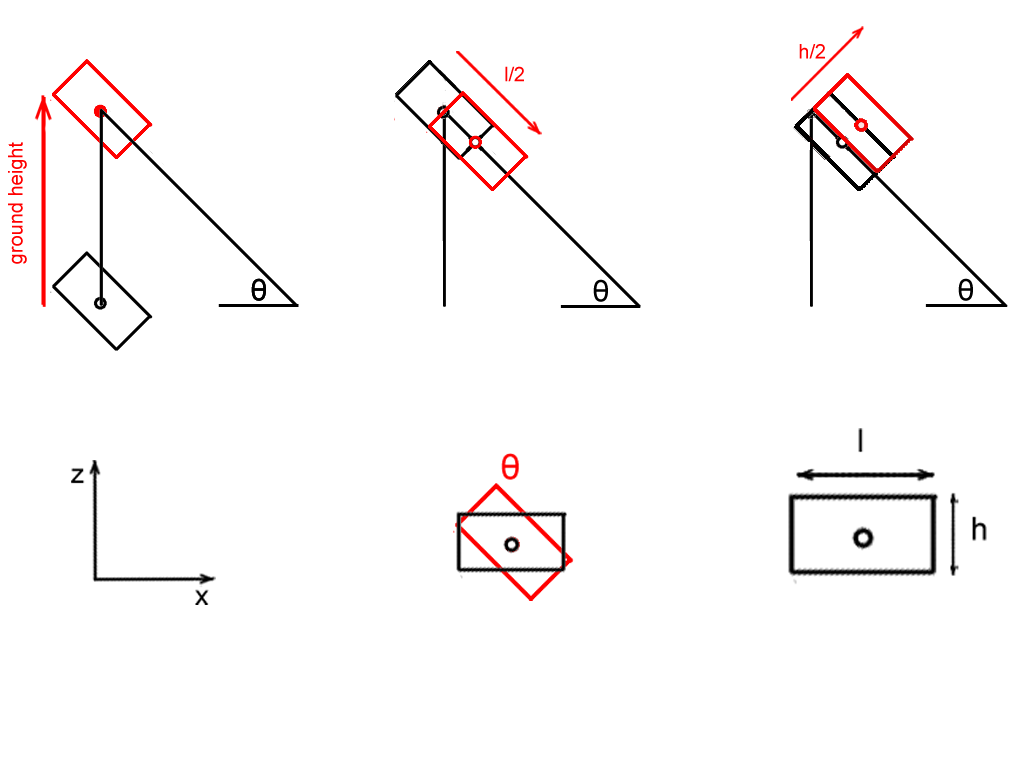
**Pile Driver in Simscape trials**

1. Edit the Simscape Multibody library Tread model
   1. Regarding the Body
      1. Replace the triangle link in Tread R & Tread L with **extruded solid** which is rectangle.
      2. Edit the Extr\_Data\_TriangleRounded\_Holes.m to make a new file Extr\_Data\_RectangleRounded\_Holes.m which plots a rectangle with rounded holes instead of triangle.
      3. The extruded solid implements the function Extr\_Data\_RectangleRounded\_Holes(L, Ri, Ro) which is in Extr\_Data\_RectangleRounded\_Holes.m 🡺successfully having the same body in simulation but with rectangular wheels.
      4. Connect the Upper Carriage and Bar mechanism to the new rectangular link and adjust it using transformations and tweaking parameters 🡺 worked somehow
   2. Regarding the Ground
      1. inclined ground
         1. Try changing the ground from a rectangle to a prism 🡺 didn’t work bec. The ground is a **brick solid**
         2. Making a new prism (**Extruded solid** - using four points method) and stick it to the old ground using a rigid transform. 🡺worked after many trials
         3. Try to make the vehicle go over the new ground(prism) in simscape 🡺 Unfortunately the vehicle fell it didn’t feel it
         4. Connect the new ground(prism) to the **Sphere to Plane Force** block, so the vehicle can feel the ground 🡺 the vehicle only felt it at the beginning of contact, then it fell.
         5. Visualize Contact Surfaces inside the **Visual** tab in the **Sphere to Plane Force** block to know the problem.
         6. The Problem is that the contact plane itself was not inclined, so after some edits (Rotational and translational Transformation and some dimensions manipulation) to make the plane inclined 🡺 The vehicle could finally move on the inclined plane

|  |  |  |  |
| --- | --- | --- | --- |
| Trials | Problems | Possible Solutions | Result |
| ‎1.2.1.4 | The **Sphere to Plane Force** block is designed to simulate forces with flat ground/plane only that’s why it failed. | * Create / edit **Sphere to Plane Force** block to feel inclined planes * Incline the plane of contact forces itself. | ✔ |
| ‎1.2.1.6 | At the very end the vehicle behaves a bit strange (the tread penetrates the ground) | * Introduce a new **Face to Plane Force** block to make contact force between the tread and the ground. | ✔ |
|  |  |  |  |

* + 1. inclined ground 2 (steeper ground + improved approach) [file 04a]
       1. Make a function for ground in file **right\_angle\_triangle.m** to plot a right angle triangle given the base length and angle in degrees and export the triangle’s dimensions to the workspace whenever the function is called.
       2. Use an **extruded solid block** to apply the function in Simulink and use **a rigid transform** (rotate 90 degrees about +X axis) because the **extruded solid block** always uses x y plane to draw on and extruded along the z axis, which differs than the Mechanics Explorer
       3. In **extruded solid block** put a frame on the face on which the vehicle moves, then rotate it using the exported triangle angle (ground\_angle) to adjust the orientation of the contact plane
       4. Adjust the length of the inclined plane to be equal to the hypotenuse of the triangle in the **Contact Sphere to Plane** blocks.
       5. Put the vehicle in the right place by editing the **Bushing Joint** block using the following functions:



* + - 1. Add rectangular solid to the triangle with new contact plane to simulate tip over 🡺TIP OVER achieved
    1. Adding Bumpers
       1. Tweaking the **Bumps** block inside the **Ground** block 🡺 it was managed to put two bumps on the inclined surface using translational transformation
    2. Adding Holes
       1. Try making a ground as a rectangle with holes and extrude it [refer to 04b]
          1. Try making it from scratch – make a new function

Put all points in n x 2 matrix and plot them 🡺 plot came out correctly but one line is connecting the rectangle to the hole.

Put the new function in the Simulink model using the extruded solid block 🡺 errors: a-The vertices specified by parameter Cross Section define a self-intersecting polygon. b- The ordering of the vertices specified by parameter Cross Section is invalid. The vertices should be ordered such that solid regions are to the left of each boundary segment and do not overlap. (even if it worked it will not create holes within the **Contact Sphere to Plane** block )

🡺 Drawing Fixed by making the lines connecting the rectangle with the circle vertical, 🡺however the holes where not seen in the **Contact Sphere to Plane** block‘s plane as expected.

* + - * 1. Using **Extr\_Data\_Box(10,5,10,5)** (square hole) 🡺 **Contact Sphere to Plane** didn’t see the hole
      1. Editing the **Sphere to Plane** library itself 🡺 a lot of hidden parameters (in the libraries) and I don’t know where their scripts are.
         1. Making local version and edit it

Editing **Check Sphere on Plane** block 🡺 even if u = 0 it have no effect.

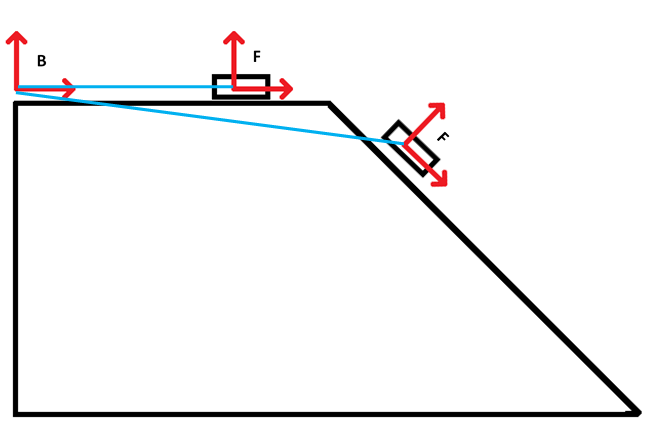
Editing **Calculate Penetration** block: a) change z by removing the deadlock block and introducing a –ve number 🡺 the vehicle had crazy response (fly opposite to the plane downwards). 🡺 b) change Vz didn’t have any effect.

Editing **Sensing** block 🡺 no effect

Understand the library more to know which block to edit – wait for the answer from github or mathworks website.

* + - 1. Try I-shaped ground to test the holes
         1. Either One whole Extrude 🡺 will not work because the **Contact Sphere to Plane** block‘s plane is always a rectangle
         2. Or stitching many **Contact Sphere to Plane** together
      2. Try the **sphere to cylinder contact** block approach
         1. Making two planes using two **Contact Sphere to Plane** blocks and leaving a space in the middle to simulate holes 🡺 the vehicle fell
         2. Putting two cones in the middle to act as holes using **two Sphere to Cone Force** blocks 🡺 better but the vehicle fell sometimes
         3. Put a plane to fill the space in the middle but lower than the two planes under the holes 🡺 worked very good for two holes 🡺 but unfortunately it didn’t work for making one hole , when one wheel is moving straight(not into the hole) the other wheel tends to do the same even if there is no plane under it.
         4. After some manipulations of some dimensions and slower speed 🡺 Left hole worked , however after width of 1 meter and speed of 1 it worked which I am not sure if it is good enough. It tipped over at speed of 5 Also I am doubting that the response on the front wheel is different than that of the back wheel. Also left hole was good while at the same time the right hole was not. (conclusion: Vehicle reacts differently to holes at different speeds and different dimensions)
         5. Connect **Extruded Solid** block to the hole(s) **contact force** blocks via the **rectangle\_with\_holes.m** script 🡺 done
         6. Trying the same with the inclined surface 🡺 if the inclined surface is steep enough (50 0)to make the vehicle go down further, this will increase the speed of the vehicle and if it is very fast it will pass the hole as if it doesn’t exist. However at lower angles (20 0) and low speeds the vehicle respond to plane under holes. (next step put hole cones, if it works(partially works) link it with the matlab script(might need a new script due to inclined surface))
         7. **Inclined slope with holes worked with the same drawbacks mentioned before, however the visualization of holes on the ramp couldn’t be done since the triangle is only drawn (extruded) from the side view.**
      3. Try using the **Face to Belt Faces Force** block instead
         1. Connecting **FaceF** to the ground plane and **PlaB** to the tread 🡺 error: ‘An implicit 6-DOF joint is attached to a degenerate mass distribution’.
         2. Try inverting the use of the **Box to Belt Force** Block by making the belt moving and change the box to the ground (fixed) 🡺 after many transformations and trials it worked fine for flat surface, also the Contact stiffness and damping were changed from 1e4 to 1e6 and from 1e2 to 1e5 respectively because the belt was “swimming” in the box before changing the values. I am not sure if it will work with inclined surface or not, also I am very sceptic that they would work about the holes.

1. Control vehicle to avoid tip over
   1. Detect when the vehicle tips over (Feedback)
      1. Use **transform sensor** block
         1. Define the frames to get the sensor data between them.
            1. **The base frame(B):** at the top left of the rectangular ground. **The second frame(F):** the centre of mass of the **solid** block in vehicle (temporary i.e must be changed to the center of mass of the entire vehicle (**maybe inertia sensor** block could be used)).



­­­­­­

* + - 1. Get data for the tipping over (e.g: Euler angles vs. angular velocity)
         1. Try to figure out when does it happen

Compare Euler angles vs. angular velocity

Vz must never be greater than zero.

* + - * 1. To get Euler angles:

Rotation matrix to get angles 🡪 need to solve 9 equations together ([3x3 matrix](https://lh3.googleusercontent.com/proxy/vTHyEtX7_Ph-AX9AqItgAFn-POz5y5XkCFnb8yHv7piqVG0jJ_5XpwJFoMfjuasgxFFeTWPep3PQqzfoNGDy_L0u93_fkdde))

Using angular velocity 🡪 integrate to get angles

From axis and angle 🡪 using the [dot product](https://ch.mathworks.com/matlabcentral/answers/151680-why-does-the-angle-measured-using-transform-sensor-block-measures-only-the-absolute-value-of-angle)

* + 1. Control the vehicle
       1. Simple on/off control (Bang-Bang)
          1. Sphere to plane case: Using Simulink switch block to change the wheels’ torque/velocity based on the roll angle (corresponding to the y –axis) of the vehicle, if no change in angle use positive velocity and if the angle changed use high negative velocity 🡪 doesn’t work at some moments (I believe this happens when the belt penetrates the ground) i.e. sometimes the front wheel reverse (CCW) and the rear wheel continue to move forward (CW) while have the same input (-ve constant) 🡪 but worked with 30 degrees ground inclination angle with torques 300N and -9000N for forward and backwards respectively. [refer to file 05b]
          2. Belt to Box case: Same as the previous point but with using two Belt to Box blocks 🡪 after many manipulations, transformations and assumptions it worked. [refer to file 06b and point ‎1.2.4.5.2]
       2. Fuzzy logic
          1. Belt to Box case:

Fuzzy logic approach was used using the assumption that the vehicle is moving forward and if it falls it will rotate on the y-axis. The fuzzy logic goes as follows: [please refer to file 06c]

* + - * + **Input variables:** Vin (driver’s input linear velocity) & angular velocity along the y axis (mimics gyroscope output)
        + **Output variables:** Vout the desired (control) linear velocity.
        + **Set of terms/linguistic descriptor:** NH, NM, NS, Z, PS, PM, PH

Where: **NH** is Negative High

**NM** is Negative Medium

**NS** is Negative Small

**Z** is Zero

**PS** is Positive Small

**PM** is Positive Medium

**PH** is Positive High.

N.B: For all variables the same however the ranges of values differ between the angular velocity and the linear velocities

* + - * + **The range of values:**

|  |  |  |
| --- | --- | --- |
| Membership fns | For Vin & Vout | For |
| NH | [-5 -5 -3.5] | [-5 -0.07 -0.045] |
| NM | [-4.5 -3.5 -2] | [-0.05 -0.04 -0.035] |
| NS | [-3 -1.25 -0.5] | [-0.04 -0.03 -0.02] |
| Z | [-1 0 1] | [-0.025 0 0.025] |
| PS | [0.5 1.75 3] | [0.02 0.03 0.04] |
| PM | [2 3.5 4.5] | [0.035 0.04 0.05] |
| PH | [3.5 5 5] | [0.045 0.07 5] |
| Range | [-5 5] [link](http://www.roadsky.org/product/ycr-solar-pile-driver/) | [-0.2 0.2] |

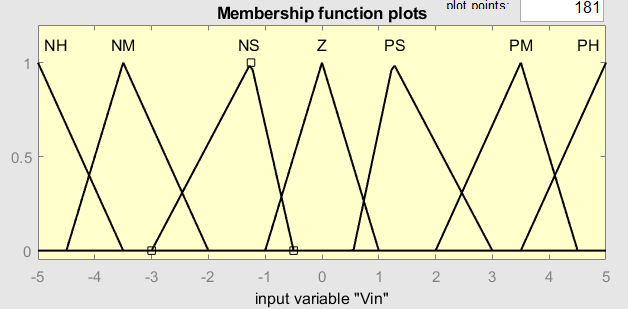
* + - * + N.B: The three values are of plotting a triangle in a graph so they correspond to: [left\_point height\_value right\_point]

These values were obtained using some experimentation (trial & error)

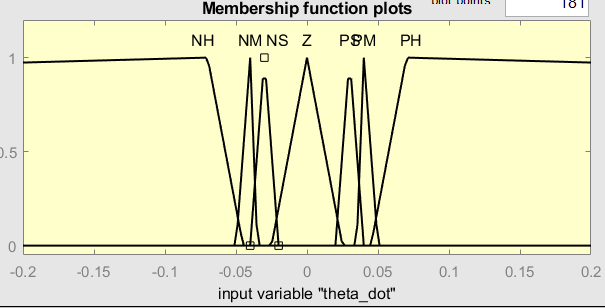
Negative and positive here only indicates the direction of motion or rotation. Positive motion is translation to the right and rotation along +ve y-axis (clockwise in video). Contrarily Negative motion is translation to the left and rotation along -ve y-axis (anti-clockwise in video).

* + - * + Graphs:

For Vin & Vout :



For :



* + - * + Rules/Rule base (until now):

|  |  |
| --- | --- |
|  |  |
| Vin |  | Z | PS | PM | PH |
|  | Z | Z | PS | PM | PH |
|  | PS | x | NS | x | x |
|  | PM | x | x | NM | x |
|  | PH | NH | NH | NH | NH |

Where the values of Vin are highlighted in blue while those of are marked in orange. The white values are of Vout.

All of these rules apply the AND operation.

For e.g.: cell 4x4 is explained as: if Vin is PM AND is PM, then Vout ­is NM (in grey)

N.B: the values x denotes don’t care or not needed to be handled.

* + - * + Inferencing and de-fuzzification are done using the mamdani method.

N.B: fuzzy logic toolbox to implement fuzzy logic in Simulink

The fuzzy logic is expanded to include the negative motion [please refer to file 06d], so the Rule base is extended as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| Vin |  | Z | PS | PM | PH |  | NS | NM | NH |
| 1 | Z | Z | PS | PM | PH | Z | NS | NM | NH |
| 2 | PS | x | NS | x | NH | PS | PS | x | PH |
| 3 | PM | x | x | NM | x | PM | x | PM | x |
| 4 | PH | PH/NH | NH | | | PH | PH | | |
| 5 | NS | x | NS | x | NH | NS | PS | x | PH |
| 6 | NM | x | x | NM | x | NM | x | PM | x |
| 7 | NH | PH/NH | NH | | | NH | PH | | |

Legend:

|  |  |
| --- | --- |
|  | possible values |
|  | Vin possible values |

Cases for Vout­­ values are classified as:

|  |  |
| --- | --- |
| -------- | Right translation motion |
| -------- | Left translation motion |
|  | Motion on downwardly inclined plane |
|  | Motion on upwardly inclined plane case |

As we can see in the 4th and the 7th rows we have a conflicting values for e.g. if = PH it will give different values to V­out ­, according to the direction of translation . That’s why we must define more 12 rules (PH w/ PS,PM,PH | NH w/ PS,PM,PH | PH w/ NS, NM, NH | NH w/ NS, NM, NH) to handle these cases , however these rules can be reduced to only 4 , if a new binary variable (dir) for the direction of translation (positive P or Negative N) is added to the fuzzy logic which can be depicted as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | P | | |  | N | | |
| Vin |  | Z | PS | PM | PH |  | NS | NM | NH |
| 1 | Z | Z | PS | PM | PH | Z | NS | NM | NH |
| 2 | PS | x | NS | x | NH | PS | PS | x | PH |
| 3 | PM | x | x | NM | x | PM | x | PM | x |
| 4 | PH | PH/NH | NH | | | PH | PH | | |
| 5 | NS | x | NS | x | NH | NS | PS | x | PH |
| 6 | NM | x | x | NM | x | NM | x | PM | x |
| 7 | NH | PH/NH | NH | | | NH | PH | | |

|  |  |
| --- | --- |
|  | possible values |

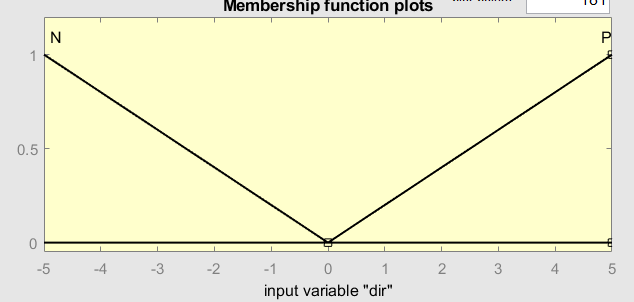
The 4 rules would be:

If dir = P AND , then Vout is NH

If dir = P AND , then Vout is NH

If dir = N AND , then Vout is PH

If dir = N AND , then Vout is PH

For dir: P = [0,5,5] and N = [-5,-5,0]

* + - 1. Model based

|  |  |  |
| --- | --- | --- |
| Trials | Problems/issues | Possible Solutions |
| ‎2.1.2.1.1 | Doesn’t work at crucial moments (most importantly at the moment of impact) | Try with the belt plane contact again [‎1.2.4.5.2]  Maybe when the belt has contact with the ground plane this effect will stop 🡪 check 2.1.2.1.2 |
| 2.1.2.1.2 | Works but with too many assumptions and some hard coding. Adv. Works at higher inclination (45 degrees) |  |
| 2.1.2.2.1 | * Works but I am not sure if I need to extend the rules to handle redundant cases or not. * Needs better stopping condition |  |

1. Tracks for the Vehicle
   1. Subcomponent Approach

This approach is based on creating a small component (sub-component) to be a part of the track of the vehicle, where multiple sub-components form a chain-like shape/structure which will eventually be rotated by a wheel or sprocket

* + 1. Creating the Sub-component
       1. Using a cylinder with two spheres as wheels
          1. Spheres/wheels actuated

This method uses two spheres connected to a cylinder (one from the right and the other from the left) similar to a simple differential drive robot to drive the motion of the whole body, where the input torque is given to the wheels, when the wheels rotate the cylinder moves along with them with the same velocity. After some trials this model worked when the 2 wheeled robot [example](https://ch.mathworks.com/matlabcentral/answers/388991-simulation-of-a-rolling-wheel-on-a-flat-surface-using-simmechanics-contact-forces-library-block) in the [Simscape Multibody Contact Forces Library](https://de.mathworks.com/matlabcentral/fileexchange/47417-simscape-multibody-contact-forces-library) library was used. [01\_Cylinder\_with\_spheres>01>01]

* + - * 1. Cylinder actuated

Using the previous model[3.1.1.1.1] , this model was made such that the cylinder is the part which is rotated then the wheels just follow its movement. [01\_Cylinder\_with\_spheres>02>01] N.B: damping changed to make cylinder move on ground

* + - 1. Roller
         1. This method intended to use the cylinder itself without wheels as a roller (rolling on the ground) trials:

Try to make only one joint (**Bushing joint**) between the cylinder and the ground (didn’t work due to a degenerate **Bushing joint** mass distribution error)

Using **sphere to plane** block, a **bushing joint** and a **Revolute Joint.** The **sphere to plane** block didn’t make any difference. The cyldiner is rotating w.r.t the ground however it can’t roll. [02\_Roller>02>01]

This trial is similar to ‎[3.1.1] , but this time the sphere(s) is(are) inside the cylinder (hidden):

One Sphere:

One sphere in the middle of the cylinder is used along with the **sphere to plane** block to make the cylinder roll on the ground when actuating the sphere. [02\_Roller>03>01>01]

Multiple Spheres:

four spheres were installed along the cylinder’s rotation axis are inside the cylinder to it roll by actuating all of them [02\_Roller>03>02>01] issue: cylinder direction differs (not always moving in a straight line)

* + - 1. Rectangle



Inspired by this photo the motivation is to create a similar subcomponent (marked in red) where in contrast to the cylinder or spheres the contact between the rectangle and the ground is a flat one not rolling/rotation.

* + - * 1. Here the rectangle is connected to the ground using a **Bushing joint** and **a face to plane contact** block. [03\_Rectangle>01]
  1. Using Chain Drive

There already exists a chain-like structure but it is used to transfer power not for movement. Here the possibility of exploiting such a readymade structure to be used as tracks for the vehicle is explored.

|  |  |  |
| --- | --- | --- |
| Trials | Problems | Solutions |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Trials | Problems | Solutions |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Trials | Problems | Solutions |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Trials | Problems | Solutions |
|  |  |  |
|  |  |  |

# Important links:

**## Simscape Multibody library functions: https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/e58b0cc7-4a80-11e4-9553-005056977bd0/ce916f87-75e0-4e54-9605-6346ebe593b1/previews/Simscape\_Multibody\_Parts\_Lib\_R18a/Scripts\_Data/Doc/html/Multibody\_Parts\_Library\_Overview.html**