Neural Networks for Model Augmentation Using OpenModelica

for MODPROD 2021, Linköping
based on
"Towards Grey Box Modelling in Modelica"

Frederic Bruder and Lars Mikelsons

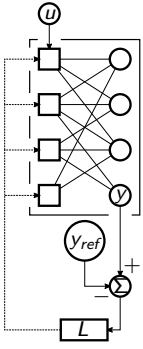
University of Augsburg, Germany

February 4, 2021

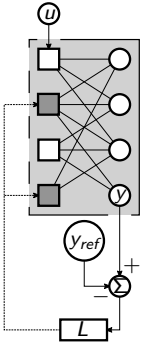
modelling techniques

DAE:
$\boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, t, \boldsymbol{\theta}) = \boldsymbol{0}$

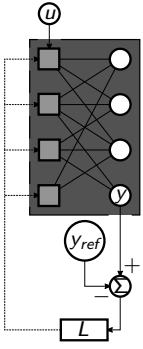
white box



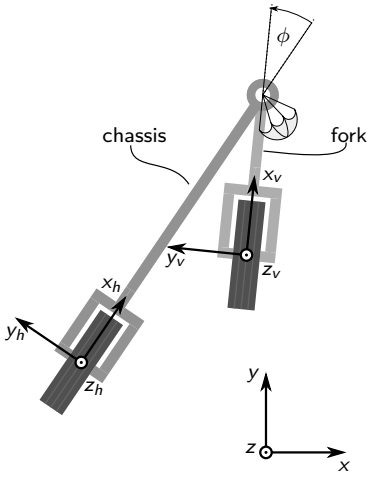
grey box

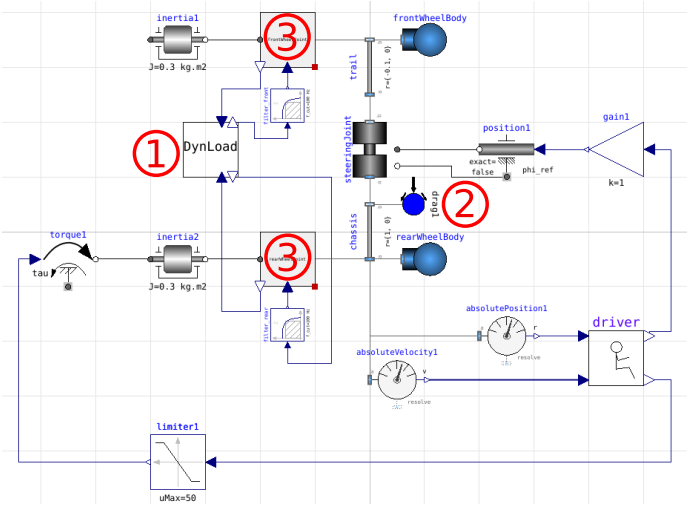


black box



top down view of the motorcycle model | modelica object diagram





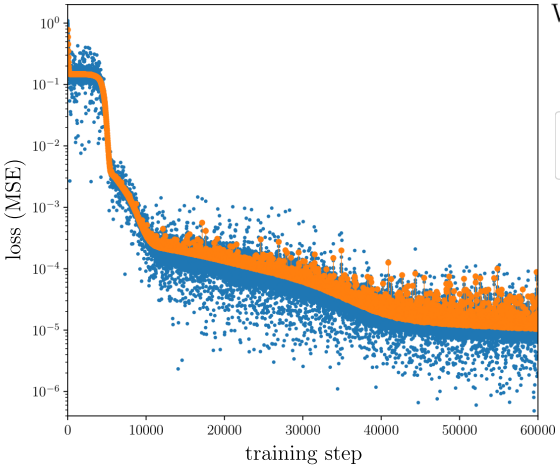
objective: learn these functions!

1 dynamicLoad	$F_{vz} = NN_{dynamicLoad\_front}(F_{hx}, F_{vx})$ $F_{hz} = NN_{dynamicLoad\_rear}(F_{hx}, F_{vx})$
2 drag	$F_{dx} = NN_{drag\_x}(v_x, v_y)$ $F_{dy} = NN_{drag\_y}(v_x, v_y)$
3 wheelJoint_f	$F_{vx} = NN_{WheelJoint\_f\_long}(v_{v,long}, v_{v,lat}, \omega_{v,r})$ $F_{v,lat} = NN_{WheelJoint\_f\_lat}(v_{v,long}, v_{v,lat}, \omega_{v,r})$

application: software

1. build the motorcycle model (OMEdit)
2. simulate the model for some maneuvers and export simulation data (OMPython or DyMat)
3. train nn (neural networks) with IO data (Tensorflow)
4. export the nn parameters as protocol buffers
5. replace the original components by nn (ext. C-interface in Modelica)

loss function over iteration steps: wheelJoint_f_lat

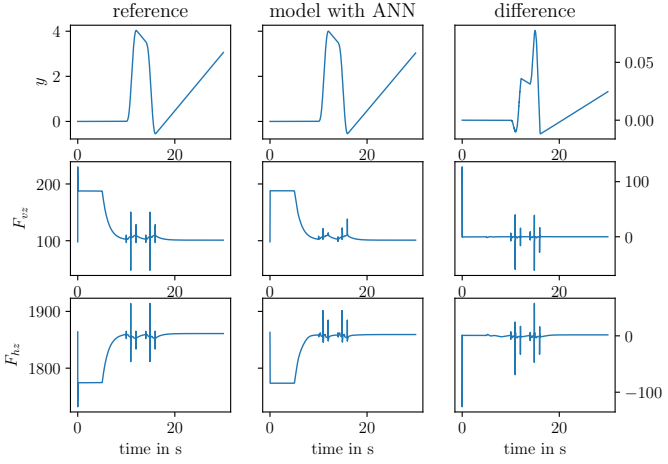


WheelJoint_f_lat1, total training steps: 336000.
Final testing data MSE / $\sigma_{train,test}$:
3.45E-06/5.52E+02=6.25E-09

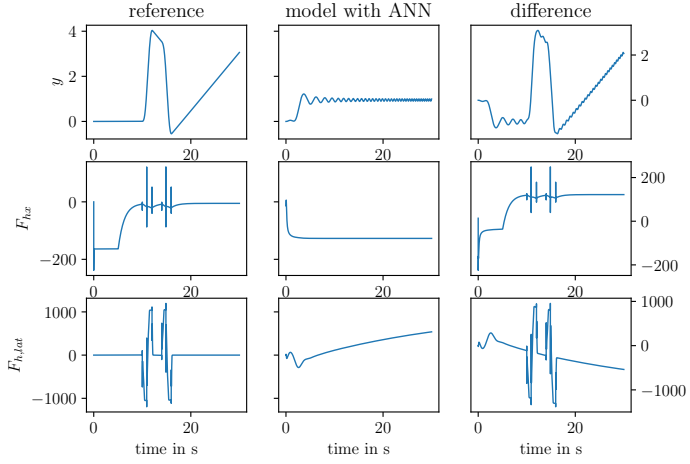
losses of the batches used for the training step
mean testing losses after each epoch

maneuver: [EightCurves, ConstAcc]

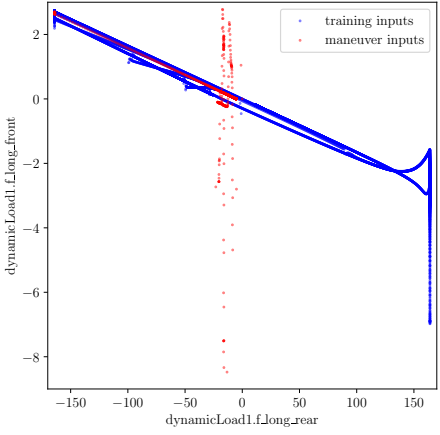
validation on maneuver [DoubleLaneChange, ConstVel]: dynamicLoad



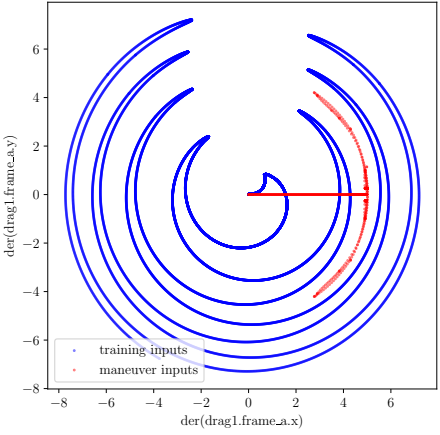
validation on maneuver [DoubleLaneChange, ConstVel]: wheelJoint_f



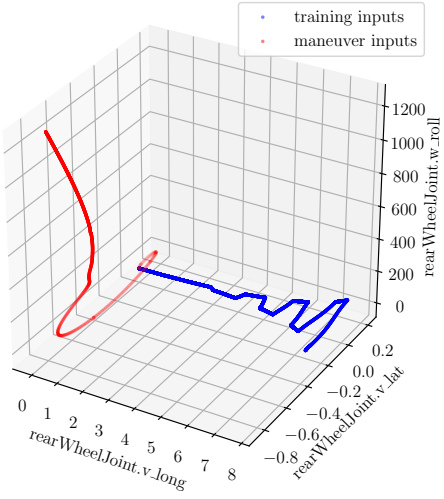
input space: dynamicLoad



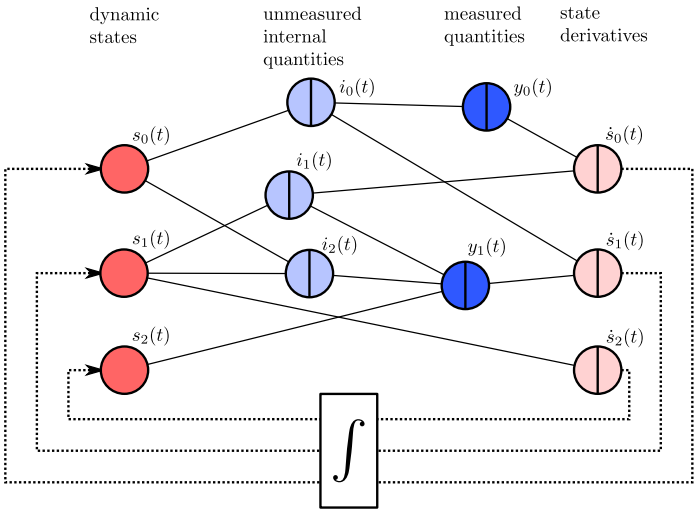
input space: drag



input space: wheelJoint_f



current/future research



1. this approach: access inputs/outputs of the replaced functions
2. next approach: access only measured quantities and differentiate through solver

Further questions (2021 update)

1. Where exactly can we trust the neural network prediction?
2. Which equations to replace?
3. How to formulate the loss function for physical systems?
4. Most promising ways to include prior knowledge about the model and its physics?
5. How to deal with BB equations that contain internal state?
6. How to calculate the gradient of the loss function? (BP through solver vs. adjoint sensitivity)