Force-Feedback and TruckMaker: A Journey

06. October 2020

Content

- Force-Feedback and Windows
- Force-Feedback and MATLAB
- Force-Feedback and Simulink
- Force-Feedback and TruckMaker

Content

- Force-Feedback and Windows
- Force-Feedback and MATLAB
- Force-Feedback and Simulink
- Force-Feedback and TruckMaker

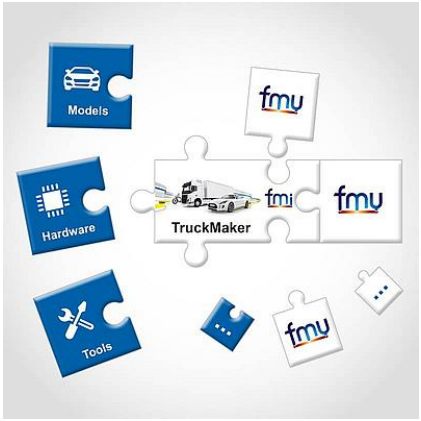
Force-Feedback and Windows

Why am I doing this - better why did I do it?

API

MATLAB® & SIMULINK®

Parallel Toolbox



Force-Feedback and Windows

Where to start? How do I connect to the Racing-Wheel?

How to read the Wheel and execute a Force-Feedback-Effect?

Different APIs available to start from:

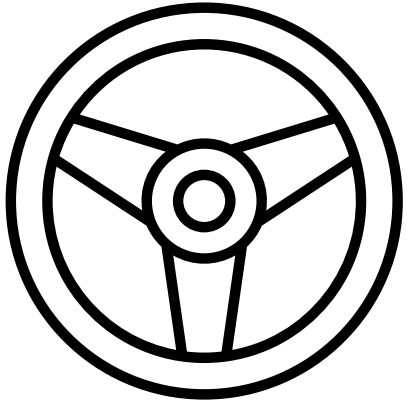
- **DirectInput**
  - very old (1995)
  - hard to code
- **XInput**
  - predecessor to DirectInput
  - Built for XBox360 controller
  - widely used
- **Windows Gaming API**
  - introduced recently, better documentation
  - easy to implement

Force-Feedback and Windows

Windows Gaming API

The API offers different force-feedback effects:

- **Constant Effect**
- **Spring Effect**
- **Damper Effect**
- **Inertia Effect**



For ease of implementation 3 modified constant force effects were implemented:

- **FF-Minus**
- **FF-Plus**
- **FF-Zero**

Additionally, the button and pedal status of the steering-pedal system are callable.

Content

- Force-Feedback and Windows
- Force-Feedback and MATLAB
- Force-Feedback and Simulink
- Force-Feedback and TruckMaker

Force-Feedback and MATLAB

Dynamic Link Library with custom C++ code and Windows Gaming API.

How to integrate custom C++ in MATLAB?

MathWorks provides several frameworks to include custom C++ code with MATLAB:

- **loadlibrary()** – Load C shared library into MATLAB
  - Functions written in C++ must be declared as extern "C“
  - Legacy tool
- **clibgen.buildInterface()** - To create a MATLAB® interface to a C++ library, use the clibgen package
  - Introduced in R2019a
- **mex-function -** Create high-performance MATLAB® functions implemented in modern C++
  - Hard to code, faster execution

Force-Feedback and MATLAB

How to integrate custom C++ in MATLAB?

clibgen Package

```
% Init Racing Wheel
clib.FF_UWP_WIN32_dll.initRacingWheel
clib.FF_UWP_WIN32_dll.initForceFeedback

% INIT STURCTS FOR BUTTON AND WHEELREADINGS
buttonReadings = clib.FF_UWP_WIN32_dll.buttonReadings;
WheelReadings = clib.FF_UWP_WIN32_dll.WheelReadings;

% REED WHEEL AND BUTTONS
clib.FF_UWP_WIN32_dll.readingButton(buttonReadings);
clib.FF_UWP_WIN32_dll.readWheelStatus(WheelReadings);

% EXECUTE FORCE-EFFECT
clib.FF_UWP_WIN32_dll.FF_minus(0.5);
```

Previously build **MATLAB interface library** is utilized by calling it with clibgen.

loadlibrary()

```
% Load Library
loadlibrary('FF_UWP_WIN32_dll','FF_UWP_WIN32_dll.h');

% Init Racing wheel
calllib('FF_UWP_WIN32_dll','initRacingWheel');
calllib('FF_UWP_WIN32_dll','initForceFeedback');

% REED WHEEL AND BUTTONS
buttonReadings =
calllib('FF_UWP_WIN32_dll','readingButton',buttonReadings);
WheelReadings =
calllib('FF_UWP_WIN32_dll','readWheelStatus',WheelReadings);

% EXECUTE FORCE-EFFECT
calllib 'FF_UWP_WIN32_dll','FF_minus',0.5)
```

C-DLL is directly called with **loadlibrary().** No intermediate steps necessary.

Content

- Force-Feedback and Windows
- Force-Feedback and MATLAB
- Force-Feedback and Simulink
- Force-Feedback and TruckMaker

Force-Feedback and Simulink

How to integrate custom C++ in Simulink.

S-Function Builder

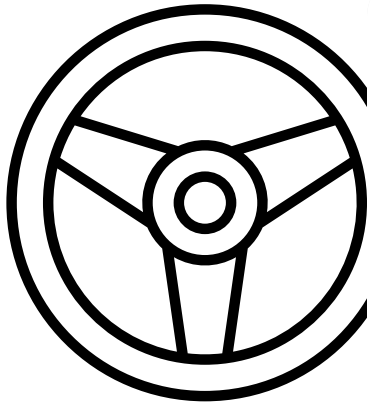
The S-Function-Builder gives full functionality over the following simulation phases:

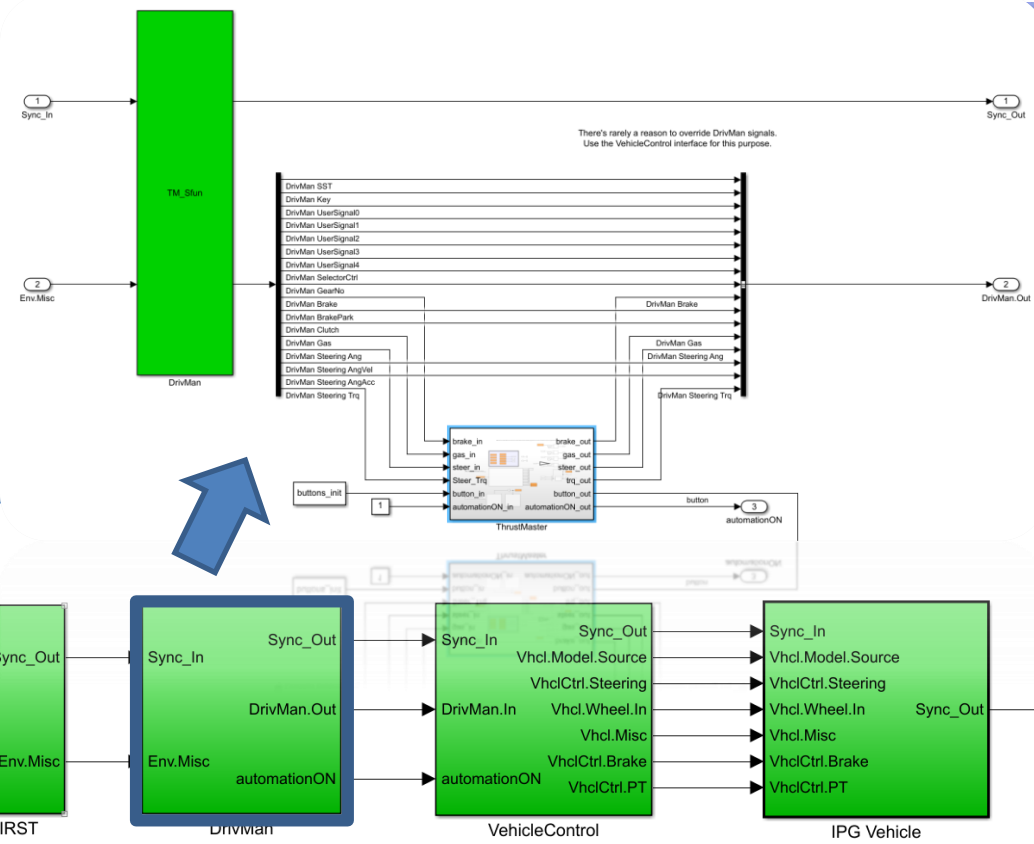
- **Initialization**
  - Connecting to the wheel
  - Initialization of Force-Feedback
- **Updating Block Data**
  - Retrieving Button/Pedal-Data
  - Sending Force-Feedback-Command
- **Termination**
  - Simulation ending: Termination of Force-Feedback

Content

- Force-Feedback and Windows
- Force-Feedback and MATLAB
- Force-Feedback and Simulink
- Force-Feedback and TruckMaker

Force-Feedback and TruckMaker





Force-Feedback and TruckMaker

Two modes were implemented:

- **Show-Case-Mode** for autonomous driving
  - Steering Input of the autonomous driving vehicle is emulated
  - Unidirectional communication
- **Force-Feedback** for manual driving
  - User input is calculated and applied to the steering rack of the vehicle
  - The vehicular response is than emulated on the steering wheel
  - Bidirectional communication

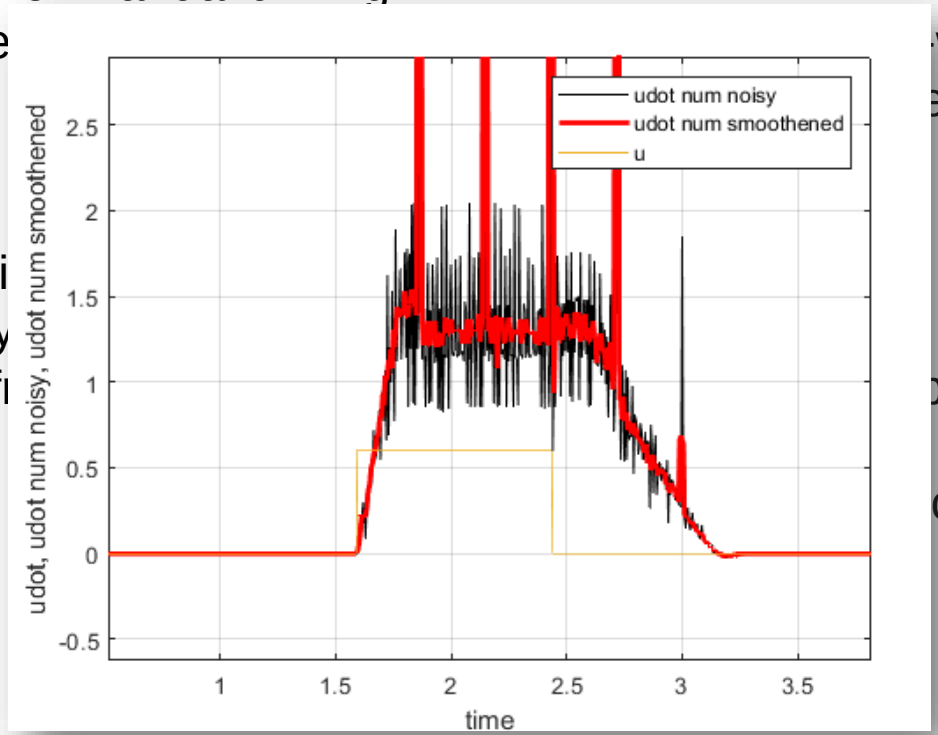
Force-Feedback and TruckMaker

Force-Feedback for manual driving:

For estimating the -wheel-system should be known. erties hindered a quality model:

- Dead-Zones in
- Stick-Slip-Fricti
- Unknown delay
- Velocity limits fi oed!

Simplest, and yet ch only consists of an inertia:

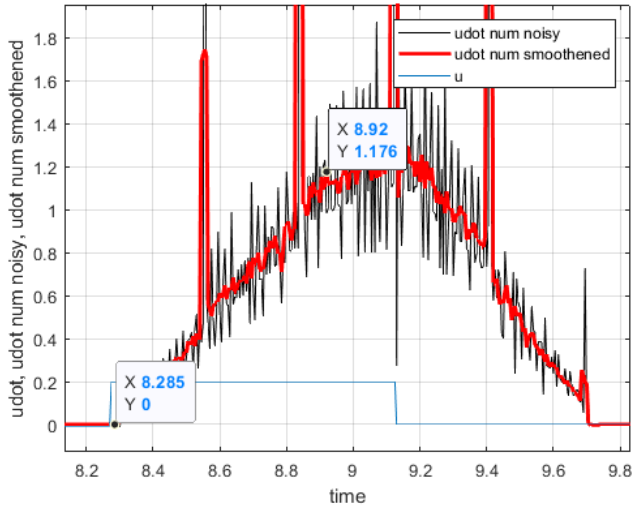


Force-Feedback and TruckMaker

Force-Feedback for manual driving:
Simplest, crude, and yet most effective model was later chosen, which only consists of an inertia:

$$J\frac{d^2\varphi}{dt} = J\frac{d\omega}{dt} = M_{motor} + M_{user}$$

Open-loop runs were used to fit the value of the inertia by hand:



$$\frac{d\omega}{dt} = \frac{1.179}{8.92 - 8.285} = 1.8567$$

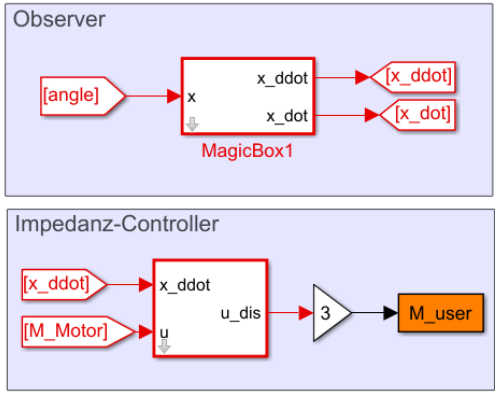
$$J = \frac{M_{motor}}{\frac{d\omega}{dt}} = \frac{0.2}{1.8567} = 0.1077$$

Force-Feedback and TruckMaker

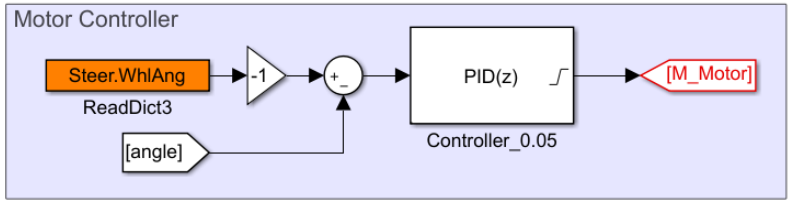
Force-Feedback for manual driving:

$$J\frac{d^2\varphi}{dt} = J\frac{d\omega}{dt} = M_{motor} + M_{user}$$

$$M_{user} = J\frac{d^2\varphi}{dt} - M_{motor}$$



M_{user} is applied to the steering rack in the TruckMaker simulation framework. The response to the user input, Steer.WhlAng, is fed back to an PID controller which is used to emulate the response on the wheel.



Conclusion

Conclusion

- Communication between the steering wheel and MATLB/Simulink/TruckMaker works flawlessly
- Show-Case-Effect works great as a demonstrational example
- Force-Feedback lacks a good model. Everything depends on a good model.

Model-Fitting-Challenge

As I would really like to improve the Force-Feedback experience I published some recorded data on confluence and annotated the datasets. People up to the task are encouraged to take a look at the data.

Thank you for your attention!