MATEC Web of Conferences **28**, 03001 (2015)
DOI: 10.1051/matecconf/20152803001

Tool-driven Design and Automated Parameterization for Real-time Generic Drivetrain Models

Christina Schwarz[1,a], Markus Bachinger[1], Michael Stolz[1] and Daniel Watzenig[2]

[1] Virtual Vehicle Research Center, Inffeldgasse 21a, 8010 Graz, Austria
[2] Institute of Electrical Measurement and Measurement Signal Processing, Graz University of Technology, Inffeldgasse 23/II A-8010 Graz, Austria

Abstract. Real-time dynamic drivetrain modeling approaches have a great potential for development cost reduction in the automotive industry. Even though real-time drivetrain models are available, these solutions are specific to single transmission topologies. In this paper an environment for parameterization of a solution is proposed based on a generic method applicable to all types of gear transmission topologies. This enables tool-guided modeling by non-experts in the fields of mechanic engineering and control theory leading to reduced development and testing efforts. The approach is demonstrated for an exemplary automatic transmission using the environment for automated parameterization. Finally, the parameterization is validated via vehicle measurement data.

1 Introduction

Replacing expensive vehicle tests by simulation has a high potential for reduction of cost and time during the development process. Furthermore, modern drivetrain layouts and customer requirements regarding driveability are crucial factors for introducing model-based drivetrain control algorithms. The importance of automatic drivetrain modeling has already been shown in [1]. Real-time simulation builds a bridge between the control strategy and the vehicle model through real-time feedback from the vehicle to the control strategy [2]. Therefore, a high demand exists for generic real-time models for different drivetrain topologies.

Friction elements ('clutches') play an important role in automotive geared transmissions. Clutches cause a variable order and structure system as the number of mechanical degrees of freedom depends on the clutch state ('slip' or 'stick') as well as the clutch torque dependency changes with the clutch state. The importance of generalized drivetrain models has been pointed out in [3], where a modeling method for such a variable order and structure system has been proposed. Another generic method was proposed in [4], additionally providing a solution for embedded execution.

Within this paper we present a novel tool, allowing a graphically-based arrangement of mechanic components applicable to all types of gear transmission. The tool provides the parameterization for drivetrain models based on the method in [4]. It is worth mentioning that the parameterization is valid for the clutch configuration with maximum number of mechanical degrees of freedom (i.e. all clutches slipping) for any linear modeling method.

2 Model description

Starting point for the development of a model design environment is the real-time drivetrain modeling approach shown in Figure 1 which will be briefly introduced. The model uses input torques **v,** provided by propulsion and load elements, and clutch slip torques $\boldsymbol{\tau}_s$.

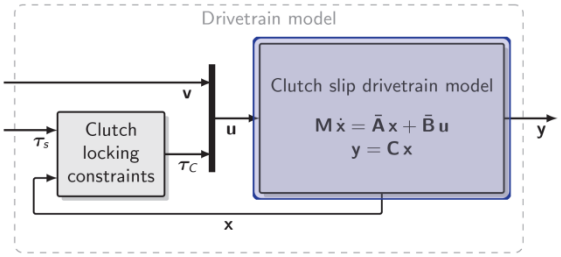

Figure 1. Block diagram of the drivetrain model in [4]

The clutch slip drivetrain model shown in Figure 1 only covers slipping and open clutches. Locked clutch torques are functions of internal states, external inputs and torques of slipping clutches. This behavior is modeled by dynamically extending the clutch slip drivetrain model depending on the specifically locked clutches in [3], while [4] considers locked clutch behavior via nonlinear feedback, as indicated in Figure 1. An approximation, that possible stiffness (k) and damping (d) factors are constant, is justified for real-time

[a] Corresponding author: Christina Schwarz, Christina.Schwarz@v2c2.at

execution limitations. This approximation allows representation of the clutch slip drivetrain model by a continuous-time state-space model using the following definitions: **x** summarizes the minimum number of required states consisting of rotational speeds of inertia elements and angular position differences of spring elements, **u** represents external inputs **v** and clutch torques $\boldsymbol{\tau}_c$ and **y** available sensor signals. The state-space model can be represented by

$$\mathbf{M}\,\dot{\mathbf{x}} = \bar{\mathbf{A}}\,\mathbf{x} + \bar{\mathbf{B}}\,\mathbf{u}, \quad \mathbf{y} = \mathbf{C}\,\mathbf{x}. \tag{1}$$

The drivetrain topology is completely defined by the parameters of the matrices $\bar{\mathbf{A}}$, $\bar{\mathbf{B}}$, **C** and **M** in the context of the present approach. The drivetrain model presented in [4] simulates generically with any model conforming to the description by these matrices. The formalism to obtain the parameters of the system matrices, states **x**, inputs **u** and outputs **y** is shown in Section "Pseudo Code". With the developed environment it is convenient to draw the schematic of a drivetrain topology and let the algorithm in the background derive the parameters of the system matrices automatically. This environment and the algorithm deriving $\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C}$ and **M** will be described in the following sections.

3 Tool-environment

The environment was developed in Java with Java Swing and is shown in Figure 2. It provides all components needed for the modeling of a drivetrain. As mentioned above distinct geared drivetrain topologies can be modeled including (automated) manual, dual clutch or automatic transmissions. The implementation of an improved torque converter will be subject to further investigations, possibly basing on the model description in [1]. After modeling the system matrices can be parameterized automatically and are saved within a MATLAB® initialization function.

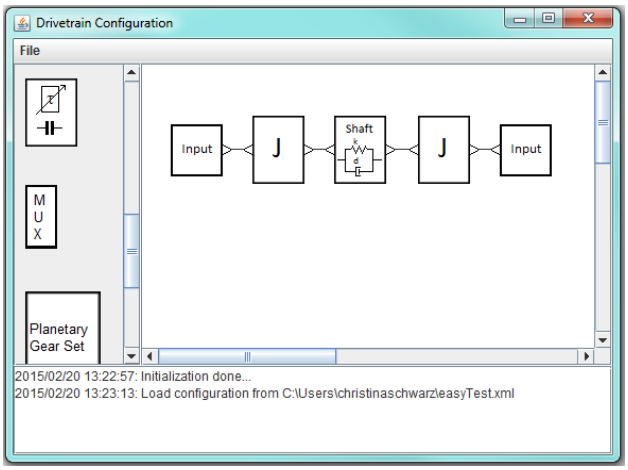

Figure 2. Environment for drivetrain model design with a simple example.

4 Implemented Components

Input. The input component represents an external input and is implemented as an interface.

Clutch/Brake. This component acts as brake or clutch, depending on the connected components The torque direction can be configured by the user, whereat the default direction is from left to right.

Inertia. The inertia can be configured as inertia or as mass in order to enable translational and rotational motion. If the inertia is configured as mass a lever arm has to be defined as well within the context of a drivetrain.

Shaft. This component serves to transmit torque and rotational speed to connect components of a drivetrain via stiffness (k) and damping (d).

Planetary gear sets. Planetary gear sets are essential in drivetrain development. The stationary gear ratios of each gear set are configurable by the user. The algorithm in the background has to ensure that the numbers of mechanical degrees of freedom and connected inertias fit. Commonly used gear sets (simple and combined) are implemented, an extension by any specific gear set is possible if the torque-relations are known.

Calculation of these torque-relations bases on the kinematic and kinetic constraints. The kinematic constraints are given by the WILLIS-Equations [5, 6], which can be set up for simple and combined gear sets. The kinetic constraints are defined by the equilibriums of power and external torques for a planetary gear set [7].

5 Systematic Approach for System Matrices Parameterization

For the calculation of the parameters it is essential to know how many states and inputs exist. The number of states can be determined by the number of inertias and shafts in a configuration, the number of inputs of the state-space model is defined by the number of external inputs and clutches. Since the inertia is involved in the identification of $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ it makes sense to take the inertias as starting points. Matrix $\bar{\mathbf{A}}$ maps the states with each other and $\bar{\mathbf{B}}$ maps the inputs with the states. The algorithm will iterate over each inertia and investigate the inputs on the left and the right side of it. First of all **C** is determined, since the outputs need to be mapped to the according inertia.

An exemplary configuration, as shown in Figure 2, serves to demonstrate the calculation of the matrices. The equations of motion for this example are defined by

$$\begin{aligned} J_1\dot{x}_1 &= -\tau_{shaft} + \tau_{in_1} \\ J_2\dot{x}_2 &= \tau_{shaft} - \tau_{in_2} \\ \dot{x}_3 &= x_1 - x_2 \\ \tau_{shaft} &= k\Delta\varphi_s + d\Delta\omega_s \end{aligned} \tag{2}$$

$$\tau_{shaft} = kx_3 + d(x_1 - x_2).$$

By transforming these equations (2), the system matrices should have the following entries granted that x_1 is the motion state of J_1, x_2 is the motion state of J_2 and the x_3 represents the position difference over shaft:

$$\mathbf{x} = [\omega_1 \quad \omega_2 \quad \varphi_1 - \varphi_2],$$

$$\mathbf{u} = [\tau_{in1} \quad \tau_{in2}],$$

$$\mathbf{M} = \mathrm{diag}([J_1 \quad J_2 \quad 1]) \tag{3}$$

$$\bar{\mathbf{A}} = \begin{bmatrix} -d & d & -k \\ d & -d & k \\ 1 & -1 & 0 \end{bmatrix},$$

$$\bar{\mathbf{B}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{C} = 0, \tag{4}$$

Pseudo Code. A simplified version of the algorithm only representing the components that are important for the example is presented in the following pseudo code. The torques of clutches and inputs can influence the states positive or negative, therefore $\{^{+}_{-}\}$ was inserted in the pseudo code. Furthermore, the variable *relationGS* is depending on the respective connectors.

```
enumerate Inertias, Shafts and init Ā
enumerate Inputs, Clutches and init B̄
enumerate Ouputs and init C matrix
for 1 to NumberOfInertias
    for 1 to NumberOfConnectors
        [Connection,Ratio]=
            calculateRatio(Connection)
        switch Connection
            case: Clutch
                B̄(idxInertia, idxClutch)={±} Ratio
            case: Input
                B̄(idxInertia, idxClutch)={±} Ratio
            case: GearSet
                for 1 to numberOfGearSetConnectors
                    [GearSetConnector, ratioGS] =
                       calculateRatio(GearSetConnector)
                    switch GearSetConnector
                        case: Clutch
                          B̄(idxInertia, idxClutch) =
                              {±}ratio*ratioGS*relationGS
                        case: Shaft
                        idxInertia2 =
                              findIdxOfSecondInertia()
                          Ā(idxInertia,idxInertia)=
                            -d*ratio*ratioGS*relationGS
                          Ā(idxInertia,idxInertia2) =
                            d*ratio*ratioGS*relationGS
                          Ā(idxInertia,idxSecInertiaGS) =
                            -d*ratio*ratioGS*relationGS
                         Ā(idxInertia,idxShaft) =
                              k*ratio*ratioGS*relationGS
            case: Shaft
                for 1:numberOfShaftConnectors
                    [ShaftConnection, ratioShaft] =
                        calculateRatio(ShaftConnection)
                    switch ShaftConnector
                        case: Inertia
                          Ā(idxInertia,idxInertia)=
                                -d*ratio*ratioShaft
                          Ā(idxInertia,idxInertia2)=
                                d*ratio* ratioShaft
                          Ā(idxInertia,idxShaft)=
                                k*ratio* ratioShaft
                        case: GearSet
                          idxInertia2 =
                                findIdxOfSecondInertia()
                          Ā(idxInertia,idxInertia)=
                            -d*ratio*ratioGS*relationGS
                          Ā(idxInertia,idxInertia2)=
                            d*ratio*ratioGS*relationGS
                          Ā(idxInertia,idxSecInertiaGS)=
                            d*ratio*ratioGS*relationGS
                          Ā(idxInertia,idxShaft)=
                            k*ratio*ratioGS*relationGS

for 1 to NumberOfShafts
    [ShaftConnection, ratioShaft] =
        calculateRatio(ShaftConnection)
    switch ShaftConnection
        case Inertia
            Ā(idxShaft,idxInertia)= {±}ratio* ratioShaft
        case GearSet
            for 1 to NumberOfGSConnectors
                [GearSetConnection, ratioGS] =
                    calculateRatio(GearSetConnection)
                if GearSetConnection is Inertia
                    Ā(idxShaft,idxInertia)= {±}ratio*
                        ratioShaft*ratioGS*relationGS

[Connection, ratio]=calculateRatio(Connection)
    if Connection is not Gear
        ratio=1
    else if Connection is Gear And TorqueDirection ==
        GearDirection
        ratio=Gear.ratio
        Connection=Gear.getConnection()
    else
        ratio=Gear.ratio^-1
        Connection=Gear.getConnection()
```

Pseudo Code 1: Simplified pseudo code for automated system matrices parameterization

6 Results and Validation

The tool for automated parameterization as well as the modeling approach in [4] are validaded on a production vehicle in a real-life driving cycle. A conventional 7-speed automatic trasmission shown in Figure 3 in a passenger car was available on a test track. This automatic transmission provides seven gears plus one reverse gear and comprises seven clutches/brakes. Figure 4 shows the drivetrain topology of the automatic transmission designed with the developed environment. Within the developed tool the vehicle mass can be directly considered. Hence, J_v is configured as vehicle mass with the wheel radius r_w being the lever arm.

The relations in (11) to (15) have been obtained with the help of computer algebra, solving kinetic and kinematic equations for the simple planetary gear set (PG3) and the combined one (PG1/2). Based on the algorithm described above the system matrices will be parameterized. After the definition of x and u the system matrices in (1) are parameterized automatically.

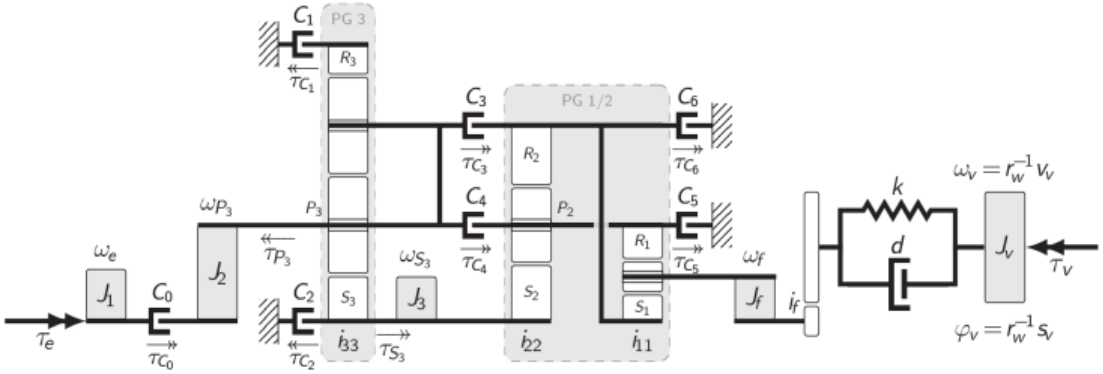

Figure 4. Diagram of a 7-gear automatic transmission in a conventional vehicle

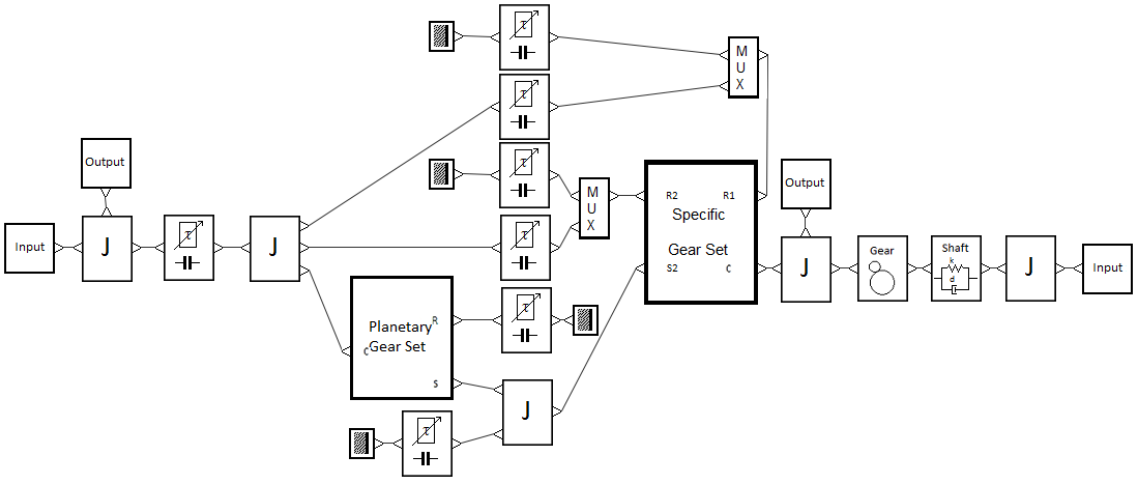

Figure 3. Scheme of automatic transmission in a conventional vehicle drawn in the developed environment

$$\mathbf{x} = [\omega_e \ \omega_{P_3} \ \omega_{S_3} \ \omega_f \ v_v \ i_f^{-1}\varphi_f - \varphi_v] \quad (5)$$

$$\mathbf{u} = [\tau_e \ \tau_v \ \tau_{C_0} \ \tau_{C_1} \ \tau_{C_2} \ \tau_{C_3} \ \tau_{C_4} \ \tau_{C_5} \ \tau_{C_6}] \quad (6)$$

$$\mathbf{M} = \mathrm{diag}([J_1 \ J_2 \ J_3 \ J_f \ M_v \ 1]) \quad (7)$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (8)$$

$$\bar{\mathbf{A}} = \begin{bmatrix} & \mathbf{0_{3x3}} & \\ \mathbf{0_{6x3}} & \begin{matrix} -i_f^{-2} d & i_f^{-1} d \, r_w^{-1} & -i_f^{-1} k \\ i_f^{-1} d \, r_w^{-1} & d \, r_w^{-2} & k \, r_w^{-1} \\ i_f^{-1} & -r_w^{-1} & 0 \end{matrix} \end{bmatrix} \quad (9)$$

$$\bar{\mathbf{B}} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & r_{R_3}^{C_3} & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & r_{S_3}^{C_3} & -1 & -r_{R_2}^{S_2} & -r_{R_1}^{S_2} & r_{R_1}^{S_2} & r_{R_2}^{S_2} \\ 0 & 0 & 0 & 0 & 0 & -r_{R_2}^{C_1} & -r_{R_1}^{C_1} & r_{R_1}^{C_1} & r_{R_2}^{C_1} \\ 0 & -r_w^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (10)$$

$$r_{R_3}^{C_3} = -\frac{-1+i_{33}}{i_{33}}, \ r_{S_3}^{C_3} = -\frac{1}{i_{33}}, \quad (11)$$

$$r_{R_1}^{S_2} = -\frac{i_{11}}{1-i_{22}-i_{11}i_{22}} \quad (12)$$

$$r_{R_2}^{C_1} = -\frac{i_{11}i_{22}-i_{22}}{1-i_{22}-i_{11}i_{22}} \quad (13)$$

$$r_{R_2}^{S_2} = -\frac{1}{1-i_{22}-i_{11}i_{22}}, \quad (14)$$

$$r_{R_1}^{C_1} = -\frac{i_{11}i_{22}-i_{22}-i_{11}-1}{1-i_{22}-i_{11}i_{22}} \quad (15)$$

The tool output, a Matlab® initialization file, was used to initialize the generic model code, available in Matlab/Simulink®. An offline simulation of a real driving cycle was performed. A comparison with the vehicle measurement is depicted in Figure 5. The comparison reveals that the simulation model represents well the behavior of the overall vehicle including different maneuvers and gear shift sequences. Moreover, the automated parameterization is fully representing the real drivetrain, while the existing differences between simulation and measurement data traced back to the parameterization derivations of peripheral models (like clutch hydraulics).

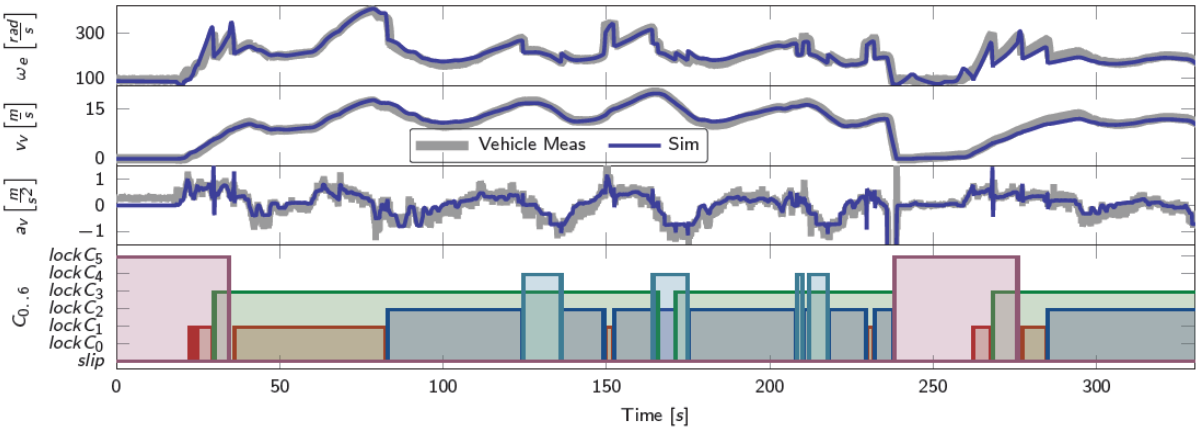

Figure 5. Vehicle measurement data of a driving cycle on a test track is used for validation of the modeling solution. The diagram illustrates the rotational speed of the engine ω_e, the vehicle velocity v_v and acceleration a_v as well as the clutch states $c_{0..6}$. The deviation of the simulation can be justified as model parameters were directly taken from mechanical design (instead of fitted to measurement). Note that a_v is very noisy and the sensor suffers from a small bias.

7 Conclusion

Existing approaches for generic drivetrain modeling base on a model with full mechanical degree of freedom (all friction elements slipping) and differently deal with handling of locking elements. Modeling of this *clutch slip drivetrain model* requires expert know-how in mechanical engineering. The proposed tool for automated parameterization enables non-experts to design drivetrain topologies and derive the models for these and reduces possible sources of errors during manual modeling. The tool was successfully used for several drivetrain topologies and in each case verified with the model code in [4]. An example for tool validation via vehicle measurement data was given.

Model Parameter

Var	Parameter	Value	Unit
J_1	Inertia engine, C_0 primary	0.098	[kg m^2]
J_2	Inertia C_0 secondary	0.0124	[kg m^2]
J_3	Inertia	0.023	[kg m^2]
J_f	Lumped inertia gearbox secondary, side shafts	1.32	[kg m^2]
M_v	Mass of vehicle	1380	[kg m]
k	Shaft stiffness gearbox secondary to vehicle	4000	[Nm/ rad]
d	Shaft damping gearbox secondary to vehicle	200	[Nms /rad]
i_{11}	Stationary gear ratio between S_1 and R_1	-1.64	
i_{22}	Stationary gear ratio between S_2 and R_2	-2.08	
i_{33}	Stationary gear ratio between S_3 and R_3	2.19	
i_f	Gear ratio final drive	3.61	
r_w	Dynamic wheel radius	0.280	[m]

Acknowledgments

VIRTUAL VEHICLE Research Center is funded within the COMET – Competence Centers for Excellent Technologies – programme by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry of Science, Research and Economy (BMWFW), the Austrian Research Promotion Agency (FFG), the province of Styria and the Styrian Business Promotion Agency (SFG). The COMET programme is administrated by FFG.

We would furthermore like to express our thanks to our supporting industrial and scientific project partners, namely AVL List GmbH and Graz University of Technology.

References

1. S. Zhao, R. Guo, L. Xu and X. L. Guo, "Modeling and Simulation of the Automatic Transmission Assembly Using Maltab/ Simulink," Applied Mechanics and Materials, pp. 291-294, 2013.
3. L. Xudong, F. Qingwu, Z. Banggui and D. Jianmin, "Real-Time Simulation Study for a Series Hybrid Electric Vehicle," Applied Mechanics and Materials, pp. 128-129,965, 2011.
5. M. Herchenhan, "Modellierung, Mehrfachregelung und optimale Steuerung eines leistungsverzweigten hybriden Antriebs", Technische Universität Braunschweig, Dissertation, 2009.
7. M. Bachinger, M. Stolz and M. Horn, "Fixed time-step drivetrain observer for embedded automotive applications," IEEE Conference on Control Applications (CCA), pp. 47-52, 2014.
9. H. W. Müller, Epicyclic Drive Trains: Analysis Synthesis, and Application, -: Wayne State University Press, 1982.
11. B. Paul, Kinematics and Dynamics of Planar Machinery, University of Michigan: Prentice-Hall, 1979.
13. F. Kurth, "Efficiency Determination and Synthesis of Complex-Compound Planetary Gear Transmissions," Dissertation, Technische Universität, München, 2012.