Extending and customizing CarSim math models at runtime

Michael Sayers, Ph.D.
CEO and Chief Technology Officer

- Core model capabilities
- Extending the models
- Runtime VS commands
- Q & A
The Parts of CarSim

- Use the database to define vehicles, conditions, and test results
- One click to make a run
- One click to view animation
- One click to view engineering plots
- Export results to other software
Timeline

UMTRI: University of Michigan Transportation Research Institute (formerly HSRI)
• 1960’s Vehicle dynamics research
• 1970’s Early vehicle and tire models
• 1989 Automated modeling (AutoSim)
• 1990 Simulation GUI
• 1995 TruckSim

Mechanical Simulation Corporation
• 1996 CarSim
• 1998 Real-time Hardware in the loop
• 1999 Simulink support
• 2002 High-quality animation
• 2005 BikeSim, event programming
• 2007 VS commands, VS API
Worldwide Customers

- 30+ Car and Truck OEMs
- 60+ Tier 1 and Tier 2 Suppliers
- 120+ Universities, Testing and Research Organizations
Simulink can access CarSim math models through S-Function blocks.

- Use Simulink from CarSim
- Use CarSim from Simulink

Run CarSim with Simulink
**Host Machine w. Windows**
- Database
- Animator & Plotter
- User Interface

**Target Machine w. RT OS**
- CarSim math models
- Hardware-in-the-loop interface
- Works with most RT systems

---

**CarSim RT for Hardware in the Loop**
Driving Simulators

- “Feel” design and/or HIL
- Reproduce established tests
- CarSim used “as is” for 70+ driving simulators
- RT animation for engineers
- CarSim RT used for huge two-track Toyota Simulator
**CarSim and Product Life Management**

- Compress design cycles
- Optimize physical testing
- Collaboration

**Vehicle Testing**
- Test with Hardware in the Loop

**Component Testing**
- Test with Software in the Loop

**Controls Development**
- Test with Software in the Loop

**System Definition**
- Simulate with CarSim

**Vehicle Definition**
- Vehicle Requirements, Capabilities, Capacities

**Marketing Tools**
- Proving Ground Optimization, Driving Simulators

**Aftermarket**
Many Applications

- 1000+ CarSim licenses (many on networks)
  - Vehicle design and testing at OEM and tier-1 (mechanical engineers)
  - Controller design and testing (electrical engineers)
  - Evaluation by specialists (brakes, powertrain, tires, steering)
  - Testing of aftermarket vehicle modifications
  - Research by scientists
  - Education (vehicle dynamics, control)
  - Driver training and human factors research (driving simulators)
  - Road design
  - Marketing
  - Accident analysis and reconstruction
  - ...

- A single vehicle model is not perfect for everyone
A CarSim vehicle model

- Core model for vehicle dynamics
  - Nonlinear 3D kinematics and dynamics from symbolic multibody program
  - Built-in models for standard systems (brakes, powertrain, tires, steering)
  - Comprehensive 3D road model
  - Closed-loop controls for basic driver actions

3D multibody vehicle description (Lisp) → VehicleSim Lisp (old name: AutoSim) → Machine-generated C Code → C compiler → CarSim Solver DLL → Developers (Mechanical Simulation) → Users

- Hand-written C code for vehicle systems and components
- VehicleSim library routines (API, commands, file i/o, etc.)
VehicleSim Lisp Multibody Code Generator

Example: 3D suspension/steering kinematics

- Extensive and fully nonlinear
- Highly optimized (but lengthy!)
- Much faster than real-time
Runtime Table Options

- Table interpolation determined at run time
- Most tables have offset and gain options
  - Support sensitivity, DOE studies
  - Define customized control functions
A CarSim vehicle model

- Core model for vehicle dynamics
  - Nonlinear 3D kinematics and dynamics from symbolic multibody program
  - Built-in models for standard systems (brakes, powertrain, tires, steering)
  - Comprehensive 3D road model
  - Closed-loop controls for basic driver actions

- The core vehicle model can be a block in Simulink, LabView, ETAS ASCET
Full Import Options

- Combine with “native variable” by add, replace, multiply
- Machine-generated documentation (Excel)
A CarSim vehicle model

- Core model for vehicle dynamics
  - Nonlinear 3D kinematics and dynamics from symbolic multibody program
  - Built-in models for standard systems (brakes, powertrain, tires, steering)
  - Comprehensive 3D road model
  - Closed-loop controls for basic driver actions

- The core vehicle model can be a block in Simulink, LabView, ETAS ASCET

- The model can be controlled by other software using the VehicleSim API
VehicleSim API

- One set of DLLs
  - VehicleSim
  - Browser (SGUI)
  - Simulink
  - LabView
  - ASCET
  - Custom EXE
  - More…

- VS API provides 3D road, tire
- Extend with custom C/C++
VS API: You control the simulation

- Load DLL
- Start (read data)
- Loop (integrate)
- Terminate

VS Solver DLL

SGUI

Wrapper EXE program
Custom Code

- External variables
- Install in model
- External equations

---

```c
/* Set up variables for the model extension. For the steering controller
define new units and parameters and set default values of the parameter
that will be used if nothing is specified at run time. */

void external_setdef (void)
{
    int idX, idY;

    // set default values for parameters defined in this file
    sUseExternal = 0;
    sLfd = 20.0;
    sGainStr = 10.0*PI/180.0;
    sLatTrack = -1.6;

    // C version of VS command: define_units deg/m 57.2957951
    gAPI.vs_define_units ("deg/m", 180.0/PI);

    // define two new parameters: L_FORWARD and LAT_ERR
    gAPI.vs_define_parameter ("L_FORWARD",
            "Distance preview point is forward of vehicle
            and "L_FORWARD", "M");
    gAPI.vs_define_parameter ("LAT_TRACK",
            "Lateral offset (to driver's left) for target
            and "LAT_TRACK", "M");

    case VS_EXT_EQ_IN: // calculations at the start of a time step
        
        // calculate X and Y coordinates of preview point
        sxprev = *sxcg + sLfd*cos(*sYaw);
        syprev = *sycg + sLfd*sin(*sYaw);

        if (!sUseExternal) ; // no effect if sUseExternal is FALSE
        else if (t <= *s?start) *sImpStr = 0.0; // no steering at the start
        else // steer proportional to the lateral error
            *sImpStr = sGainStr*(sLatTrack - gAPI.vs_road_1(sXprev, sYprev));
        break;
```
A CarSim vehicle model

- Core model for vehicle dynamics
  - Nonlinear 3D kinematics and dynamics from symbolic multibody program
  - Built-in models for standard systems (brakes, powertrain, tires, steering)
  - Comprehensive 3D road model
  - Closed-loop controls for basic driver actions

- The core vehicle model can be a block in Simulink, LabView, ETAS ASCET

- The model can be controlled by other software using the VehicleSim API

- The core model can be extended at runtime with VS commands
  - Use events for changing controls or vehicle properties
  - Add new variables and equations (algebraic and differential)
  - Redefine forces, moments, and controls in the core model
VehicleSim Commands

- Only a few commands
- Yet, powerful options
Outrigger Example

- Add 4 equations
- Enable 4 imports
- Use existing reference points and forces (7,8,9,10)
- Add 1 variable (parameter) K_RIG
- Define new units: N/mm (gain = 0.001)

VS Commands Example
ESC Test: FMVSS 126

- Run tests to find steer for $A_y = \pm 0.3$ g
- Run series of “sine with dwell tests”
- Compare yaw rate at two times to peak yaw rate
- Check lateral position
- Test until steer > 270°

**VS Commands: Example ESC Test**
Failure: Yaw rate > 35% peak
VS Commands extend the model

- Add new variables
- Add equations
- Define events
CarSim Has the Core Vehicle Model

Extend the core model as needed

- VS Commands
- Simulink, LabView, ASCET
- Custom C
- RT with HIL
- Driving Simulator