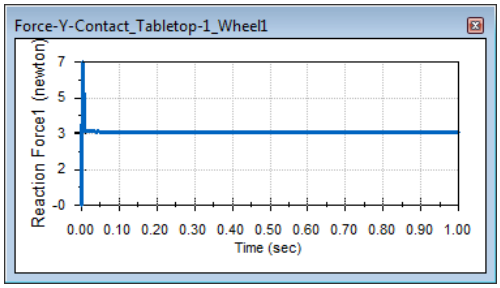
53 Reaction torque.

The torque magnitude oscillates from negative to positive 29 Newton-mm, just as was prescribed.

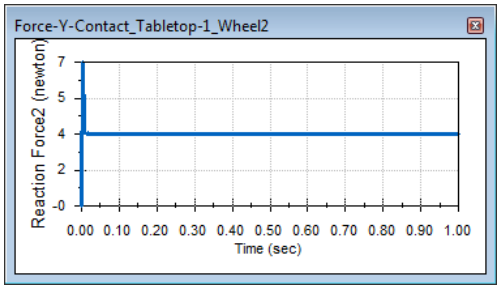


54 Contact force.

Both contact forces are nearly constant at close to 3 N and 4 N respectively. In sum (7N), this correlates very well with the weight of the robot 7.06 N $(719.65x10^{-3} \cdot 9.81)$.



Note: The two forces are not equal because the mass of the robot is not distributed symmetrically.



This result represents significant improvement, but still leaves space for improvement. If the control algorithm was to work properly, we would expect the rotational velocity (as well as the tilt) to attenuate. Because it appears that we have exhausted the capability of the current elementary control algorithm (single IF logical statement with constant input torque) a more advanced algorithm is necessary.

PID Control

PID control is one of the numerous control algorithms used in the industry and robotics. The PID stands for Proportional, Integral and Derivative control algorithm. To demonstrate this algorithm, consider the following.

Our goal is to achieve and maintain the robot's balance. The first approach was to eliminate tilt by means of applying external torque at the wheels. This, thus far, has proven not effective.

Consider the result plot of the tilt generated in step **52**. The robot reaches zero tilt configuration in each cycle, but fails to maintain it. At each zero tilt configuration, the robot reaches significant rotational velocity (pendulum swing) increasing the tilt on the opposite side. In fact, such behavior can be self-propelling and stable balance may not be found. In order to stabilize the robot at zero tilt configuration, we must also require that its rotational velocity is zero (to stop it from swinging past the stable configuration). This requirement may be enough, or we may go even further and require that not only rotational velocity but also rotational acceleration be zero.

The case where tilt and rotational velocity are controlled is referred to as PI algorithm (Proportional and Integral). The most advanced case when all three quantities are controlled is known as PID (Proportional, Integral and Derivative).

In the next section of this lesson we will implement a PID controller. To apply PID control, let us define the rotational velocity of the rider's head as error. Then, since the angular displacement (or tilt) is equal to the integral of the angular velocity, the tilt will represent the I-portion (or the integral portion) of the PID controller. Finally, because the angular acceleration is equal to the derivate of the angular velocity, the angular acceleration will represent the D-portion (or the derivative portion) of the PID controller. With the about definition of error, the control equation for the PID controller will be in the following form:

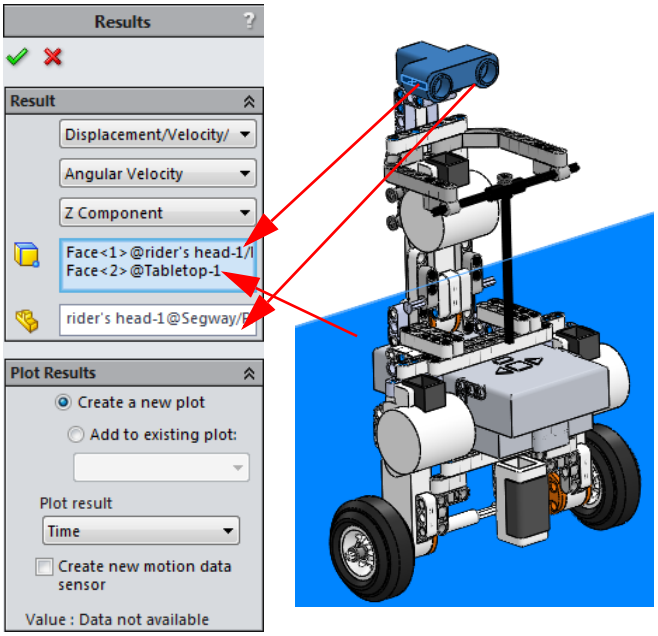
$$T = k_1 \cdot \phi + k_2 \cdot v + k_3 \cdot a$$

where T is the applied torque, ϕ is the tilt, v is the angular velocity of the `rider's head`, a is the angular acceleration of the `rider's head`, and multipliers $k_{1,2,3}$ are the proportionality constants, or gain coefficients. These constants represent our input for the control algorithm.

To implement PID control, we first have to plot the angular velocity and the angular acceleration of the `rider's head` (angular displacement, or tilt, was already plotted a few times in the previous steps).

55 Angular velocity of `rider's head`.

To plot angular velocity of the `rider's head` in its local coordinate system, select **Results and Plots, Displacement/ Velocity/Acceleration, Angular Velocity**, and **Z Component**.

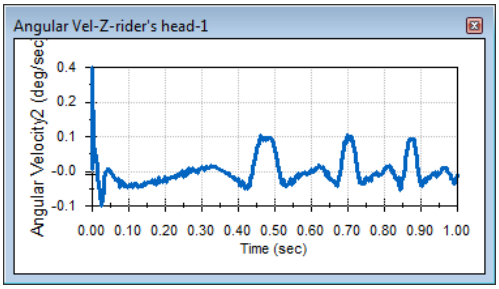


Select any face of the `rider's head` and any face of the `Tabletop` (exactly in this order) in the first field.

Then, to define the output coordinate system, select one more time any face of the `rider's head` in the second selection field.

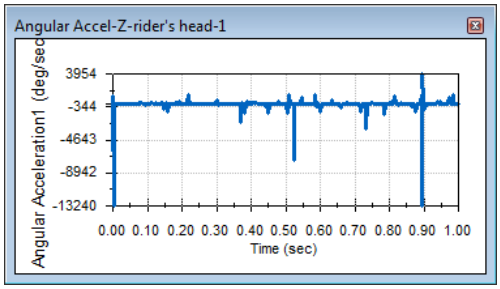
Click **OK**.

One can observe that the angular velocity of `rider's head` fluctuates between approximately -0.1 and 0.1 deg/sec.



56 Angular acceleration of rider's head

Follow the procedure in the previous step to plot the angular acceleration of the rider's head in its local coordinate system.



Tip: Review all three plots: tilt, angular velocity and angular acceleration. Do they all make sense to you? Does constant angular acceleration correspond to the linear change of angular velocity? Does the linear increase of angular velocity correspond to parabolic change in the angular displacement (tilt)? Does zero velocity correspond to nearly maximum acceleration? Try to discuss these points. Their understanding will help you to understand why it is beneficial to control all three quantities (PID control) rather than just the tilt.

PID Control of SegWay Robot

As was explained in the previous paragraphs, the PID control requires continuous adjustment of the input quantity (Torque) as a function of the error quantity (angular velocity), and its integral and derivative (tilt and angular acceleration). The control equation was shown in the PID Control on page 24. Now, consider the directions of the tilt, angular velocity, angular acceleration and the input torque to conclude that in order to balance the robot, the following holds:

- When tilt of the rider's head is positive, the input torque must be positive.
- When angular velocity of the rider's head is positive, the input torque must be negative.
- When angular acceleration of the rider's head is positive, the input torque must be negative.

With this in mind, the PID control equation will look as follow:

$$T = k_1 \cdot \phi - k_2 \cdot v - k_3 \cdot a$$

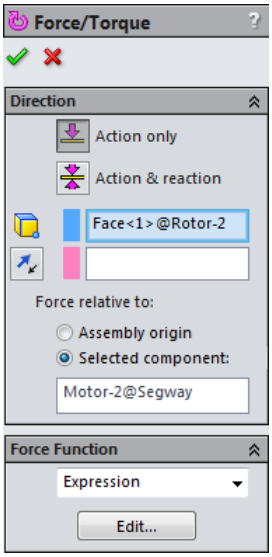
We will now edit the expression for the input torque to reflect the PID control equation above.

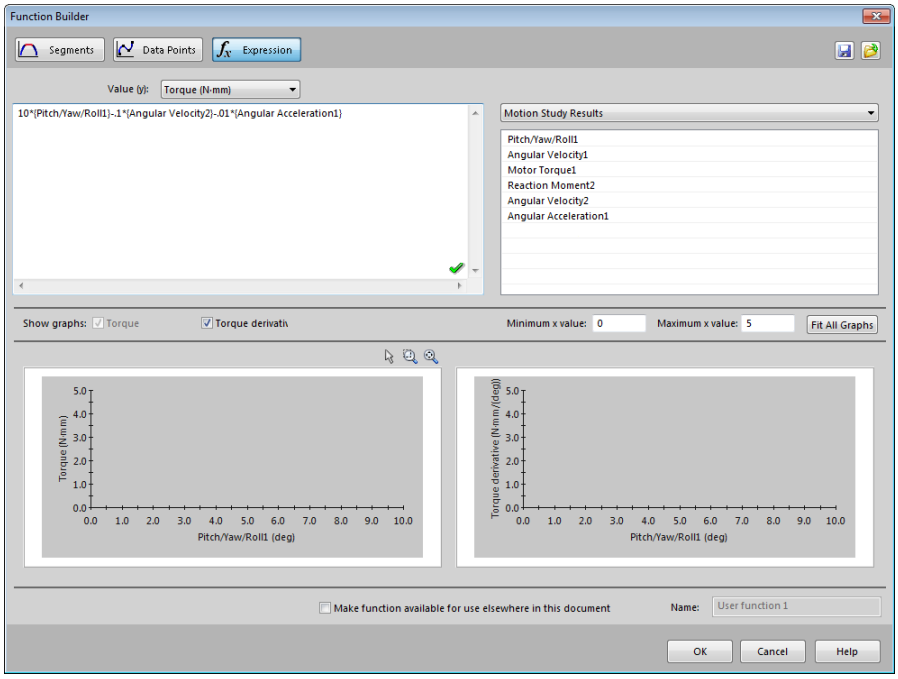
57 Edit input torque

Edit both of the two torque features.

Modify their expression to the following: **10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}**.

Note: The plot identifiers in your simulation may again have different numbers attached to them, i.e. {Angular Velocity1} or {Angular Acceleration3}.



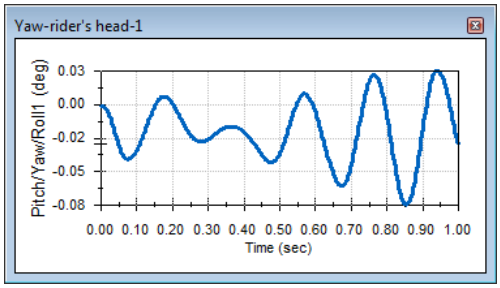


58 Calculation

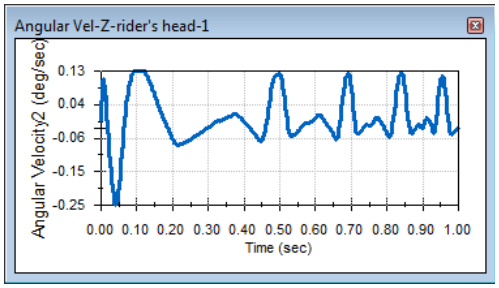
Calculate the simulation again for 1 second.

59 Results.

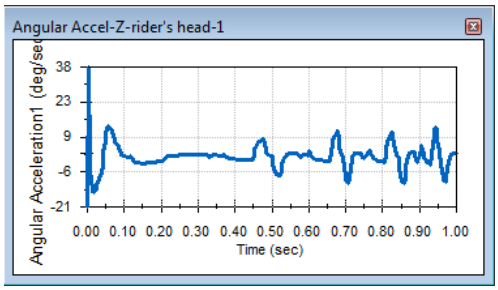
Review the plots for the tilt, angular velocity, angular acceleration of the `rider's head`, and of the applied torque.



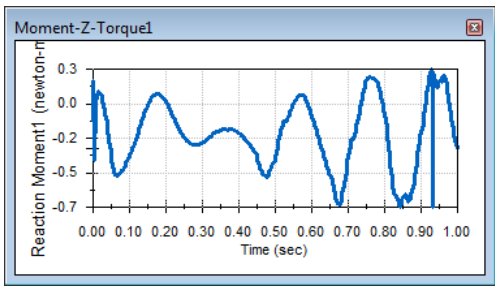
Tilt



Rotational velocity



Rotational acceleration



Applied torque

Reviewing the above plots and the animation of the robot motion suggests that the control is working pretty good. The problem is that it does not correspond too closely to the real observation. The robot does not tilt too much and the applied torque is very small (it is apparent from the videos that the maximum torque is applied occasionally). We can say that the controller is tuned too stiff, not fitting the current situation well.

In order for the control to work effectively and correspond to the observations, the system needs to be tuned.

When the controller is tuned, we expect the torque to occasional reach the maximum magnitude of 29 N.

The next section will set the limit of the applied torque to 29 N and tune the controller by adjusting the gain coefficients.

Maximum Torque Limit

The torque magnitude must be limited to the maximum which can be transmitted to the ground without the wheels slipping on the surface of the `Tabletop` (29.57 Newton-mm, Maximum torque on page 20). We can make use of the IF statement (page 17) and construct the following logical statement.

IF(PositiveLimit-Torque:PositiveLimit,PositiveLimit,(IF(NegativeLimit-Torque:Torque,NegativeLimit,NegativeLimit)))

In the statement above, Torque is the expression for the input torque from step **57**, **PositiveLimit=29 Newton-mm** and **NegativeLimit=-29 Newton-mm**. Take a moment and try to understand the above statement and how it works.

Combining the above complex IF statement with the expression for the torque (step **57**), and its limiting values (-29 and 29 Newton-mm) results in the following lengthy statement:

IF(29-(10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}):29,29,(IF(-29-(10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}):(10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}),-29,-29)))

The above expression uses the correct control expression for the input torque (step **57**) and limits its absolute magnitude to 29 Newton-mm. The next section elaborates somewhat on tuning the control expression.

Tuning PID Control

Tuning the digital PID control mechanism requires adjustment of the three gain constants until users are satisfied with the outcome. As you can expect, tuning three constants without any educated methodology can be a time consuming task. In this lesson, our goal is to tune the constants so that the robot gains balance and its behavior corresponds to the videos. The following coefficients will be used:

$k_1 = 10$

$k_2 = 0.1$

$k_3 = 0.01$

60 Edit input torque.

Modify the gain coefficients in the expression for both input torques as follows:

IF(29-(10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}):29,29,(IF(-29-(10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}):(10*{Pitch/Yaw/Roll1}-0.1*{Angular Velocity2}-0.01*{Angular Acceleration1}),-29,-29)))

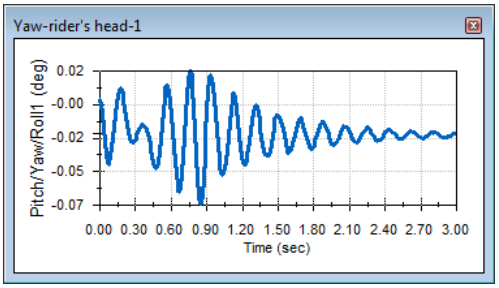
61 Calculation.

Calculate the simulation for **3s.**

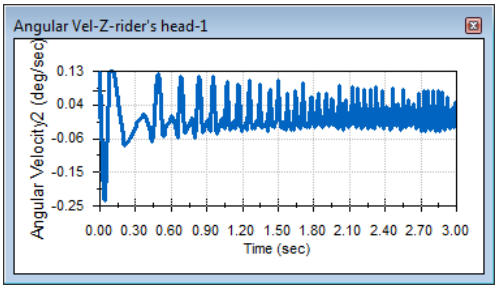
Note: The time calculation is increased in order to give the control equation enough time to reach balance.

62 Results.

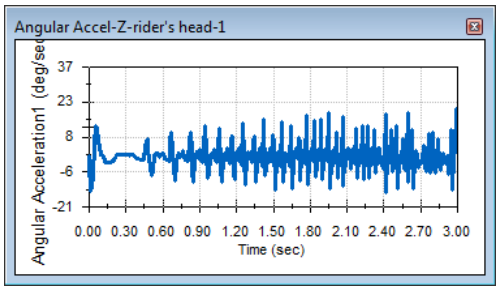
Review the plots for the tilt, angular velocity, angular acceleration of rider's head, and of the applied torque.



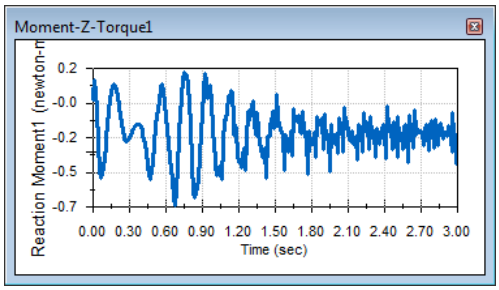
Tilt



Rotational velocity



Rotational acceleration



Applied torque

Notice that the control equation seems to work very well. All monitored quantities are slowly reduced to zero and the robot reaches balance. As the tilt, angular velocity, and angular acceleration decrease, so does the input torque.

It can be seen that the torque is not reaching its maximum value of 29 Newton-mm. To challenge the controller, we are going to add an initial short external force impulse tilting the robot forward, just as can be seen in the video when a person momentarily pushes the robot off balance.

To apply a short impulse force, we will make use of the STEP function.

STEP Function

A STEP function prescribes the given quantity (displacement, velocity, acceleration or force magnitude, for example) between two values with a smooth transition. Before and after the transition, the displacement, velocity or acceleration magnitude is constant.

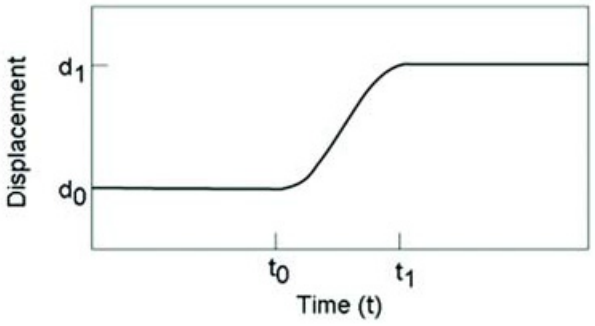
For example, consider the illustration at the right where:

d_0 = Initial value of displacement

d_1 = Final value of displacement

t_0 = Start step time

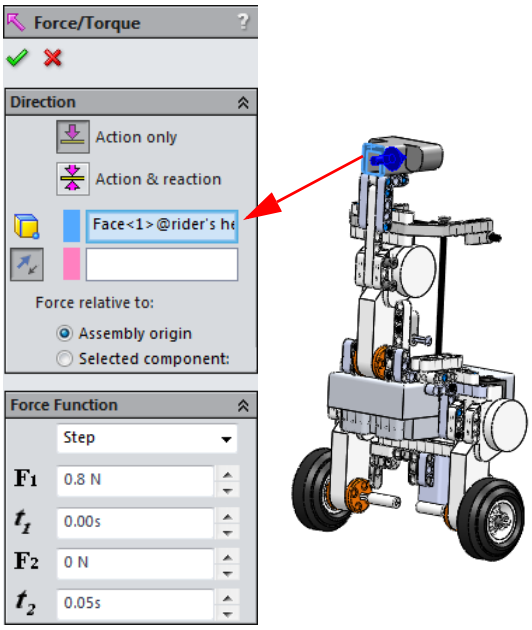
t_1 = Final step time



63 Force impulse

Apply **Action only Force** on the back face of the driver's head.

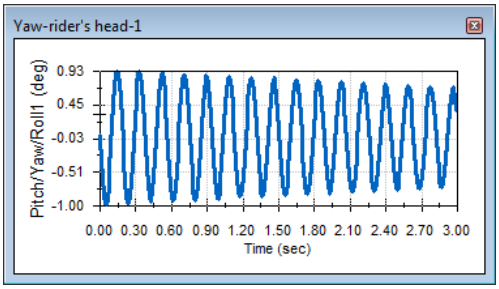
Enter **0.8 N** for **Initial Value**, **0s** for **Start Step Time**, **0 N** for **Final Value** and **0.05s** for **End Step Time**.



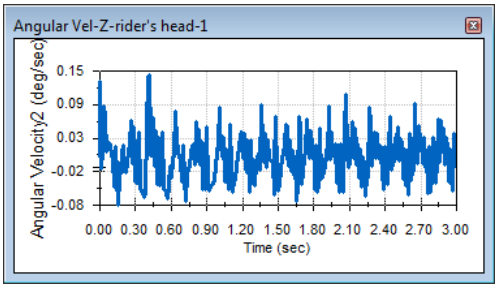
64 Calculation

Calculate the simulation again for **3s**.

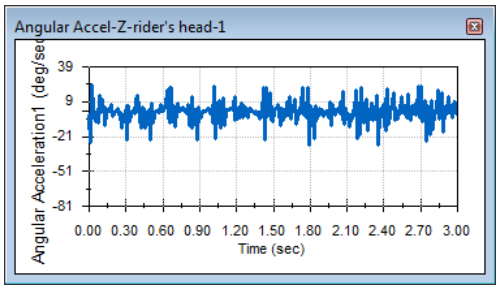
65 Results



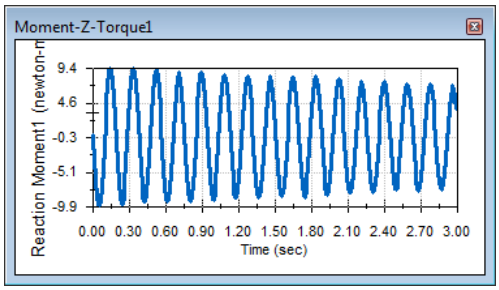
Tilt



Rotational velocity



Rotational acceleration



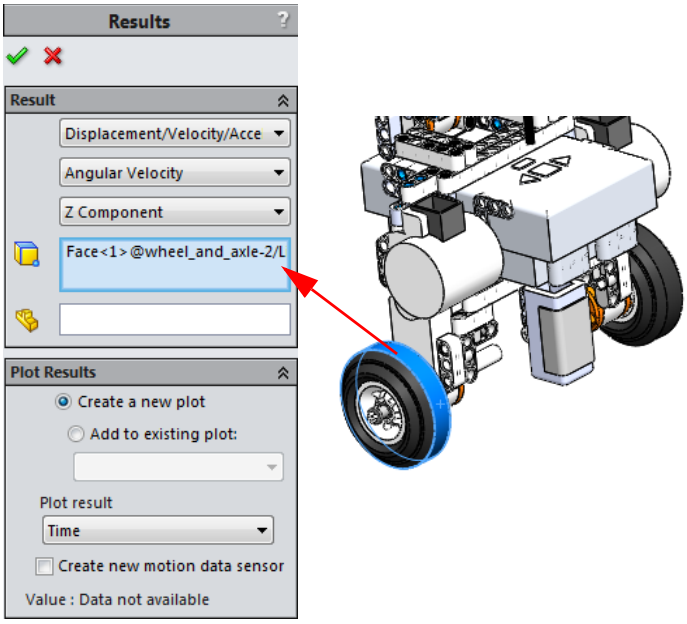
Applied torque

We can observe, that the tilt of the rider's head is controlled well, but some disturbances are seen in the plot of the rotational acceleration and the input torque (input torque is dependent on the acceleration).

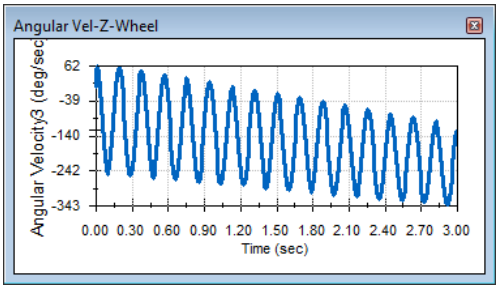
66 Angular velocity of wheel.

To plot angular velocity of one of the wheels, select **Results and Plots, Displacement/ Velocity/Acceleration, Angular Velocity, Z Component**.

Select the face of the tire indicated in the figure and click **OK**.



We observe that all of the quantities are controlled and decay in time.



Notice, that at the beginning, the maximum input torque of 29 Newton-mm is applied for a short moment.

The rotational velocity of the wheel is also continuous indicating that no slippage occurs. Also notice, that it stabilizes to a state where it decreases along a line. This indicates that while the robot and the rider experience near zero tilt, rotational velocity and rotational acceleration, the robot moves and accelerates in the horizontal direction. An analogy to this situation is a rider on the real SegWay transporter at a stable position, leaning slightly forward and accelerating. Further control to slow and speed up the horizontal motion of the robot would require a more advanced algorithm.

67 Disturbance force.

Increase the peak disturbance force from 0.8 N to **2.5 N**.

68 Calculation.

Calculate the simulation for **1s**.

69 Results.

Observe, that with this force magnitude the maximum torque and wheel traction cannot balance the robot any longer. The force became too large.

This phenomenon can also be observed on the video supplied along with the part files.