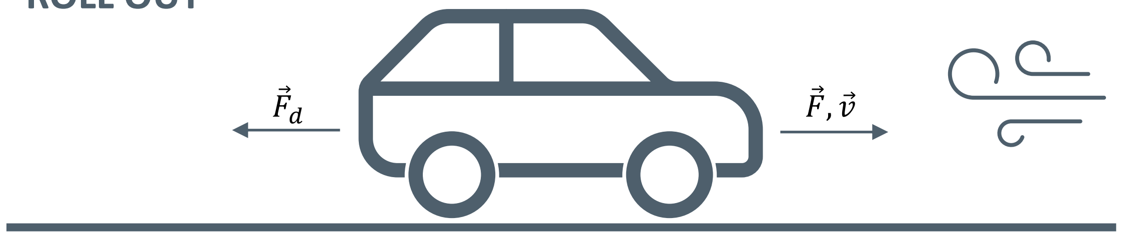
Robin Leuering
robin-leuering.de

ROLL OUT



DYNAMIC FORCE

$$F(v) = F_d(v) + m\frac{dv}{dt}$$

DRAG FORCE

$$F_d(v) = Cv^2 + Bv + A$$

ZERO FORCE

$$-m\frac{dv}{dt} = Cv^2 + Bv + A \longrightarrow$$

VALUE SEPARATION

$$\frac{1}{m}dt = \frac{-1}{Cv^2 + Bv + A}dv$$

LEFT SIDE

$$\int \frac{1}{m}dt = \frac{t}{m} + K_1$$

RIGHT SIDE

$$\int \frac{-1}{Cv^2 + Bv + A}dv = \frac{-2}{\sqrt{D}}\arctan\left(\frac{2Cv + B}{\sqrt{D}}\right) + K_2 \qquad D = 4\mathrm{AC} - B^2$$

COMMON SOLUTION

$$t = \frac{-2m}{\sqrt{D}}\arctan\left(\frac{2Cv + B}{\sqrt{D}}\right) + K_2 - K_1 = \frac{-2m}{\sqrt{D}}\arctan\left(\frac{2Cv + B}{\sqrt{D}}\right) + K$$

INITIAL VALUES

$$v(t = 0) = v_0 \longrightarrow$$

$$t = 0 = \frac{2m}{\sqrt{\mathrm{D}}}\arctan\left(\frac{2Cv_0 + B}{\sqrt{\mathrm{D}}}\right) - K$$

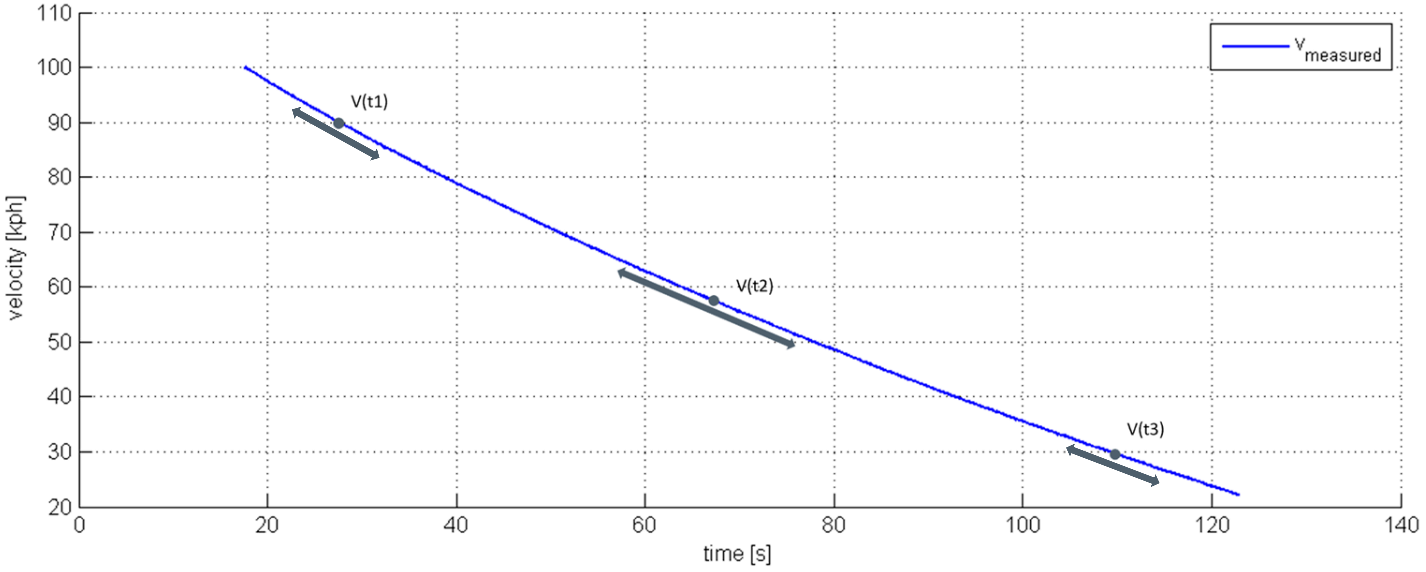
$$K = \frac{2m}{\sqrt{\mathrm{D}}}\arctan\left(\frac{2Cv_0 + B}{\sqrt{\mathrm{D}}}\right)$$

REARRANGE EQUATION

$$t = \frac{2m}{\sqrt{\mathrm{D}}}\arctan\left(\frac{2Cv_0 + B}{\sqrt{\mathrm{D}}}\right) - \frac{2m}{\sqrt{\mathrm{D}}}\arctan\left(\frac{2Cv + B}{\sqrt{\mathrm{D}}}\right)$$

$$v(t) = \frac{\sqrt{\mathrm{D}}\tan\left(\arctan\left(\frac{2Cv_0 + B}{\sqrt{\mathrm{D}}}\right) - \frac{t\sqrt{\mathrm{D}}}{2m}\right) - B}{2C}$$

ROLL OUT



ZERO FORCE

$$\begin{bmatrix} Cv(t1)^2 + Bv(t1) + A \\ Cv(t2)^2 + Bv(t2) + A \\ Cv(t3)^2 + Bv(t3) + A \end{bmatrix} = -m \begin{bmatrix} \frac{dv}{dt}\big|_{t1} \\ \frac{dv}{dt}\big|_{t2} \\ \frac{dv}{dt}\big|_{t3} \end{bmatrix}$$

MATRIX-VECTOR SEPARATION

$$\begin{bmatrix} Cv(t_1)^2 + Bv(t_1) + A \\ Cv(t_2)^2 + Bv(t_2) + A \\ Cv(t_3)^2 + Bv(t_3) + A \end{bmatrix} = \begin{bmatrix} v(t_1)^2 & v(t_1) & 1 \\ v(t_2)^2 & v(t_2) & 1 \\ v(t_3)^2 & v(t_3) & 1 \end{bmatrix} \begin{bmatrix} C \\ B \\ A \end{bmatrix} = -m \begin{bmatrix} \frac{dv}{dt}\big|_{t_1} \\ \frac{dv}{dt}\big|_{t_2} \\ \frac{dv}{dt}\big|_{t_3} \end{bmatrix}$$

$$\vec{z} \quad = \quad T \quad \vec{x} \quad = \quad \vec{z}$$

SOLVE LINEAR SYSTEM

$$T^{-1}\, T\, \vec{x} = I\, \vec{x} = T^{-1}\, \vec{z} \longrightarrow \vec{x} = T^{-1}\, \vec{z}$$

$$\begin{bmatrix} C \\ B \\ A \end{bmatrix} = -m \begin{bmatrix} v(t_1)^2 & v(t_1) & 1 \\ v(t_2)^2 & v(t_2) & 1 \\ v(t_3)^2 & v(t_3) & 1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{dv}{dt}\big|_{t_1} \\ \frac{dv}{dt}\big|_{t_2} \\ \frac{dv}{dt}\big|_{t_3} \end{bmatrix}$$

ITERATION

$$v(t_1) = v_{meas}(t_1)$$
$$v(t_2) = v_{meas}(t_2)$$
$$v(t_3) = v_{meas}(t_3)$$

$$\begin{bmatrix} C \\ B \\ A \end{bmatrix} = -m \begin{bmatrix} v(t_1)^2 & v(t_1) & 1 \\ v(t_2)^2 & v(t_2) & 1 \\ v(t_3)^2 & v(t_3) & 1 \end{bmatrix}^{-1} \begin{bmatrix} \frac{dv}{dt}\big|_{t_1} \\ \frac{dv}{dt}\big|_{t_2} \\ \frac{dv}{dt}\big|_{t_3} \end{bmatrix}$$

$$D = 4\mathrm{AC} - B^2$$

$$v(t) = \frac{\sqrt{D}\tan\left(\arctan\left(\frac{2Cv_0 + B}{\sqrt{D}}\right) - \frac{t\sqrt{D}}{2m}\right) - B}{2C}$$

$$\int_0^T (\, v_{meas}(t) - v_{calc}(t)\,)^2 \, dt \approx 0$$

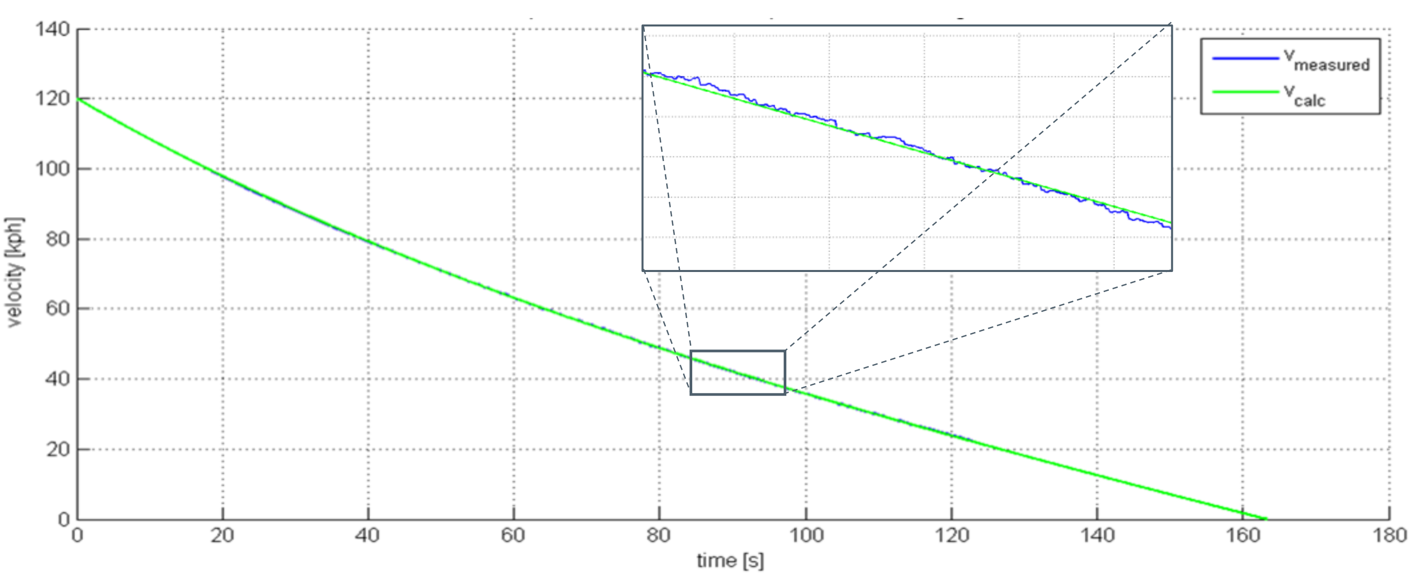
NO

$$v(t_1) = v_{meas}(t_1 \pm \Delta t)$$
$$v(t_2) = v_{meas}(t_2 \pm \Delta t)$$
$$v(t_3) = v_{meas}(t_3 \pm \Delta t)$$

YES

SIMULATION

SIMULATION



```python
"""@author: robin"""
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
"""Load measurment"""
measure = pd.read_csv('rollout_1850.csv',delimiter=';')
vmeas = measure.v.values
tmeas = measure.t.values
"""Parameters"""
m = 1850     # Vehicle mass
k = 50       # Slope calc value
e = np.inf   # Error value init
"""Variable Init"""
coeff = np.zeros(3)
index_f = len(measure)
vappr = np.polyval(np.polyfit(tmeas,vmeas,7), tmeas)
vpart = np.zeros([len(coeff),index_f])
tpart = np.zeros([len(coeff),index_f])
v = np.zeros(len(coeff))
t = np.zeros(len(coeff))
dvdt = np.zeros(len(coeff))
index = np.int64(np.zeros(len(coeff)))
"""Parameter Identification"""
while(e>len(vmeas)*0.01):
    for i in range(0,len(coeff)):
        index[i] = np.random.randint(k,index_f-k)
        v[i] = vappr[index[i]]
        t[i] = tmeas[index[i]]
        dvdt[i] = (vappr[index[i]+k]-vappr[index[i]-k])/(tmeas[index[i]+k]-tmeas[index[i]-k])
    if(index[0] == index[1] or index[0] == index[2] or index[1] == index[2]): continue
    T = np.matrix([[v[0]**2,v[0],1],
                   [v[1]**2,v[1],1],
                   [v[2]**2,v[2],1]])
    z = np.matrix(-m*dvdt.reshape(3,1))
    x = np.matmul(T.I,z)
    A = x[2].item()/3.6
    B = x[1].item()
    C = x[0].item()*3.6
    if (4*A*C-B**2) > 0:
        D = np.sqrt(4*A*C-B**2)
        vtemp = 3.6*(D*np.tan((-tmeas*D/(2*m))+np.arctan((2*C*(vmeas[0]/3.6)+B)/D))-B)/(2*C)
        etemp = np.sum(np.power(vmeas-vtemp,2))
        if etemp < e:
            e = etemp
            vcalc = vtemp
            coeff = np.array([A,B,C])
"""Plot"""
plt.figure(1); plt.rcParams['figure.figsize'] = [16, 12]
plt.xlabel('Time [s]',fontsize=16)
plt.ylabel('Velocity [kph]',fontsize=16)
plt.plot(tmeas,vmeas,c='#284b64',label = 'v_meas',linewidth=3.0)
plt.plot(tmeas,vcalc,label = 'v_calc',linewidth=3.0)
plt.title('Roll out',fontsize=20)
plt.legend(loc='upper right',fontsize=16)
plt.grid(True)
plt.show()
```