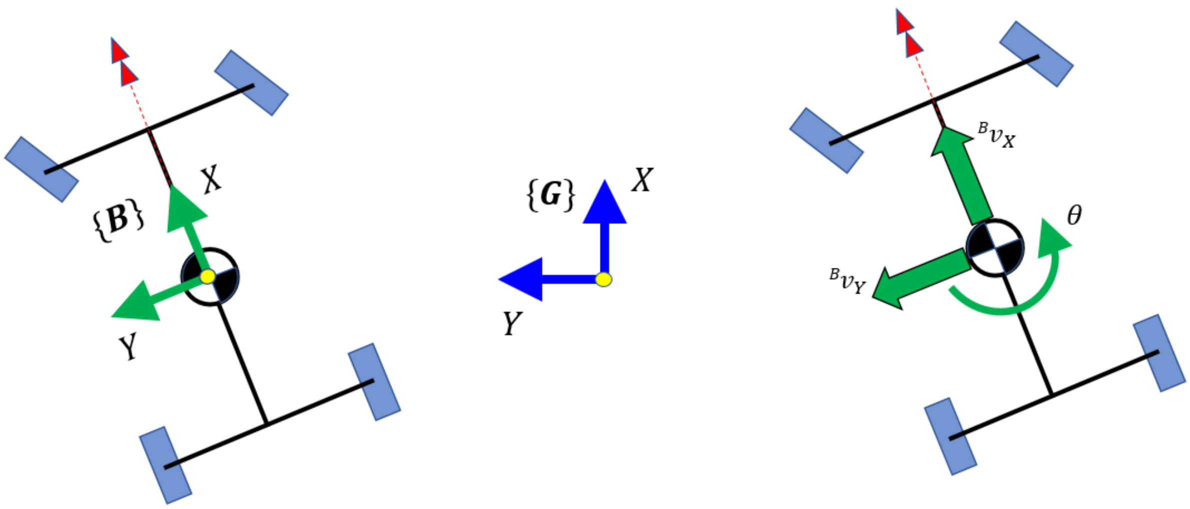
# Derive the equations of motion of a 4 wheel, 3-dof car model:

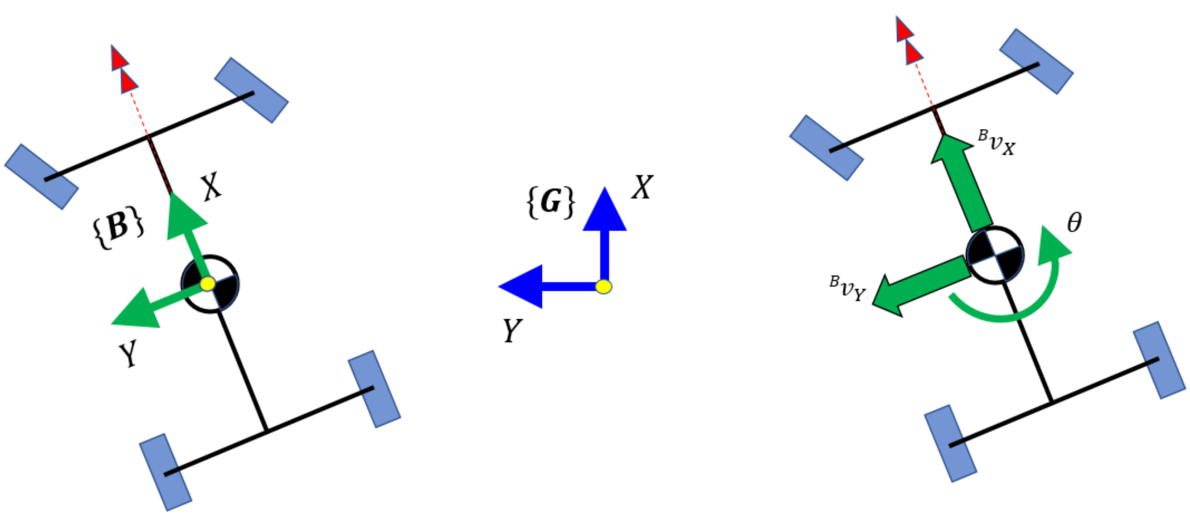



In this script we'll derive the equations of motion for a 4 wheel remote controlled car. Specifically we'll represent the car dynamics via 3 degrees of freedom.  Some highlights include:

1. We'll start by defining Newton's 2nd law for a rigid body
2. We'll use the Symbolic math toolbox to help us implement and manipulate Newton's laws.
3. We'll automatically convert the derived equations of motion into Simulink blocks

```
Bradley Horton : 27-Nov-2017, bradley.horton@mathworks.com.au
```

## Equations of motion according to a body fixed frame:




Recall our fundamental equations of motion for a RIGID body - these equations are expressed in the body fixed vehicle {B}-frame:

$$^{B}F = m.(\ {}^{B}_{-}\dot{v}_C \ + {}^{B}_{G}\omega_B \times {}^{B}_{G}v_C \ )$$

$$ {}^{B}M = {}^{B}I \,.{}^{B}_{-}\dot{\omega}_B \;+\; {}^{B}_{G}\omega_B \times ({}^{B}I \,.{}^{B}_{G}\omega_B) $$

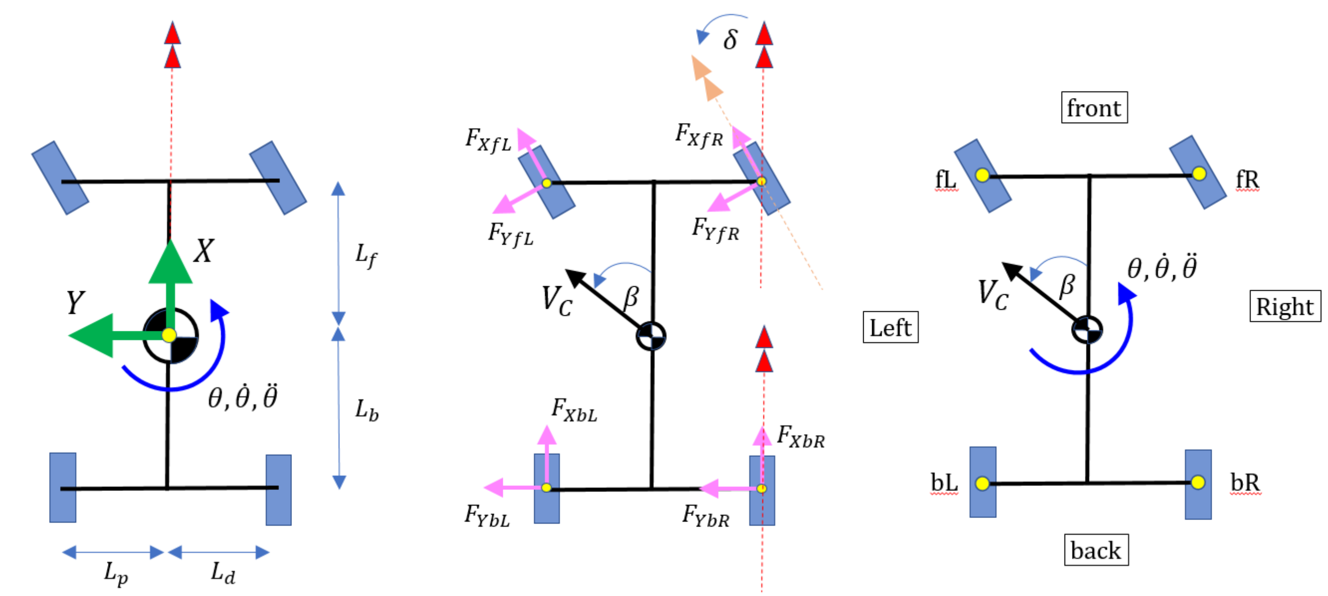
where:

- ${}^{B}_{G}v_C$ : A vector representing the vehicles ***velocity*** of the centre of mass C. The vector is expressed in components of the B-frame. The G subscript indicates that the "measurement" of the velocity is as seen by the G-frame.
- ${}^{B}_{-}\dot{v}_C = {}_{B}\left(\dfrac{d\ {}^{B}_{G}v_C}{dt}\right)$ : the derivative of ${}^{B}_{G}v_C$ as seen by the B-frame, and expressed in components of the B-frame. So if we integrate ${}^{B}_{-}\dot{v}_C$, then we'll get ${}^{B}_{G}v_C$.
- ${}^{B}_{G}\omega_B$ : the angular ***velocity*** of the B-frame as observed by the G-frame, and expressed in components of the B-frame.
- ${}^{B}_{-}\dot{\omega}_B = {}_{B}\left(\dfrac{d\ {}^{B}_{G}\omega_B}{dt}\right)$ : the derivative of ${}^{B}_{G}\omega_B$ as seen by the B-frame, and expressed in components of the B-frame. So if we integrate ${}^{B}_{-}\dot{\omega}_B$, then we'll get ${}^{B}_{G}\omega_B$.
- ${}^{B}I$ : the Inertia of the body computed about the B-frame which is attached to the body's center of mass.
- *B-frame* : the body fixed frame attached to the body's center of mass.
- G-*frame* : the inertial reference frame.
- $A \,.\, B$ : matrix A multiplied by matrix B.
- $a \times b$ : vector cross product.

For our system, the degrees of freedom are:

1. ${}^{B}_{G}v_{CX}$
2. ${}^{B}_{G}v_{CY}$
3. $\theta$

## So let's explore our FORCE equation:




```
syms t theta(t) v_xB(t) v_yB(t) m
```

```
theta_dot = diff(theta(t), t);

w_B       = [       0,         0,    theta_dot  ].' ;
v_B       = [ v_xB(t),    v_yB(t),          0  ].';
v_B_dot   = diff(v_B, t);
w_B_dot   = diff(w_B, t);
```

So our FORCE equation becomes:

```
F_RHS = m*(v_B_dot + cross(w_B, v_B) )
```

F_RHS =

$$\begin{pmatrix} -m\left(v_{yB}(t)\frac{\partial}{\partial t}\theta(t) - \frac{\partial}{\partial t}v_{xB}(t)\right) \\ m\left(v_{xB}(t)\frac{\partial}{\partial t}\theta(t) + \frac{\partial}{\partial t}v_{yB}(t)\right) \\ 0 \end{pmatrix}$$

If we focus on the XY plane we see that the FORCE equations for our differential drive mobile robot is:

- $$\begin{bmatrix} {}^{B}F_{X} \\ {}^{B}F_{Y} \end{bmatrix} = m.\begin{bmatrix} {}^{B}_{G}a_{Cx} \\ {}^{B}_{G}a_{Cy} \end{bmatrix} = m.\begin{bmatrix} {}^{B}_{G}\dot{v}_{Cx} - \dot{\theta}.\,{}^{B}_{G}v_{Cy} \\ {}^{B}_{G}\dot{v}_{Cy} + \dot{\theta}.\,{}^{B}_{G}v_{Cx} \end{bmatrix}$$ ............. ***(1.)***

## Similarly let's consider our MOMENT equation:

```
syms I_xx I_xy I_xz I_yy I_yz I_zz

I       =  [ I_xx, I_xy, I_xz;
             I_xy, I_yy, I_yz;
             I_xz, I_yz, I_zz; ];

M_RHS     = I*w_B_dot + cross(w_B, I*w_B)
```

M_RHS =

$$\begin{pmatrix} I_{xz}\frac{\partial^2}{\partial t^2}\theta(t) - I_{yz}\left(\frac{\partial}{\partial t}\theta(t)\right)^2 \\ I_{yz}\frac{\partial^2}{\partial t^2}\theta(t) + I_{xz}\left(\frac{\partial}{\partial t}\theta(t)\right)^2 \\ I_{zz}\frac{\partial^2}{\partial t^2}\theta(t) \end{pmatrix}$$

If we focus on the rotational motion about the Z-axis, we see that the TORQUE equation for our car is simply:

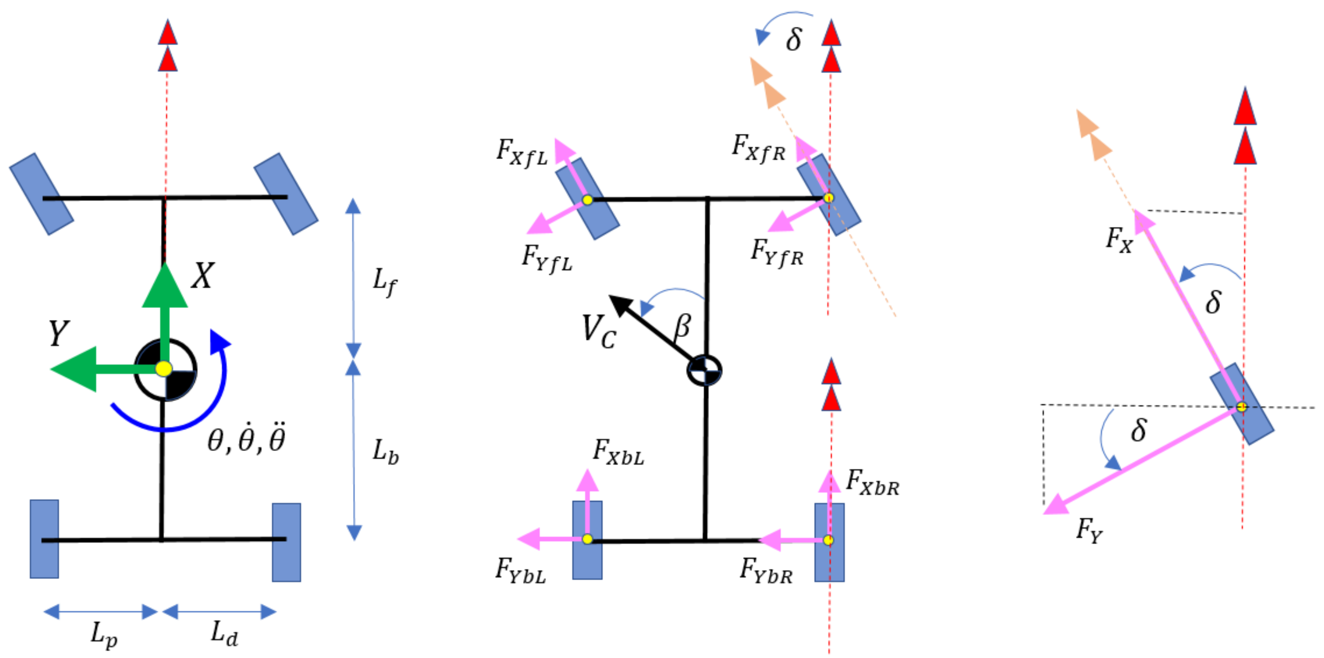
- $M_{zB} = I_{ZZ}\,.\,\ddot{\theta}$ ............. ***(2.)***

## Determine the Net Force and moments in vehicle body {B}-frame:

```
syms L_f  L_b  L_p  L_d
syms delta          % steering angle

syms F_XfR F_YfR   % Front RIGHT tyre forces
syms F_XfL F_YfL   % Front LEFT  tyre forces

syms F_XbR F_YbR   % Back RIGHT tyre forces
syms F_XbL F_YbL   % Back Left  tyre forces
```




Let's define the X and Y force components at each wheel AND also the position of each wheel's centroid - we'll express these quantities in componentsof the **{B}-frame.**

```
Cd = cos(delta);
Sd = sin(delta);
```

Note that the columns in the matrices represent the wheels in the following order:

- $[\mathrm{FRONT_{Right}} \quad \mathrm{FRONT_{Left}} \quad \mathrm{BACK_{Right}} \quad \mathrm{BACK_{Left}}]$

So here are the Force and Position matrices  - expressed  in componentsof the **{B}-frame**

```
%                          #1_fR                    #2_fL          #3_bR     #4_bL
Fw_mat = [ (F_XfR*Cd - F_YfR*Sd),   (F_XfL*Cd - F_YfL*Sd),         F_XbR,    F_XbL;
           (F_XfR*Sd + F_YfR*Cd),   (F_XfL*Sd + F_YfL*Cd),         F_YbR,    F_YbL;
                               0,                       0,             0,        0;
         ];
%                          #1_fR                    #2_fL          #3_bR     #4_bL
Pw_mat = [                  L_f ,                    L_f,        (-L_b),   (-L_b);
                          (-L_d),                    L_p,        (-L_d),     L_p ;
                               0,                      0,             0,        0;
         ];
```

```
% and I'll write out a matrix to a Simulink block too:
Fb_mat_fR_fL_bR_bL = Fw_mat(1:2,:);
```

And now process:

```
F_X = 0;   F_Y = 0;  M_Z = 0;
for kk=1:4
   % accumulate the x and y components of Force
   F_X =  F_X + Fw_mat(1,kk);
   F_Y =  F_Y + Fw_mat(2,kk);

   tmp_F = Fw_mat(:,kk);
   tmp_r = Pw_mat(:,kk);
   % compute moment created by each force:  M = r x F
   tmp_M = cross(tmp_r, tmp_F);

   % accumulate the moments
   M_Z = M_Z + tmp_M(3);
end
```

and echo the results

```
F_X
```

F_X = $F_{\mathrm{XbL}} + F_{\mathrm{XbR}} + F_{\mathrm{XfL}} \cos(\delta) + F_{\mathrm{XfR}} \cos(\delta) - F_{\mathrm{YfL}} \sin(\delta) - F_{\mathrm{YfR}} \sin(\delta)$

and

```
F_Y
```

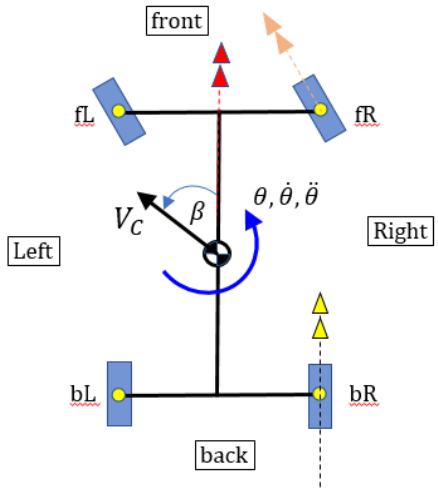
F_Y = $F_{\mathrm{YbL}} + F_{\mathrm{YbR}} + F_{\mathrm{YfL}} \cos(\delta) + F_{\mathrm{YfR}} \cos(\delta) + F_{\mathrm{XfL}} \sin(\delta) + F_{\mathrm{XfR}} \sin(\delta)$

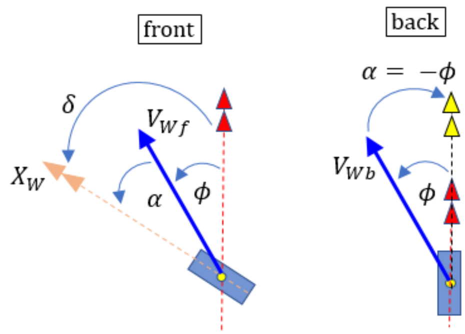
and

```
M_Z
```

M_Z = $F_{\mathrm{XbR}} L_d - F_{\mathrm{YbR}} L_b - F_{\mathrm{YbL}} L_b - F_{\mathrm{XbL}} L_p + L_d \left(F_{\mathrm{XfR}} \cos(\delta) - F_{\mathrm{YfR}} \sin(\delta)\right) + L_f \left(F_{\mathrm{YfL}} \cos(\delta) + F_{\mathrm{XfL}} \sin(\delta)\right) +$

And also compute the velocities of each wheel centroid: $v_W = v_{\mathrm{car}} + \omega \times r_W$




```
Vw_fR = v_B  +  cross(w_B, Pw_mat(:,1) )
```

Vw_fR =

$$\begin{pmatrix} L_d \frac{\partial}{\partial t}\, \theta(t) + v_{xB}(t) \\ L_f \frac{\partial}{\partial t}\, \theta(t) + v_{yB}(t) \\ 0 \end{pmatrix}$$

```
Vw_fL = v_B  +  cross(w_B, Pw_mat(:,2) )
```

Vw_fL =

$$\begin{pmatrix} v_{xB}(t) - L_p \frac{\partial}{\partial t}\, \theta(t) \\ L_f \frac{\partial}{\partial t}\, \theta(t) + v_{yB}(t) \\ 0 \end{pmatrix}$$

```
Vw_bR = v_B  +  cross(w_B, Pw_mat(:,3) )
```

Vw_bR =

$$\begin{pmatrix} L_d \frac{\partial}{\partial t}\, \theta(t) + v_{xB}(t) \\ v_{yB}(t) - L_b \frac{\partial}{\partial t}\, \theta(t) \\ 0 \end{pmatrix}$$

```
Vw_bL = v_B  +  cross(w_B, Pw_mat(:,4) )
```

Vw_bL =

$$\begin{pmatrix} v_{\mathrm{xB}}(t) - L_p \frac{\partial}{\partial t}\ \theta(t) \\ v_{\mathrm{yB}}(t) - L_b \frac{\partial}{\partial t}\ \theta(t) \\ 0 \end{pmatrix}$$

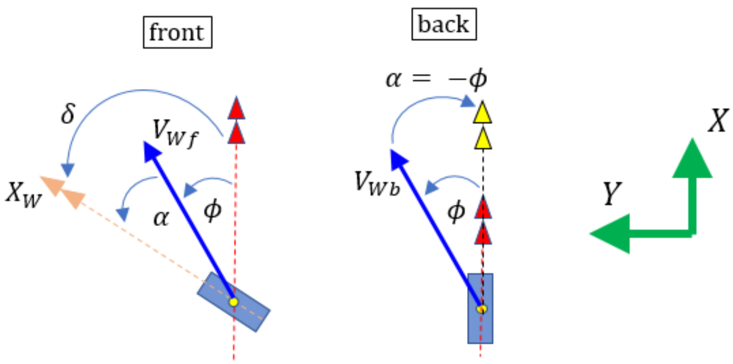
And just remove the Z component of the wheel velocity - (we don't need it):

```
Vw_fR(3) = [];
Vw_fL(3) = [];
Vw_bR(3) = [];
Vw_bL(3) = [];
```

### Define lateral slip angles for each wheel:

We'll define the lateral slip angle as the angle between the wheel's velocity vector and the wheel fixed X axis ${}^{W}X$ . Positive rotations are according to the right hand rule:

- **REF:** **Definition of Lateral tyre slip angle**




The FRONT wheel lateral slip angles:

```
% Y          X
alpha_fR = delta - atan( Vw_fR(2) / Vw_fR(1) )
```

alpha_fR =

$$\delta - \mathrm{atan}\left(\frac{L_f \frac{\partial}{\partial t}\ \theta(t) + v_{\mathrm{yB}}(t)}{L_d \frac{\partial}{\partial t}\ \theta(t) + v_{\mathrm{xB}}(t)}\right)$$

```
alpha_fL = delta - atan( Vw_fL(2) / Vw_fL(1) )
```

alpha_fL =

$$\delta - \operatorname{atan}\left(\frac{L_f \frac{\partial}{\partial t}\ \theta(t) + v_{\mathrm{yB}}(t)}{v_{\mathrm{xB}}(t) - L_p \frac{\partial}{\partial t}\ \theta(t)}\right)$$

The BACK wheel lateral slip angles:

```
                         % Y          X
alpha_bR =  -1 * atan( Vw_bR(2) / Vw_bR(1) )
```

alpha_bR =

$$-\operatorname{atan}\left(\frac{v_{\mathrm{yB}}(t) - L_b \frac{\partial}{\partial t}\ \theta(t)}{L_d \frac{\partial}{\partial t}\ \theta(t) + v_{\mathrm{xB}}(t)}\right)$$

```
alpha_bL =  -1 * atan( Vw_bL(2) / Vw_bL(1) )
```

alpha_bL =

$$-\operatorname{atan}\left(\frac{v_{\mathrm{yB}}(t) - L_b \frac{\partial}{\partial t}\ \theta(t)}{v_{\mathrm{xB}}(t) - L_p \frac{\partial}{\partial t}\ \theta(t)}\right)$$

### Define some new symbols :

```
syms THE_theta   THE_theta_D   THE_theta_DD
syms THE_v_xB    THE_v_xB_D
syms THE_v_yB    THE_v_yB_D

ACTUAL_list = [theta(t),  diff(theta(t),t),  diff(theta(t),2),      v_xB(t),   diff(v_xB(t)),      v_yB
HOLDER_list = [THE_theta,       THE_theta_D,       THE_theta_DD,   THE_v_xB,       THE_v_xB_D,    THE_v_yB
```

### Combine LHS and RHS of equations:

Combine the left and right hand sides of each equation:

```
F_X_EQN = F_X == F_RHS(1)
```

F_X_EQN =

$$F_{\mathrm{XbL}} + F_{\mathrm{XbR}} + F_{\mathrm{XfL}} \cos(\delta) + F_{\mathrm{XfR}} \cos(\delta) - F_{\mathrm{YfL}} \sin(\delta) - F_{\mathrm{YfR}} \sin(\delta) = -m \left(v_{\mathrm{yB}}(t) \frac{\partial}{\partial t}\ \theta(t) - \frac{\partial}{\partial t}\ v_{\mathrm{xB}}(t)\right)$$

```
F_Y_EQN = F_Y == F_RHS(2)
```

F_Y_EQN =

$$F_{\mathrm{YbL}} + F_{\mathrm{YbR}} + F_{\mathrm{YfL}} \cos(\delta) + F_{\mathrm{YfR}} \cos(\delta) + F_{\mathrm{XfL}} \sin(\delta) + F_{\mathrm{XfR}} \sin(\delta) = m \left( v_{\mathrm{xB}}(t) \frac{\partial}{\partial t} \theta(t) + \frac{\partial}{\partial t} v_{\mathrm{yB}}(t) \right)$$

```
M_Z_EQN = M_Z == M_RHS(3)
```

M_Z_EQN =

$$F_{\mathrm{XbR}} L_d - F_{\mathrm{YbR}} L_b - F_{\mathrm{YbL}} L_b - F_{\mathrm{XbL}} L_p + L_d \left(F_{\mathrm{XfR}} \cos(\delta) - F_{\mathrm{YfR}} \sin(\delta)\right) + L_f \left(F_{\mathrm{YfL}} \cos(\delta) + F_{\mathrm{XfL}} \sin(\delta)\right) +$$

Now populate with new symbols:

```
F_X_EQN = subs(F_X_EQN, ACTUAL_list, HOLDER_list);
F_Y_EQN = subs(F_Y_EQN, ACTUAL_list, HOLDER_list);
M_Z_EQN = subs(M_Z_EQN, ACTUAL_list, HOLDER_list);
```

## Solve for $\dot{v}_{\mathrm{xb}}$ , $\dot{v}_{\mathrm{yb}}$ , $\ddot{\theta}$ :

```
my_sol = solve([F_X_EQN, F_Y_EQN, M_Z_EQN], [THE_v_xB_D, THE_v_yB_D, THE_theta_DD] )
```

```
my_sol = struct with fields:
      THE_v_xB_D: [1×1 sym]
      THE_v_yB_D: [1×1 sym]
    THE_theta_DD: [1×1 sym]
```

And you can echo the results if you want:

```
my_sol.THE_v_xB_D
```

ans =

$$\frac{F_{\mathrm{XbL}} + F_{\mathrm{XbR}} + F_{\mathrm{XfL}} \cos(\delta) + F_{\mathrm{XfR}} \cos(\delta) - F_{\mathrm{YfL}} \sin(\delta) - F_{\mathrm{YfR}} \sin(\delta) + \mathrm{THE}_{v,\mathrm{yB}} \mathrm{THE}_{\theta,D} m}{m}$$

and

```
my_sol.THE_v_yB_D
```

ans =

$$\frac{F_{\mathrm{YbL}} + F_{\mathrm{YbR}} + F_{\mathrm{YfL}} \cos(\delta) + F_{\mathrm{YfR}} \cos(\delta) + F_{\mathrm{XfL}} \sin(\delta) + F_{\mathrm{XfR}} \sin(\delta) - \mathrm{THE}_{v,\mathrm{xB}} \mathrm{THE}_{\theta,D} m}{m}$$

and

```
my_sol.THE_theta_DD
```

ans =

$$\frac{F_{\mathrm{XbR}}\,L_d - F_{\mathrm{YbR}}\,L_b - F_{\mathrm{YbL}}\,L_b - F_{\mathrm{XbL}}\,L_p + F_{\mathrm{XfR}}\,L_d\cos(\delta) + F_{\mathrm{YfL}}\,L_f\cos(\delta) + F_{\mathrm{YfR}}\,L_f\cos(\delta) - F_{\mathrm{XfL}}\,L_p\cos(\delta)}{I_{\mathrm{zz}}}$$

### Convert expressions into HOLDER variables:

The slide slip angles:

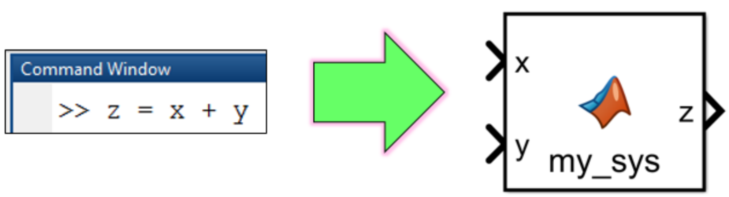
```
alpha_fR = subs(alpha_fR, ACTUAL_list, HOLDER_list);
alpha_fL = subs(alpha_fL, ACTUAL_list, HOLDER_list);
alpha_bR = subs(alpha_bR, ACTUAL_list, HOLDER_list);
alpha_bL = subs(alpha_bL, ACTUAL_list, HOLDER_list);
```

The wheel velocities:

```
Vw_fR    = subs(Vw_fR, ACTUAL_list, HOLDER_list);
Vw_fL    = subs(Vw_fL, ACTUAL_list, HOLDER_list);
Vw_bR    = subs(Vw_bR, ACTUAL_list, HOLDER_list);
Vw_bL    = subs(Vw_bL, ACTUAL_list, HOLDER_list);
```

### Create Simulink blocks of our $\dot{v}_{\mathrm{xb}}$ , $\dot{v}_{\mathrm{yb}}$ , $\ddot{\theta}$ equations:

To use/solve these derived equations of motion we'll create a MATLAB Function block that can be used inside Simulink:




Now create the MATLAB function blocks for Simulink:

```
tf_i_should_create_SL_block = true;
if(true==tf_i_should_create_SL_block)
    MODEL_NAME = 'bh_tmp_model_for_3dof_car';

    if(4==exist(MODEL_NAME))
        close_system(MODEL_NAME, 0);
        delete(MODEL_NAME);
    end
    new_system(MODEL_NAME)
    open_system(MODEL_NAME)

    % ---------------------------------------------------------------------------------
    BLOCK_NAME = [MODEL_NAME,'/3DOF_EOM'];

    % specify the order of the signals for the input ports
    vars_INPUTS = {m, I_zz,             ...
                   L_f,  L_b,  L_p,  L_d, ...
                   delta,        ...
                   F_XfR, F_YfR, ...   % Front RIGHT tyre forces
```

```
                F_XfL, F_YfL, ...   % Front LEFT  tyre forces
                F_XbR, F_YbR, ...   % Back RIGHT tyre forces
                F_XbL, F_YbL, ...   % Back Left  tyre forces
                THE_v_xB,     ...
                THE_v_yB,     ...
                THE_theta_D   ...
                };

 % Put BOTH equations into one block
 matlabFunctionBlock( BLOCK_NAME, ...
                      [my_sol.THE_v_xB_D; my_sol.THE_v_yB_D], ...
                      my_sol.THE_theta_DD,                    ...
                      [F_X; F_Y; M_Z],                        ...
                      Fb_mat_fR_fL_bR_bL,                     ...
                     'Optimize', false, ...
                     'Vars', vars_INPUTS, ...
                     'Outputs', {'vB_xy_D', 'theta_DD', 'FxyMz', 'Fb_mat_fR_fL_bR_bL'}   );

% make the block YELLOW
set_param( BLOCK_NAME,'BackgroundColor', '[0.996078, 1.000000, 0.705882]');
 % ---------------------------------------------------------------------------
BLOCK_NAME = [MODEL_NAME,'/LAT_SLIP_ANGLES'];
 % specify the order of the signals for the input ports
 vars_INPUTS = {L_f,  L_b,  L_p,  L_d, ...
                delta,         ...
                THE_v_xB,     ...
                THE_v_yB,     ...
                THE_theta_D   ...
                };

 % Put BOTH equations into one block
 matlabFunctionBlock( BLOCK_NAME, ...
                      alpha_fR, alpha_fL,  alpha_bR, alpha_bL, ...
                     'Optimize', false, ...
                     'Vars', vars_INPUTS, ...
                     'Outputs', {'alpha_fR', 'alpha_fL',  'alpha_bR', 'alpha_bL'}   );

% make the block YELLOW
set_param( BLOCK_NAME,'BackgroundColor', '[0.996078, 1.000000, 0.705882]');
 % ---------------------------------------------------------------------------
BLOCK_NAME = [MODEL_NAME,'/WHEEL_VEL'];
 % specify the order of the signals for the input ports
 vars_INPUTS = {L_f,  L_b,  L_p,  L_d, ...
                THE_v_xB,     ...
                THE_v_yB,     ...
                THE_theta_D   ...
                };

 % Put BOTH equations into one block
 matlabFunctionBlock( BLOCK_NAME, ...
                      Vw_fR, Vw_fL,  Vw_bR, Vw_bL, ...
                     'Optimize', false, ...
                     'Vars', vars_INPUTS, ...
                     'Outputs', {'Vw_fR', 'Vw_fL',  'Vw_bR', 'Vw_bL'}   );

% make the block YELLOW
set_param( BLOCK_NAME,'BackgroundColor', '[0.996078, 1.000000, 0.705882]');
```

```
end
```

```
Evaluating callback 'PostLoadFcn' for simulink
Callback: setsysloc_simulink(bdroot)
Evaluating callback 'LoadFcn' for simulink/Sources/Signal Builder
Callback: sigbuilder_block('load');
Evaluating callback 'LoadFcn' for simulink/Sinks/XY Graph
Callback: sfunxy([],[],[],'LoadBlock')
Evaluating callback 'LoadFcn' for simulink/Sources/Waveform Generator
Callback: set_param(gcb,'LoadFlag','1');


Evaluating callback 'LoadFcn' for simulink/Model-Wide Utilities/Model Info
Callback: slcm LoadBlock;
Evaluating callback 'LoadFcn' for simulink/Math Operations/Slider Gain
Callback: sliderGain_cb(gcbh, 'load');
```