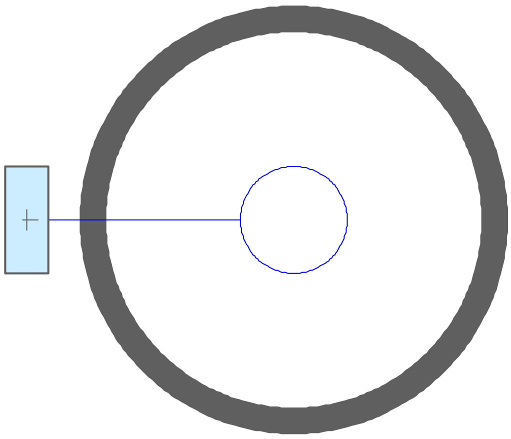
Dr. Dirk Zimmer

Virtual Physics

07.02.2021

Exercise 7: 2D-Mechanics: Ideal rolling wheel
Solution

Task A: Develop a component for an ideal wheel joint.



Since this component has only one connector with 3 effort-flow pairs, we need 3 equations to describe the dynamic behavior. But first, let us declare a few auxiliary variables with their corresponding equations:

```
phi = frame_a.phi;
w = der(phi);
z = der(w);
vx = der(frame_a.x);
```

Now, we can add the three missing equations that describe the physical behavior:

```
//holonomic constraint equation for the position of the wheel (It cannot move vertically)
frame_a.y = R;
//non-holonomic constraint equation for the ideal rolling (horizontal movement)
vx = w*R;
//the balance of force and torque
frame_a.fx*R = -frame_a.t;
```

Modelling the initialization is a little more tricky, since due to non-holonomic constraint we have more states at the level of position (x and phi) than at the level of velocity (vx or w). Hence we need 3 equations for a full initialization.

```
if initialize then
  phi = phi_start;
  w = w_start;
  frame_a.x = x_start;
end if;
```

For the visualization, a disc and two rods have been chosen. Using MB.Frames.planarRotation enables us to avoid the computation of the rotation by hand.

Final Remark: Since this component is rolling just in one-dimension, it is possible to replace the non-holonomic constraint by a holonimc one:

//~~non-~~holonomic constraint equation for the ideal rolling
~~vx = w*R;~~ frame_a.x = phi*R;

For the full solution, see PlanarMechanicsV3.mo

Task B: Test your component

See: PlanarMechanicsV3.mo (Examples.WheelBasedCraneCrab)

Task C: Model a rigid wheel with dry friction

First, we have to add the required parameters of the dry-friction model:

```
parameter SI.Velocity vAdhesion "adhesion velocity";
parameter SI.Velocity vSlide "sliding velocity";
parameter Real mu_A "friction coefficient at adhesion";
parameter Real mu_S "friction coefficient at sliding";
```

Then we have to replace the non-holonomic constraint equation by the friction law. To this end, we replace one equation by three equations and two additional variables (v_slip, N)

```
v_slip = vx - w*R;
N = -frame_a.fy;
frame_a.fx = N*noEvent(Utilities.TripleS_Func(vAdhesion,vSlide,mu_A,mu_S,v_slip));
```

The normal force is represented by -frame_a.fy and results from the holonomic constraint equation. The normal force becomes negative when the wheel is torn off the ground (or if gravity would point upwards…) The slip velocity v_slip represents a reformulation of the former non-holonomic constraint. The dry-friction law has been used before many times.

Since the non-holonomic constraint equation has been removed there are now 4 potential states to be initialized:

```
//Initialization of Position and Velocity
if initialize then
  phi = phi_start;
  w = w_start;
  frame_a.x = x_start;
  vx = vx_start;
end if;
```

For the complete solution and an application example see PlanarMechanicsV3.mo (Examples.CounterSpin)

Task D: Apply Pantelides Algorithm

Transform the following system of differential-algebraic system into state-space form.

```
dx/dt = 5 * z * b

dy/dt = a

2 * dz/dt = b

b = y * x

y = 1 - x

a = c - d

d/2 = b
```

Causalize each equation and transform the set of equations into a sequence of assignments. You may differentiate equations if necessary.

First, let us identify the potential states by looking at the time-derivatives:

These are `x`, `y` and `z`. We can hence assume them to be known. There are no further inputs specified and time itself does not occur. Hence these are also the only a-priori knowns.

We start with forward casualization and look for the equation with the least unknows. This is

```
y = 1 - x
```

with 0 unknowns. It does represent a constraint between the two states. We choose to remove `y` from the set of states and it becomes unknown. Also we add the time derivative of the constraint to the set of equations: `dy/dt = -dx/dt`

We restart forward causalization by iteratively looking for the equation with the least amount of unknowns:

```
y := 1 - x

b := y * x

d := 2*b

dz/dt := b/2

dx/dt := 5 * z * b

dy/dt := -dx/dt

a := dy/dt

c := a + d
```

This is it. At each iteration there was at least one equation with exactly 1 unknown but no residual equation with 0 unknowns. Everything can simply be causalized.