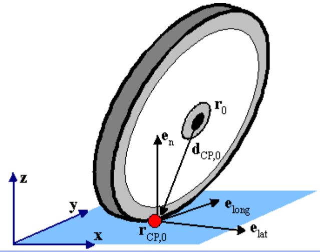
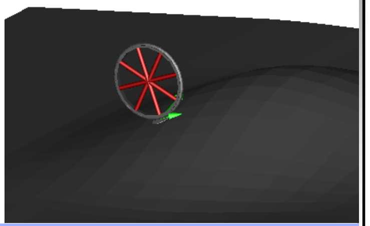
Virtual Physics
Equation-Based Modeling

TUM, November 18, 2014

Wheels and Tires: Realization in Planar Mechanics





Dr. Dirk Zimmer

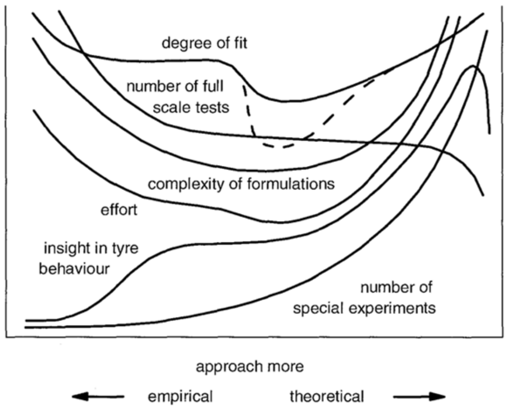
German Aerospace Center (DLR), Robotics and Mechatronics Centre

Outline

In this lecture, we are going we study the design of semi-empirical wheel models and their implementation in Modelica.

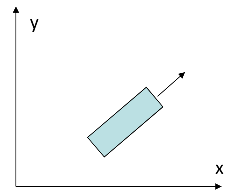
- Motivation behind semi-empirical models
- Stepwise modeling approach: Wheel and tyre models
  - **Level 1:** ideally rolling wheel
  - **Level 2:** slick-tyre wheel (Dry-Friction)
  - **Level 3:** tread-tyre wheel (Slip-Based Characteristic)
- Here, we model only in planar mechanics

Motivation



Wheels

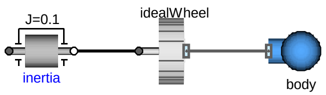
- In our planar-mechanical world, the wheel shall roll on the whole xy-plane



- The angle phi describes the orientation (driving direction) of the wheel.
- The wheel rotation around the axis is described by an extra rotational flange.
- The wheel cannot tilt. It is always in upright position. So the third angle is neglected.

Wheels

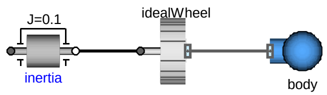
- The actual wheel can be decomposed into three components:



- A one-dimensional inertia that models the inertia of the wheel around the wheel axis.
- A two dimensional body-component that models the mass and inertia with respect to the planar domain.
- A “wheel joint” that implements the non-holonomic constraints of motion.
- Only the wheel joint needs to be modeled.

Wheels

- The actual wheel can be decomposed into three components:

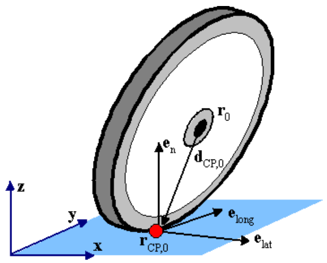


- The wheel joint establishes non-holonomic constraints on the level of velocity.
  - The lateral velocity is zero
  - The longitudinal velocity is proportional to the wheels rotation so that the velocity of the virtual contact point is zero.

Level 1: Ideal rolling

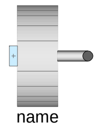
Fundamental assumptions

- The wheel is treated as a freely moving body.
- The fundamental equations of motion apply.
- The contact-forces result out of the constraint equations.



Ideal Rolling Wheel

Let us model a simple version of the wheel joint.

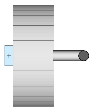


- Let us assume that the driving direction is the x-axis and that the orientation phi is fixed to 0°.

```
model IdealWheelJoint
  Interfaces.Frame_a frame_a;
  Rotational.Interfaces.Flange_a flange_a;
  parameter SI.Length radius;

  SI.AngularVelocity w_roll;
  SI.Velocity v[2], v_long;
  SI.Force f_long;

equation

  v = der({frame_a.x, frame_a.y});
  w_roll = der(flange_a.phi);

  v_long = radius*w_roll;

  v_long = v[1];
  v[2] = 0;

  -f_long*R = flange_a.tau;
  frame_a.phi = 0;
  frame_a.fx= f_long;

end IdealWheelJoint;
```

Ideal Rolling Wheel

Let us model a simple version of the wheel joint.

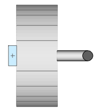


- Retrieving the velocities
- Projecting the driving velocity
- Non-holonomic constraints
- Transmission of force

```
model IdealWheelJoint
  Interfaces.Frame_a frame_a;
  Rotational.Interfaces.Flange_a flange_a;
  parameter SI.Length radius;

  SI.AngularVelocity w_roll;
  SI.Velocity v[2], v_long;
  SI.Force f_long;

equation

  v = der({frame_a.x, frame_a.y});
  w_roll = der(flange_a.phi);

  v_long = radius*w_roll;

  v_long = v[1];
  v[2] = 0;

  -f_long*R = flange_a.tau;
  frame_a.phi = 0;
  frame_a.fx= f_long;

end IdealWheelJoint;
```


Ideal Rolling Wheel

Let us model a simple version of the wheel joint.

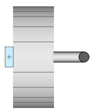


- Now let us parameterize the driving direction by sx and sy
- We project the velocity from 1D into 2D
- We project the force from 2D into 1D.

```
model IdealWheelJoint
  Interfaces.Frame_a frame_a;
  Rotational.Interfaces.Flange_a flange_a;
  parameter SI.Length radius;
  parameter SI.Length r[2];
  final parameter SI.Length l = sqrt(r*r);
  final parameter Real e[2] =  r/l;
  SI.AngularVelocity w_roll;
  SI.Velocity v[2], v_long;
  SI.Force f_long;

equation
  R = {{cos(frame_a.phi), sin(frame_a.phi)},
      {-sin(frame_a.phi),cos(frame_a.phi)}};
  e0 = R*e;

  v = der({frame_a.x,frame_a.y});
  v = v_long*e0;
  w_roll = der(flange_a.phi);
  v_long = radius*w_roll;
  -f_long*radius = flange_a.tau;
  frame_a.t = 0;
  {frame_a.fx, frame_a.fy}*e0 = f_long;
end IdealWheelJoint;
```


Ideal Rolling Wheel

Let us model a simple version of the wheel joint.

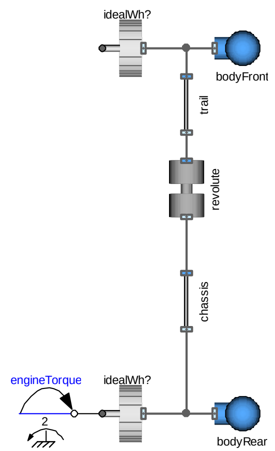


- Now we remove the holonomic constraint on the angle.
- We know this procedure from the prismatic joint.

```
model IdealWheelJoint
  Interfaces.Frame_a frame_a;
  Rotational.Interfaces.Flange_a flange_a;
  parameter SI.Length radius;
  parameter SI.Length r[2];
  final parameter SI.Length l = sqrt(r*r);
  final parameter Real e[2] =  r/l;
  SI.AngularVelocity w_roll;
  SI.Velocity v[2], v_long;
  SI.Force f_long;

equation
  R = {{cos(frame_a.phi), sin(frame_a.phi)},
      {-sin(frame_a.phi),cos(frame_a.phi)}};
  e0 = R*e;

  v = der({frame_a.x,frame_a.y});
  v = v_long*e0;
  w_roll = der(flange_a.phi);
  v_long = radius*w_roll;
  -f_long*radius = flange_a.tau;
  frame_a.t = 0;
  {frame_a.fx, frame_a.fy}*e0 = f_long;
end IdealWheelJoint;
```

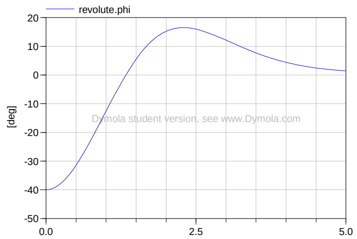

Single-Track Model

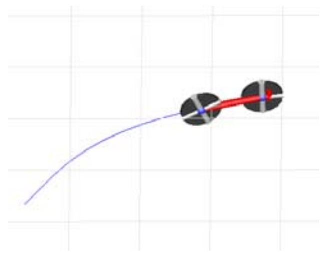
- We can use the wheel joints to construct a single-track model of a vehicle.
- This model has simply two masses: One representing the rear frame and one representing the front part.
- The wheels have no separate inertia.



Single Track Model: Results

Robotics and Mechatronics Centre

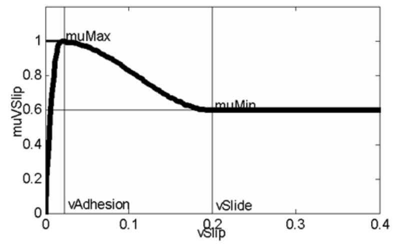




Level 2: Wheel with Dry Friction

Robotics and Mechatronics Centre

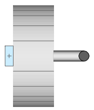
- The model of a rigid wheel resembles roughly a train-wheel.
- We maintain the holonomic constraint: The wheel is bounded to the track-plane (that is anyway the case in planar mechanics)
- The two non-holonomic constraints are released: slippage is allowed.
- The contact forces become now a function of the slip-velocity:



Wheel with Dry Friction

Robotics and Mechatronics Centre

Now let us implement a rigid wheel with the dry-friction law:



name

Let us determine the parameters:

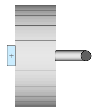
- Coefficients for stiction and friction (common for lateral and longitudinal direction)
- Normal Force
- Adhesive velocity, Sliding Velocity (for regularization purposes)

```
model IdealWheelJoint

  parameter SI.Force N;

  parameter SI.Velocity vAdhesion;
  parameter SI.Velocity vSlide;

  parameter Real mu_A ;
  parameter Real mu_S;


 […]


equation
  […]

end IdealWheelJoint;
```


Wheel with Dry Friction

Robotics and Mechatronics Centre

Now let us implement a rigid wheel with the dry-friction law:



name

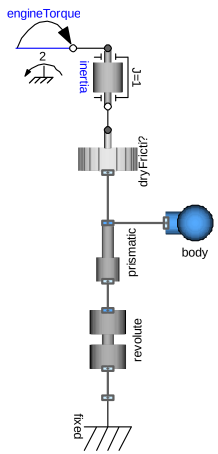
1. First, we determine the longitudinal and lateral velocities
2. Then we compute the slip velocities
3. Given the slip-velocities, we can compute the force
4. This projected on the frame-forces

```
model IdealWheelJoint
  […]

equation
  v_long = v*e0;
  v_lat = -v[1]*e0[2] + v[2]*e0[1];

  v_slip_lat = v_lat - 0;
  v_slip_long = v_long - radius*w_roll;
  v_slip = sqrt(v_slip_long^2 +
             v_slip_lat^2)+0.0001;

  -f_long*R = flange_a.tau;
  frame_a.t = 0;
  f = N*TripleS_Func(vAdhesion,
               vSlide,mu_A,mu_S,v_slip);
  f_long =f*v_slip_long/v_slip;
  f_lat  =f*v_slip_lat/v_slip;

  f_long = {frame_a.fx,frame_a.fy}*e0;
  f_lat = {frame_a.fy,-frame_a.fx}*e0;

  […]
end IdealWheelJoint;
```

Dry Friction: Test Model

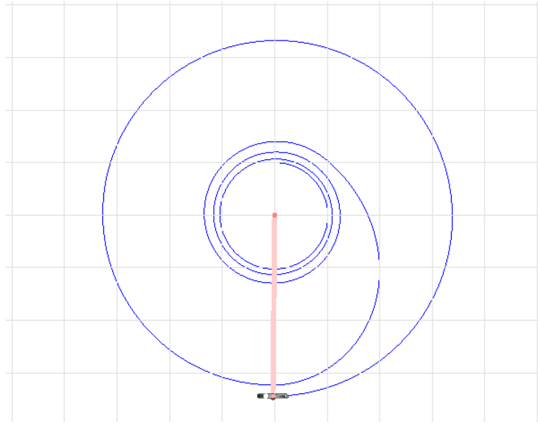
Robotics and Mechatronics Centre

- In order to test our dry-friction wheel model, let us build the following virtual test rig.
- The wheel is forced on a circular path by a mechanic construction.
- The ideal wheel would turn on a circle with constant radius in ever increasing speed.
- What does the wheel with the dry-friction model?



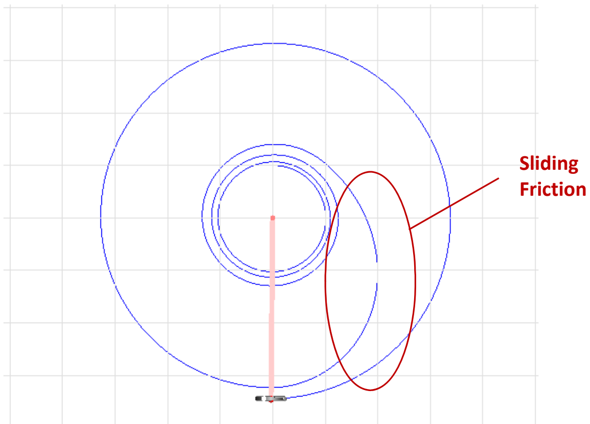
Dry Friction: Trajectory

Robotics and Mechatronics Centre



Dry Friction: Trajectory

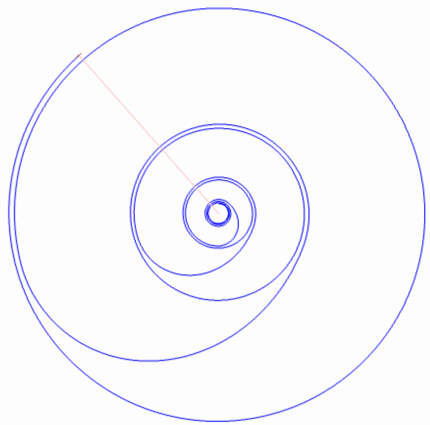
Robotics and Mechatronics Centre



Dry Friction: Trajectory

Robotics and Mechatronics Centre

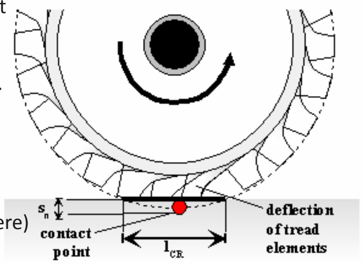
- The wheel behaves approximately like an ideal rolling wheel as long as the tire adheres to the surface.
- There is only a small lateral deflection
- When the speed becomes to large, the wheel enters sliding friction until the radius is wide enough to move the lateral force below the threshold value.



Level 3: Slip-Based Wheel

TUM + DLR Robotics and Mechatronics Centre

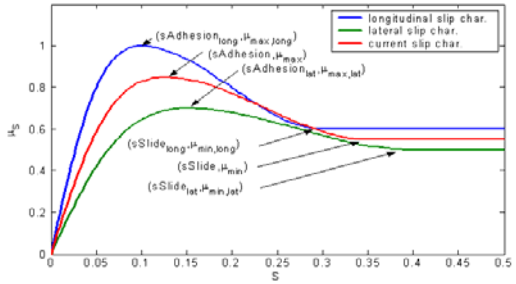
- The tread elements are temporarily deflected in the tread shuffle. The force is transmitted according to this deflection.
- To describe the force transmission, the concept of "slip" is widely used.
- The slip is defined to be the quotient of the slip-velocity and the roll-velocity and represents (roughly speaking) the fraction of wheel spin.
- The slip is a dimensionless size that is proportional to the mean deflection of the tread elements. (Presuming the tread elements adhere)



Level 4: Slip Characteristics

TUM + DLR Robotics and Mechatronics Centre

- Dependence of the transmission forces on the slip.

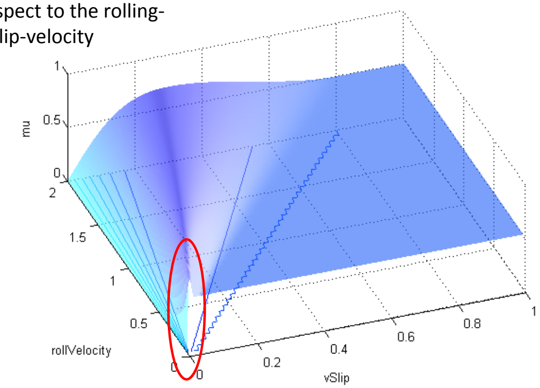


- Unfortunately, the slip turns out to be inappropriate for low rolling-velocities. Thus, its explicit computation is avoided.

Level 4: Slip Characteristics

TUM + DLR Robotics and Mechatronics Centre

Here, the slip-characteristics are displayed with respect to the rolling-velocity and the slip-velocity

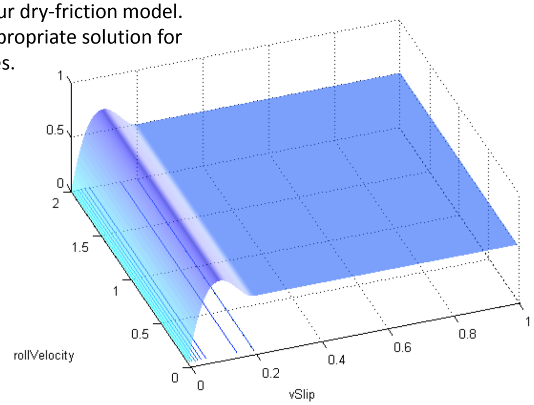


the curve reaches a singular point for vRoll->0

Level 4: Slip Characteristics

TUM + DLR Robotics and Mechatronics Centre

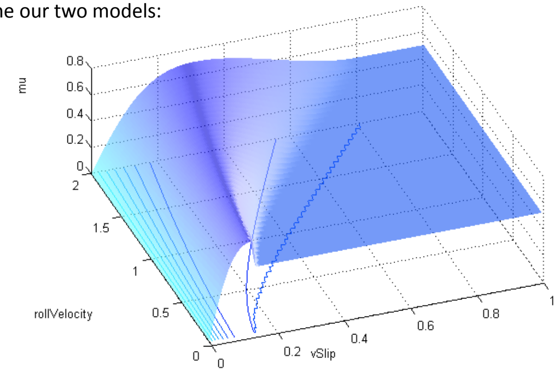
But, we still have our dry-friction model. It represents an appropriate solution for low rolling velocities.



Level 4: Slip Characteristics

Robotics and Mechatronics Centre

So… let's combine our two models:

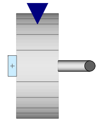


Finally, the computation of the slip is avoided and the model is stable and accurate for all rolling-velocities.

Slip Based Wheel

Robotics and Mechatronics Centre

Now let us implement a slip-based wheel:



The only thing we need to do is:

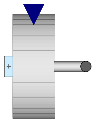
- make vAdhesion and vSlip proportional to the rolling speed.
- Provide minimum values in order to avoid a singularity at w = 0
- Furthermore, we make the normal load dynamic. (we need this later on)

```
model IdealWheelJoint

  RealInput dynamicLoad(unit="N")
  parameter SI.Velocity vAdhesion_min ;
  parameter SI.Velocity vSlide_min ;
  parameter Real sAdhesion ;
  parameter Real sSlide;
  […]

equation
  […]

  vAdhesion = max(
    sAdhesion*abs(radius*w_roll),
    vAdhesion_min
  );
  vSlide = max(
    sSlide*abs(radius*w_roll),
    vSlide_min
  );
  fN = max(0, N+dynamicLoad);
  f = fN*TripleS_Func(vAdhesion,vSlide,
                      mu_A,mu_S,v_slip);
end IdealWheelJoint;
```


Slip Based Wheel

Robotics and Mechatronics Centre

Now let us implement a slip-based wheel:



Still the model is very simple

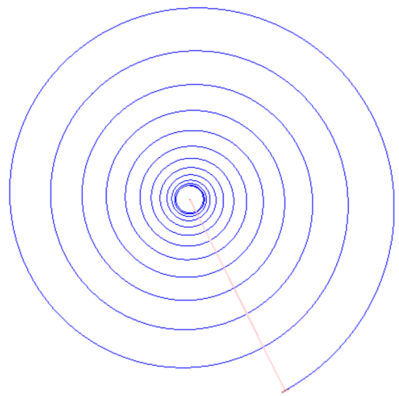
- No camber influence
- No self-alignment
- No bore torque
- No dynamic tire behavior.
- Etc..

```
model IdealWheelJoint

  RealInput dynamicLoad(unit="N")
  parameter SI.Velocity vAdhesion_min ;
  parameter SI.Velocity vSlide_min ;
  parameter Real sAdhesion ;
  parameter Real sSlide;
 […]

equation
  […]

  vAdhesion = max(
    sAdhesion*abs(radius*w_roll),
    vAdhesion_min
  );
  vSlide = max(
    sSlide*abs(radius*w_roll),
    vSlide_min
  );
  fN = max(0, N+dynamicLoad);
  f = fN*TripleS_Func(vAdhesion,vSlide,
                      mu_A,mu_S,v_slip);
end IdealWheelJoint;
```


Slip Based: Trajectory

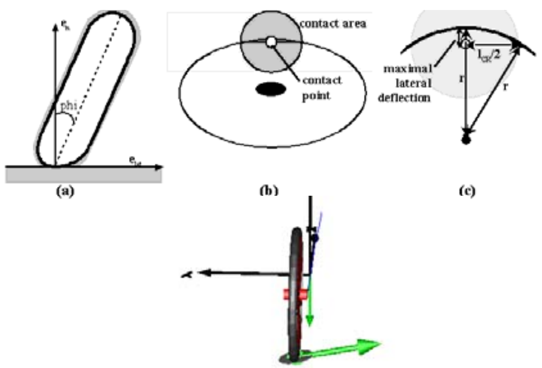
Robotics and Mechatronics Centre

- The increasing speeds leads enables a higher lateral slip-velocity.
- Hence, the trajectory resembles a spiral.



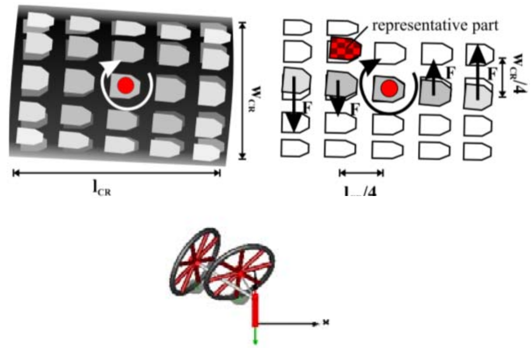
Bonus: Influence of Camber

Robotics and Mechatronics Centre



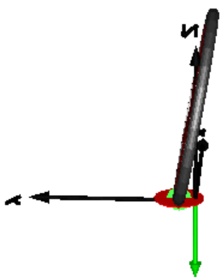
Bonus: Influence of Bore-Torque…

Robotics and Mechatronics Centre



Bonus: Influence of Self-Alignment

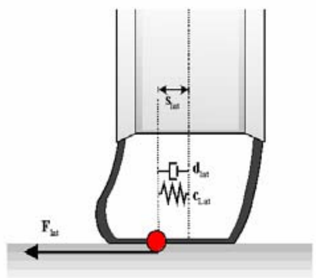
Robotics and Mechatronics Centre



Bonus: Tyre Deformation

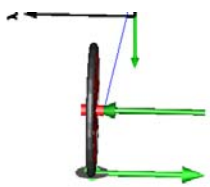
Robotics and Mechatronics Centre

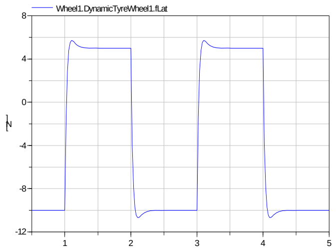
- Longitudinal and lateral deflections are modeled by virtual spring-damper systems.
- The velocity of the deformation influences the slip-velocity.
- The shift of the contact-point leads to additional torques.



Bonus: Tyre Deformation

TUM + DLR
Robotics and Mechatronics Centre





Questions ?