Electronic Stability Program

Development and verification

Development: Federico Galli

Validation: Alessandro Luppi

Last revision on 22/06/2020

# Introduction

The electronic stability program (ESP) is a subsystem of the complete all-wheel-drive electric vehicle under development. Its aim is to produce a wheel speed variation, for each of the four wheels, which will be added to the wheel speed given by the virtual differential. The ESP controller will be implemented by an electronic control unit actively controlling actuators which will drive the wheels’ rotation.

The objective of this dissertation is to introduce the ESP control model, and how it has been developed and verified.

In the following there will be introduced:

* the requirements of the ESP subsystem which drove the design;
* the development process;
* use and interface of the developed subsystem;
* the verification and validation processes.

## System requirements

In order to interface with the model, the following software modules are required:

* MATLAB R2019b and Simulink

No additional libraries are required.

# Design requirements

The ESP control model must provide the speed variations for the four wheels of the all-wheel-drive electric vehicle.

A list of the requirements which drove the design is reported:

* The ESP control must provide the variation of the speed of all four wheels of the electric vehicle.
* The variations must be compatible with wheel speeds provided by the virtual differential subsystem.
* The wheel speed variations are intended to counteract possible deficiencies of the vehicle performance due to non-ideal conditions of the road/track (wet asphalt because of rainy conditions, etc.).
* The ESP is allowed to take advantage of some of the vehicle’s signals generated by in-vehicle sensors or by other electronic control units, such as throttle pedal position and steering angle.

# Model-based design

The design of the ESP subsystem has been based on the vehicle dynamics: it has been obtained by integration of the kinematics equations, the inertial accelerations and the rotative dynamics of the vehicle with the expression of the forces (aerodynamics and wheel forces) acting on it. The system we get can then be expressed as a generic nonlinear system; of course, a linearization is needed in order to obtain a linear system in state x represented by slip angle β and yaw rate ωz, disturbance w represented by steering angle δw, and control u represented by the four wheel speed variations dω1,2,3,4. Thus, linearized matrices of the system are computed. The optimization function is defined through the two matrices Q and R, which carry information about the maximum allowable error on the sate and on the control, respectively. Then, the gain matrix Klqr is computed by making use of the MATLAB *lqr* function, which takes as input linearized system matrices A and B2, and the optimization function matrices Q and R.

The assumptions for the design are:

* Front and rear axles have equal length, and wheels have the same dimension (radius); these parameters are known, as well as the distance between front and rear axles.
* There are sensors providing the instantaneous angular position of the two front wheels.
* Linearization is performed around a straight trajectory, so linearization point is
* Total slip λTi is kept close to 0 so that it is possible to linearize the friction coefficient μ.

These assumptions conform with the other developed systems for the all-wheel-drive electric vehicle. The integration of the ESP subsystem with other vehicle’s systems introduce further complexity and eliminates some of the idealized assumptions.

# Usage

The ESP control model may be used whenever the ideal speed variations of the four wheels of a vehicle have to be determined in order to address the lack of performance of the vehicle due to slippery condition of the road. The speeds are ideal as they are calculated on a basis of specific assumptions and a vehicle implementing a steering mechanism compliant with Ackerman steering geometry.

In order to use the ESP control model, some variable parameters, usually provided by in-vehicle sensors or electronic control units, and some fixed parameters, known from car geometry, must be available. These are described in more details below.

## System interfaces

The ESP control model requires 3 input signals:

* **SlpAng** – scalar (rad)
  + Slip angle is a scalar because we are assuming that each wheel has the same slip angle β. The slip angle is the angle of the orientation of the wheel reference system with respect to the body reference system of the vehicle.
  + Values are in radians; allowed values range from to .
* **YawRte**– scalar (rad/s)
  + Yaw rate ωz of the vehicle is a scalar. The yaw rate is the angular velocity of its rotation around the z-axis of the body reference frame.
  + Values are in radians per second; allowed values range from to 4π.
* **CrnSpd** – scalar (m/s)
  + Real scalar number.
  + Current speed the vehicle is travelling at. It is supposed to be constant and equal to the reference speed for the purpose of this system.
  + Values are in meters per second, ranging from to . Negative values refer to vehicle moving backward.

The virtual differential model provides 1 output signal:

* **dWhlSpd** – 4x1 vector (rad/s)
  + Vector composed of four real scalar elements.
  + Each element reports the wheel angular speed variation of a wheel of the vehicle. Wheels’ speed variations are, in order: front-left, front-right, rear-left, rear-right.
  + Values are in radians per second, ranging to . Negative values refer to wheels rotating in reverse direction, that is when vehicle is backward moving.

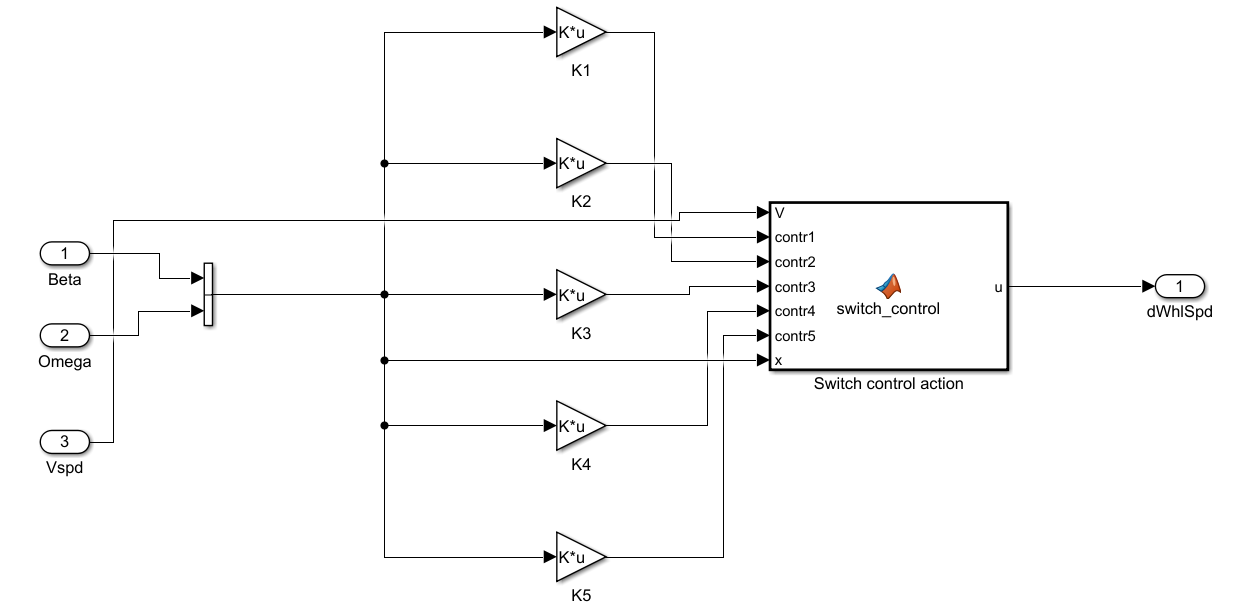
## System parameters

The ESP control model requires vehicle parameters:

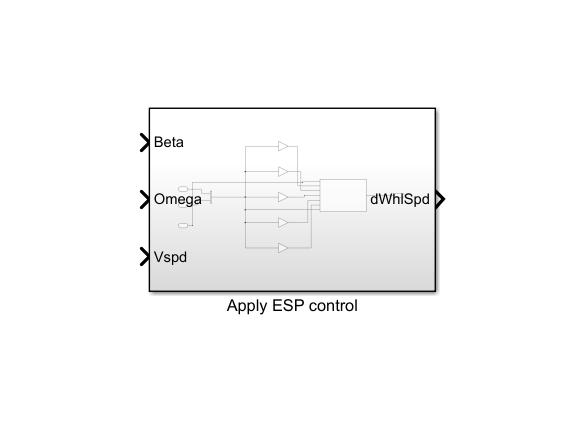
* **wheel\_base** – scalar (m)
  + Real positive scalar number.
  + It represents the distance between front and rear axles of the vehicle.
  + Values are in meters, ranging from to .
* **wheel\_distance** – scalar (m)
  + Real positive scalar number.
  + It represents the distance between wheel’s center of rotation and the center of the front/rear axle, assumed to be equal for all the four wheels.
  + Values are in meters, ranging from to .
* **wheel\_radius** – scalar (m)
  + Real positive scalar number.
  + It represents the radius of the wheels, assumed to be equal for all the four wheels. It is the distance between wheel’s center of rotation and a planar ground neglecting elastic effects of wheel.
  + Values are in meters, ranging from to .
* **vehicle\_mass** – scalar (kg)
  + Real positive scalar number.
  + It represents the mass of the vehicles.
  + Values are in kilograms, ranging from to .
* **aerodynamic\_coefficients** – 3x1 (-)
  + Three real positive scalar numbers.
  + It consists of the cross-surface S of the vehicle, and the two drag coefficients Cx0 and Cyb. They are used to compute the aerodynamic force the vehicle is subject to.
  + Cross-surface is in square meters, ranging from m2 to m2; drag coefficients are adimensional, ranging from to 1.

## How to use

To use the ESP control model, input signals and parameters must be provided as explained above. Here an example of use is shown.



Five-scenario control actions are precomputed and logged onto the system as Klqr1,2,3,4,5: these values are calculated at five current-velocity values (50,70,90,110 and 130 km/h). Then, based on the actual current velocity of the vehicle (*Vspd* in the schematic), the *switch\_control* block selects the most suitable control action to the situation (among the possible five options it takes as inputs, *contr1,2,3,4,5*). Moreover, the block performs also a check on the two state variables *beta* and *omega*, slip angle and yaw rate of the vehicle: if the two variables are out of range (described above, in the list of input variables of the subsystem), then the ESP control model does nothing; otherwise, it chooses the best control action and gives as output the wheel-speed variations of the four wheels to counteract the slippery conditions of the road. An unwrapping of the slip angle *beta* in range [-π;+π] is also provided.



# Validation

{TODO: describe verification process (i.e. does it satisfy requirements?)}