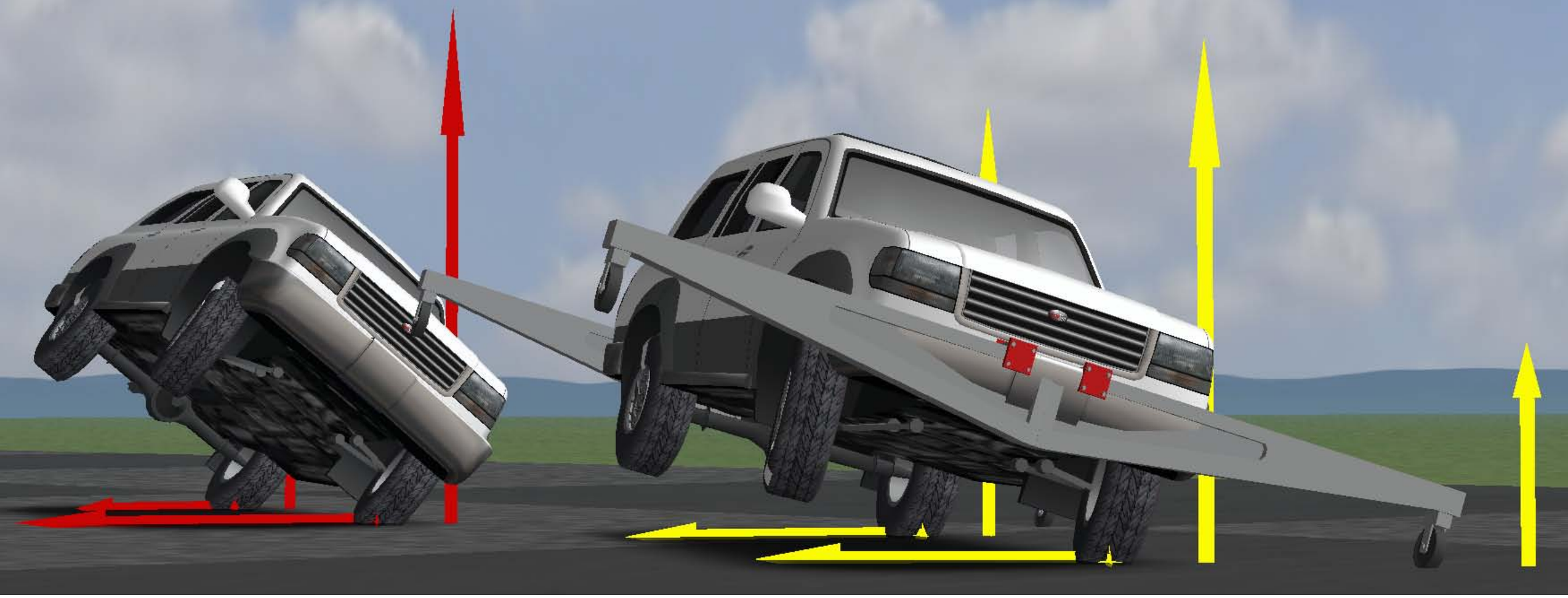
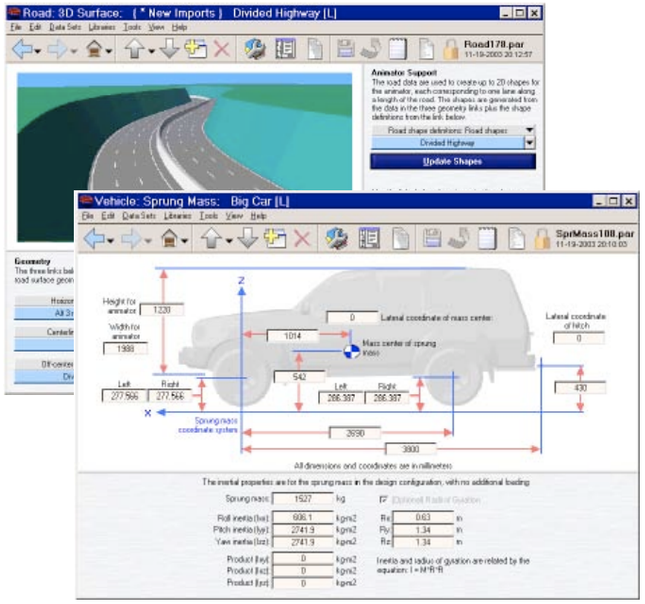


Extending and customizing CarSim math models at runtime

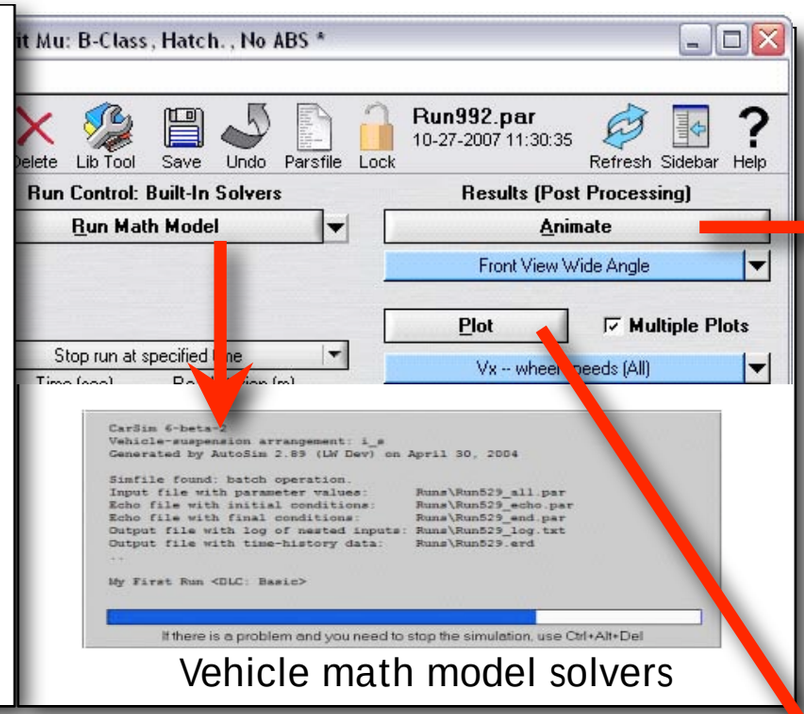
Michael Sayers, Ph.D.

CEO and Chief Technology Officer

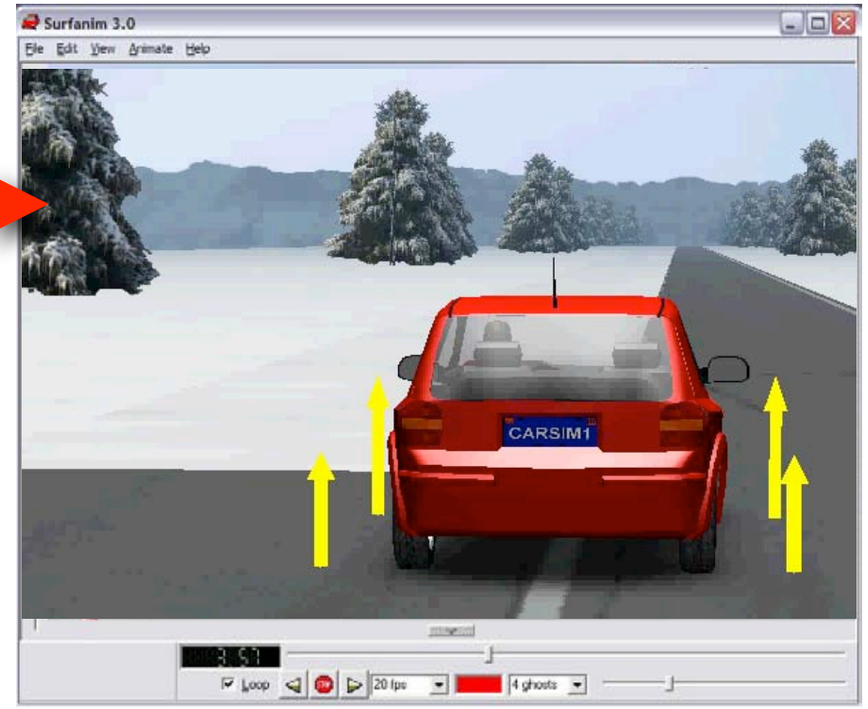
- Core model capabilities
- Extending the models
- Runtime VS commands
- Q & A



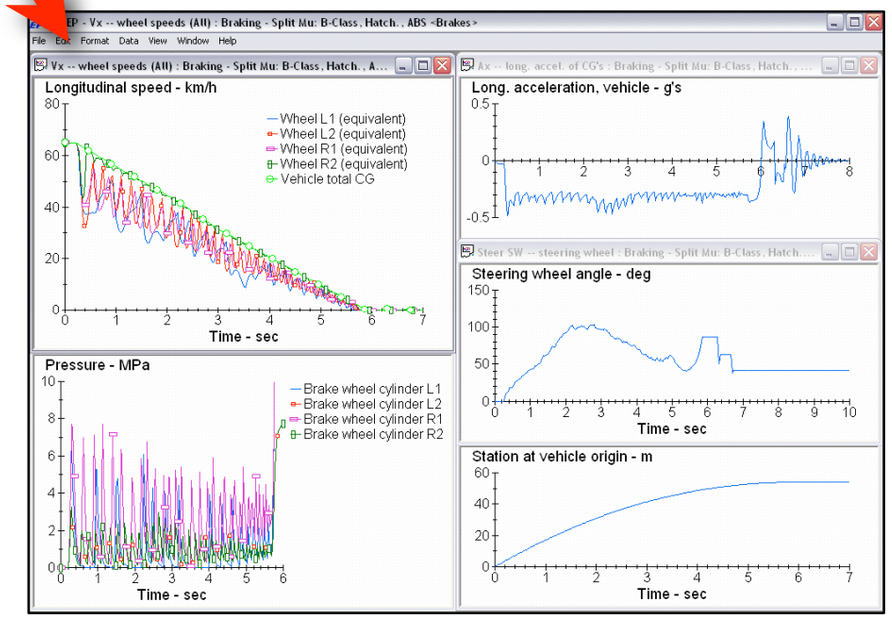
Graphical database



Vehicle math model solvers



Surface animator



Engineering plotter

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

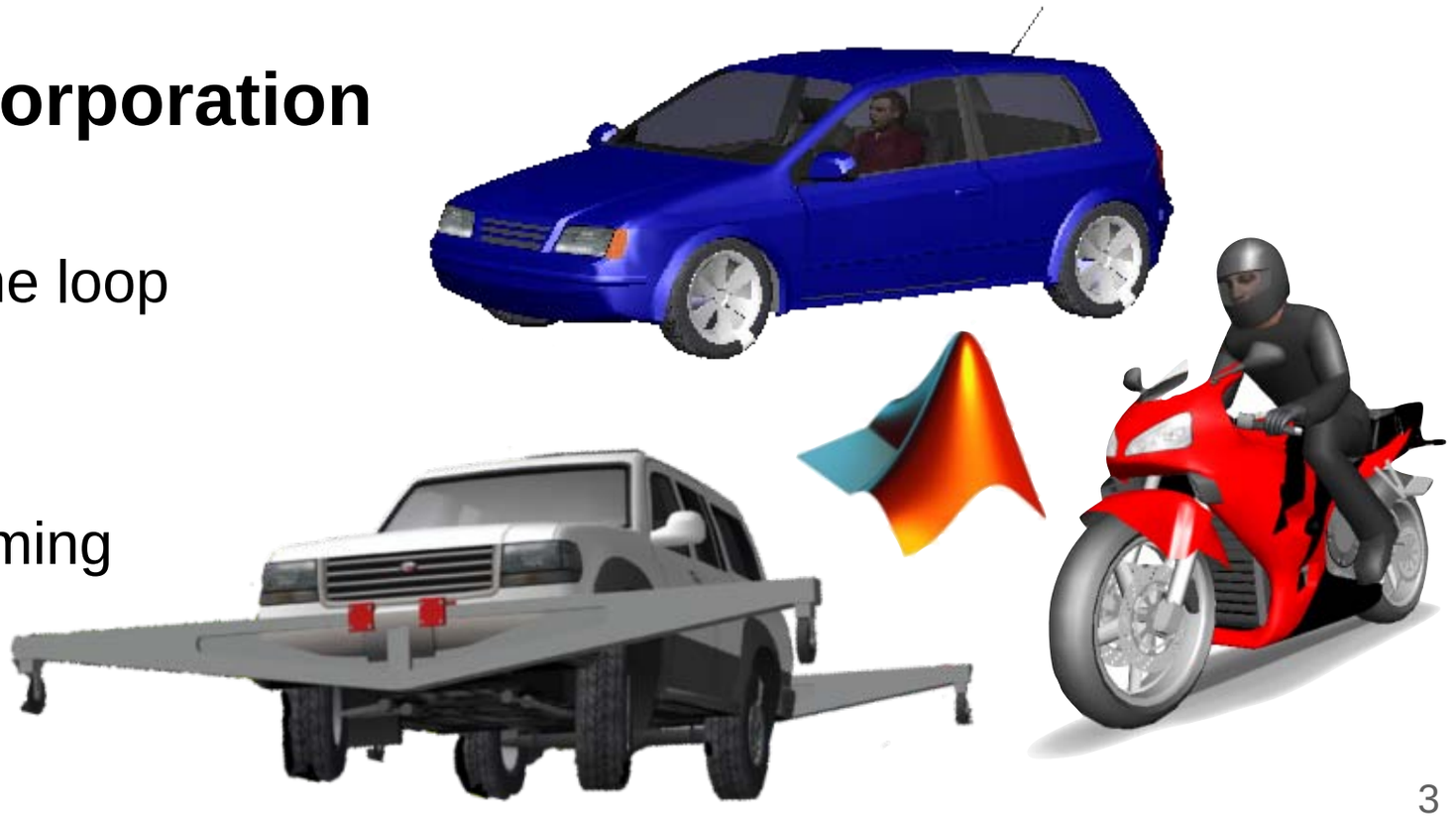
Timeline

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

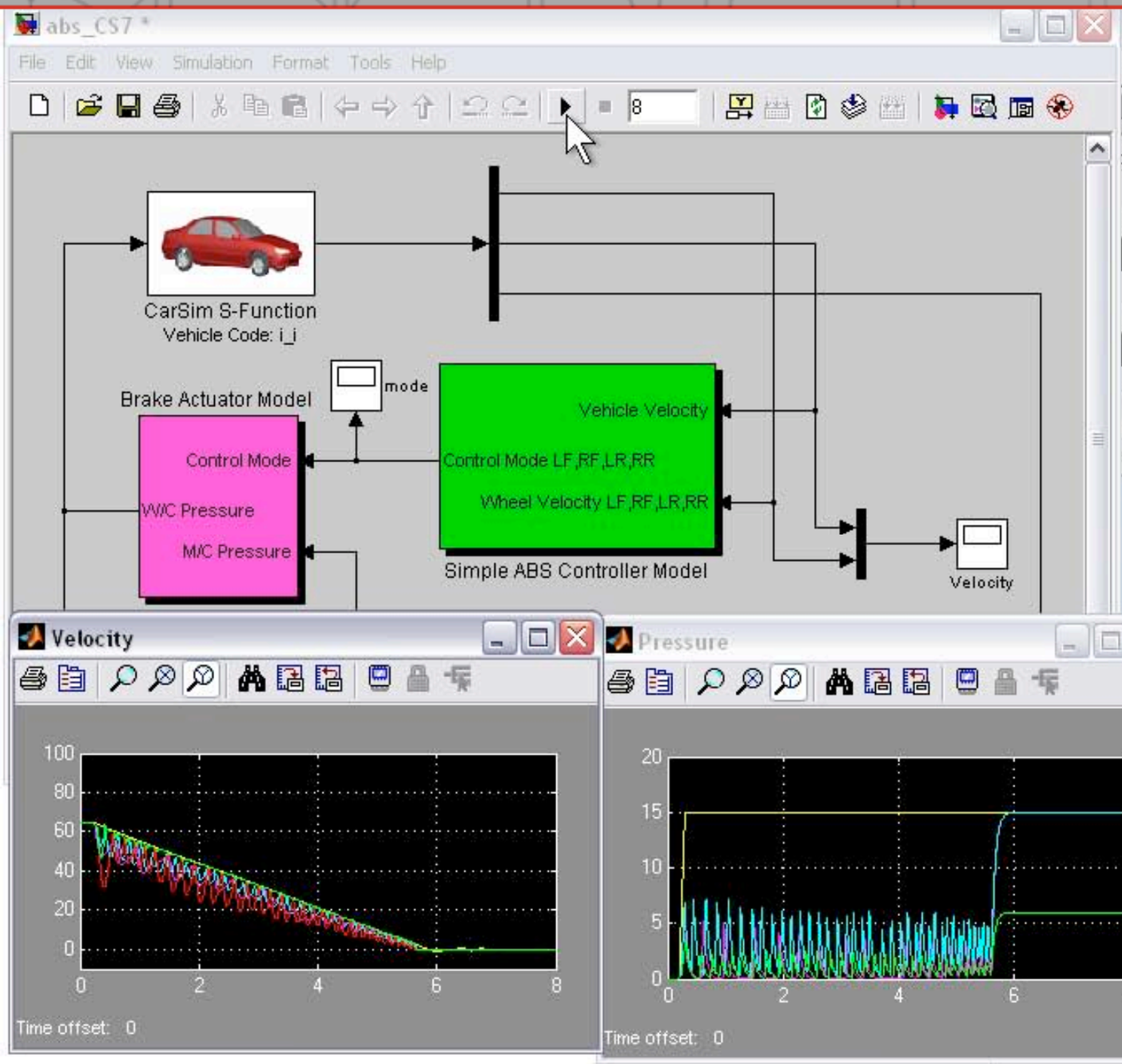
- 1996 CarSim
- 1998 Real-time Hardware in the loop
- 1999 Simulink support
- 2002 High-quality animation
- 2005 BikeSim, event programming
- 2007 VS commands, VS API

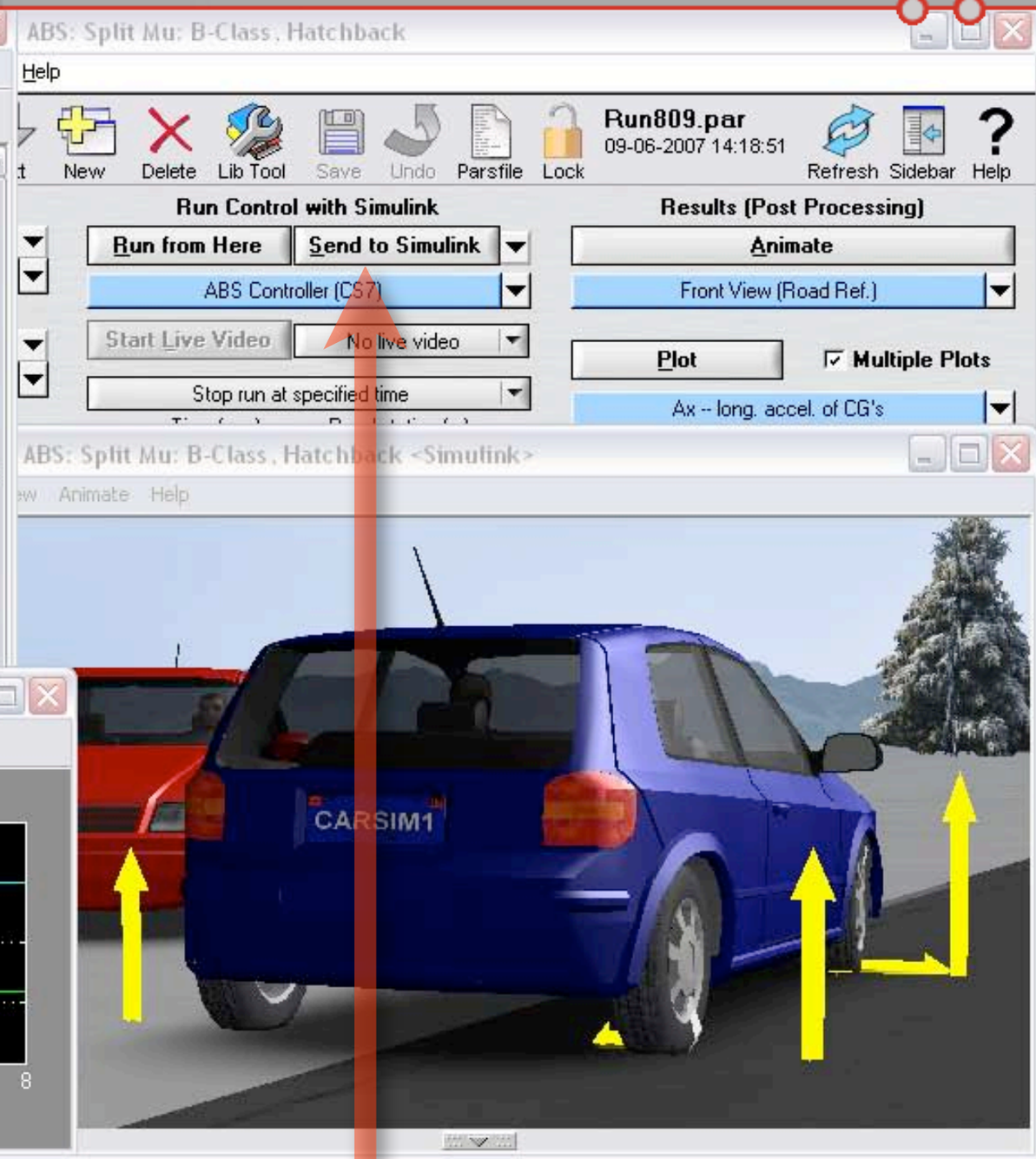


Worldwide Customers

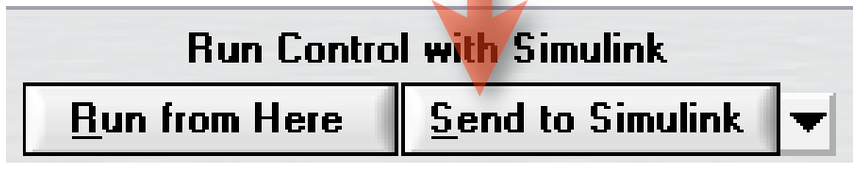
- 30+ Car and Truck OEMs
- 60+ Tier 1 and Tier 2 Suppliers
- 120+ Universities, Testing and Research Organizations

Mechanical Simulation





- Simulink can access CarSim math models through S-Function blocks
- Use Simulink from CarSim
- Use CarSim from Simulink



Run CarSim with Simulink

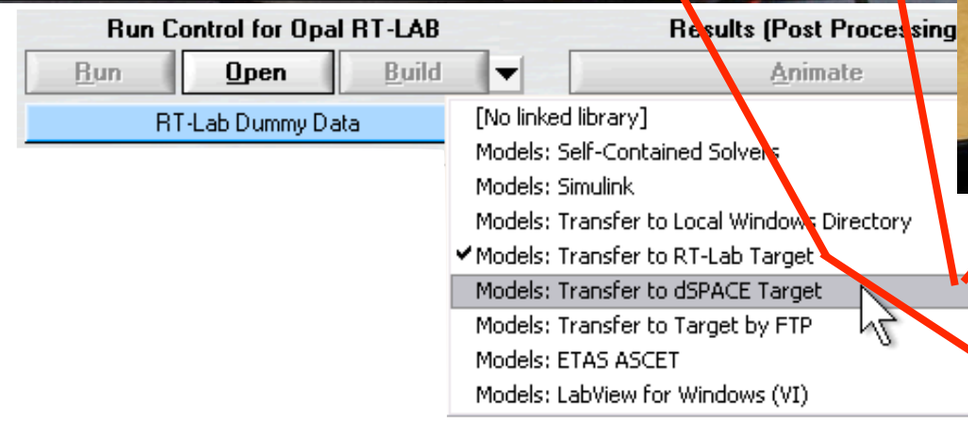
Host PC Windows

RT Lab

Choose RT Platform

LabView RT

dSPACE



- 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

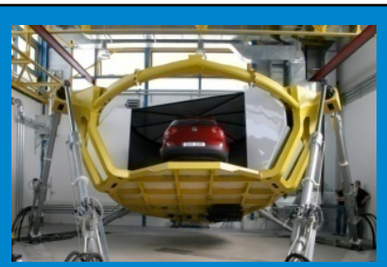
CarSim and Product Life Management

- Compress design cycles
- Optimize physical testing
- Collaboration

Product Launch

Marketing Tools

Vehicle Testing

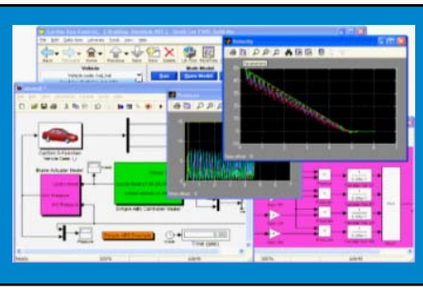


Proving Ground Optimization, Driving Simulators

Component Testing

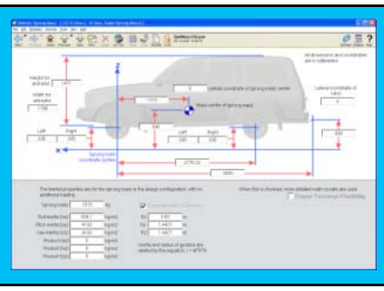
Test with Hardware in the Loop

Controls Development



Test with Software in the Loop

System Definition



Simulate with CarSim

Vehicle Definition

Vehicle Requirements, Capabilities, Capacities

Aftermarket

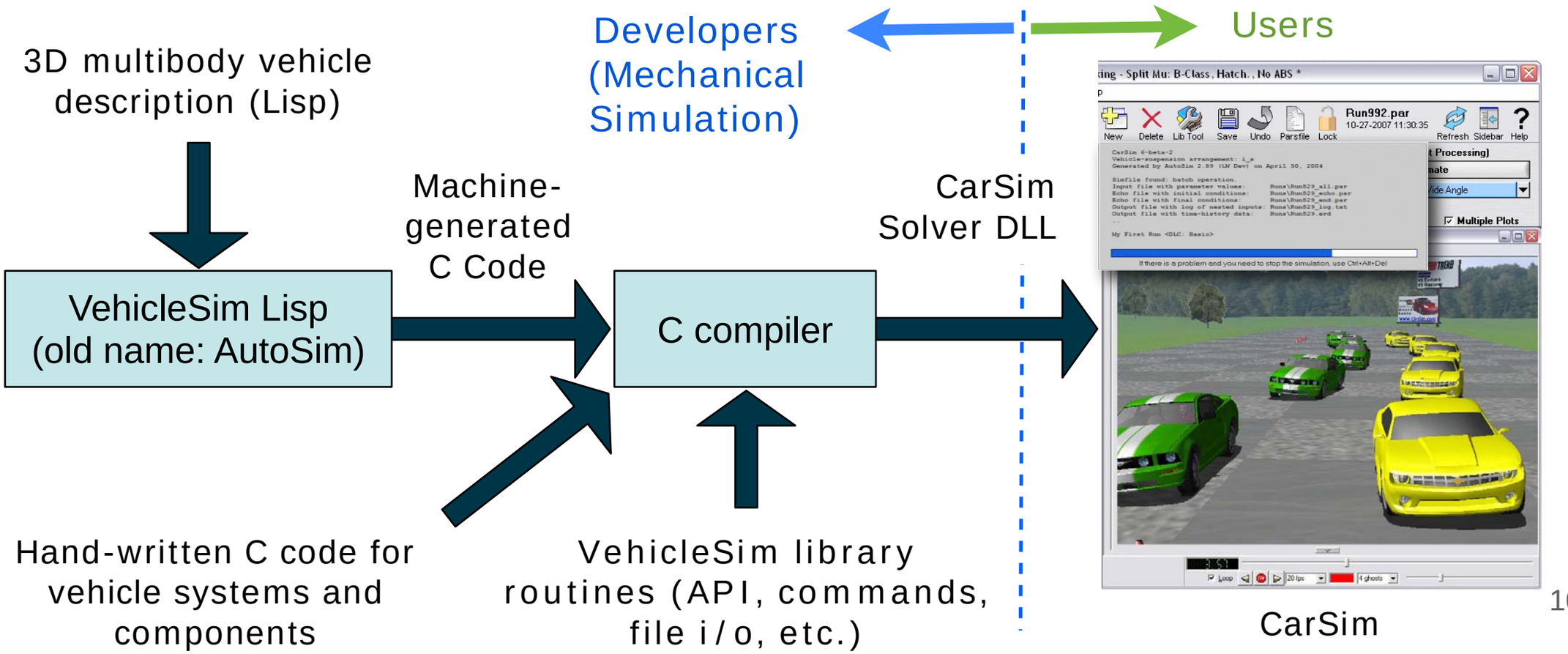
Many Applications

- 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



VehicleSim Lisp Multibody Code Generator

vsm_difeqn.c - Microsoft Visual Studio

File Edit View Debug Tools Window Community Help

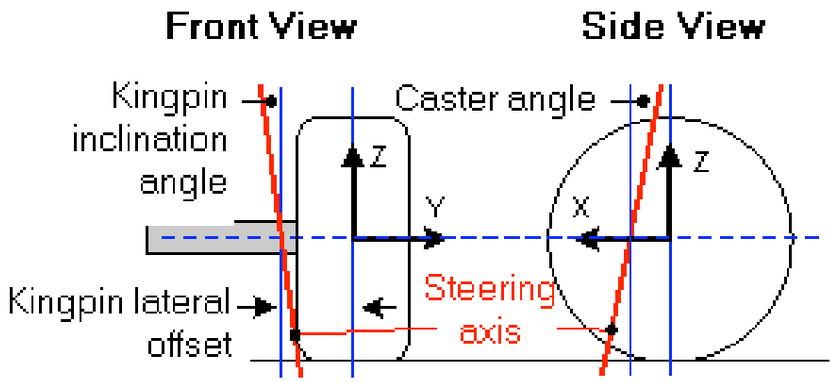
vsm_difeqn.c

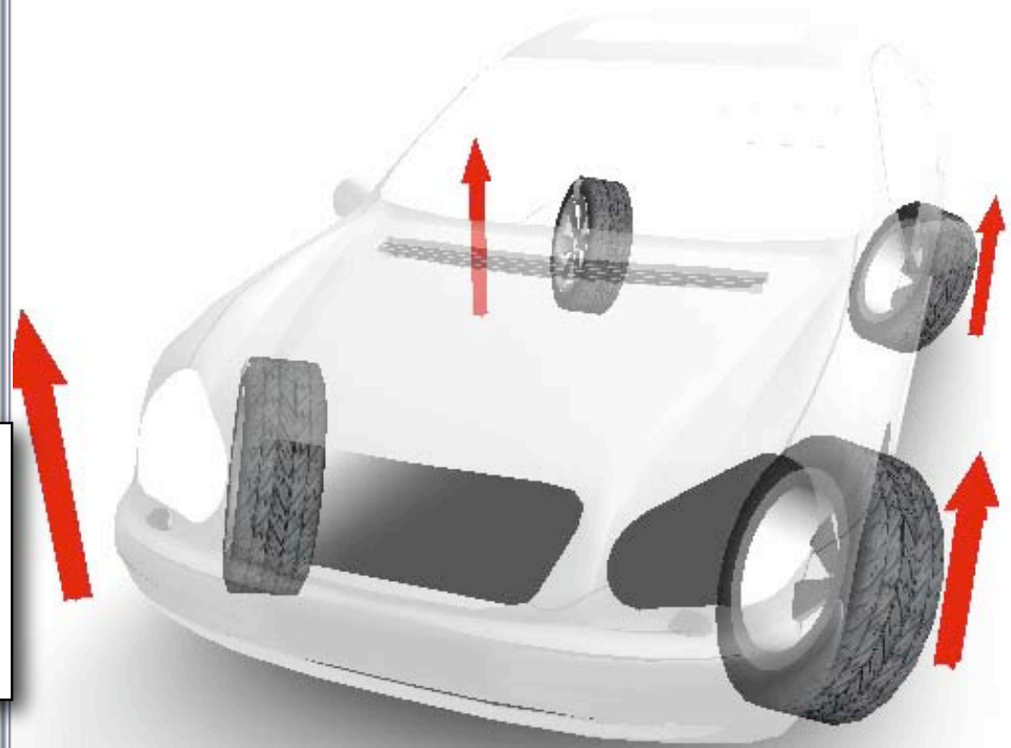
```
/* ------------------------------------------------------------------------
   Define the equations of motion for CarSim 7.01 (Torsionally Rigid)
   vehicle-suspension arrangement: i_i, which has 19 degrees of freedom. Each
   derivative evaluation requires 8008 multiply/divides, 7001 add/subtracts,
   and 492 function calls. Copyright 2006. Mechanical Simulation Corporation.
------------------------------------------------------------------------ */
void  vsm_difeqn (vs_real t, vs_real q[], vs_real qp[], vs_real u[], vs_real
            up[])
{

  // Compute sines and cosines of angular coordinates
  dyvars.s[3] = sin(q[3]);
  dyvars.s[4] = sin(q[4]);
  dyvars.s[5] = sin(q[5]);
  dyvars.s[16] = sin(q[16]);
  dyvars.s[17] = sin(q[17]);
  dyvars.s[18] = sin(q[18]);
```

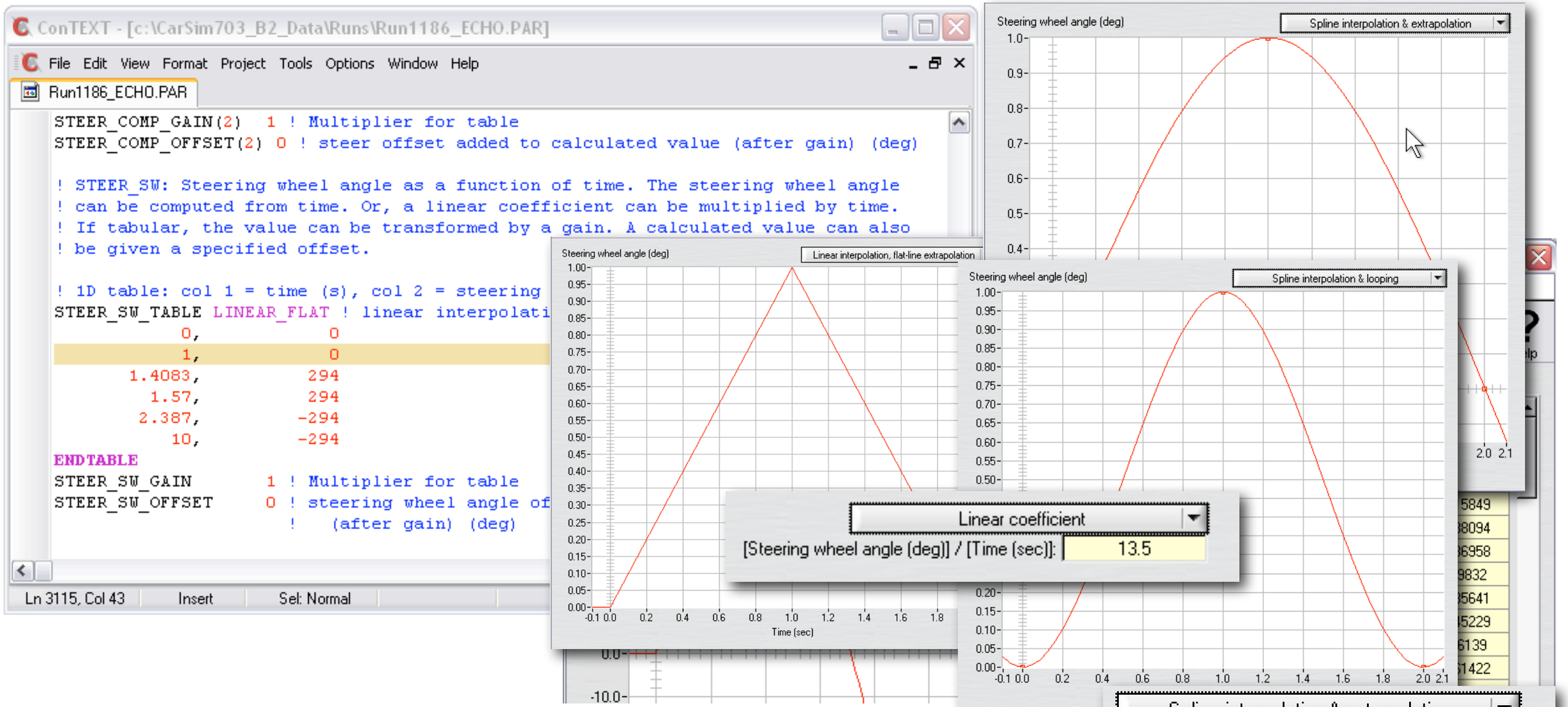
```
  dyvars.z[10600] = dyvars.z[6969] + dyvars.z[10596] + dyvars.z[10597] +
          dyvars.z[10599] -dyvars.z[10595] -dyvars.z[10598];
  dyvars.z[10601] = dyvars.z[10600]/dyvars.z[3812];
  dyvars.z[10602] = dyvars.z[9234]*dyvars.z[10508];
  dyvars.z[10603] = dyvars.z[9238]*dyvars.z[10511];
  dyvars.z[10604] = dyvars.z[9236]*dyvars.z[10515];
  dyvars.z[10605] = dy
  dyvars.z[10606] = dy
  dyvars.z[10607] = dy
          -dyvars.z[10
  dyvars.z[10608] = dy
  dyvars.z[10609] = in
  dyvars.z[10610] = dy
  dyvars.z[10611] = pr
  up[3] = dyvars.z[10508];
  up[7] = -dyvars.z[10511];
  up[5] = dyvars.z[10515];
  up[4] = dyvars.z[10520];
```

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

Example: 3D suspension/ steering kinematics





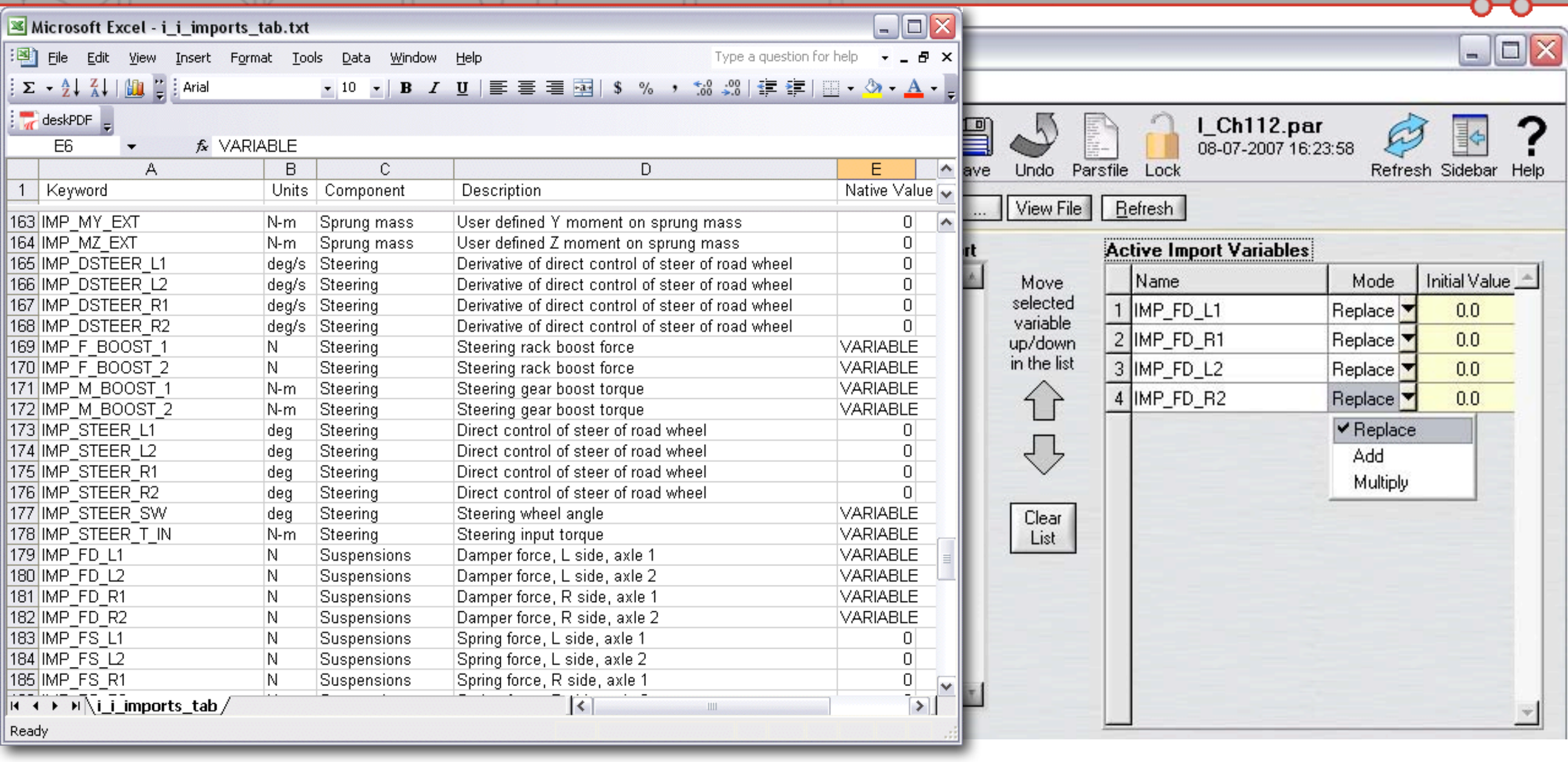


Runtime Table Options

- 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

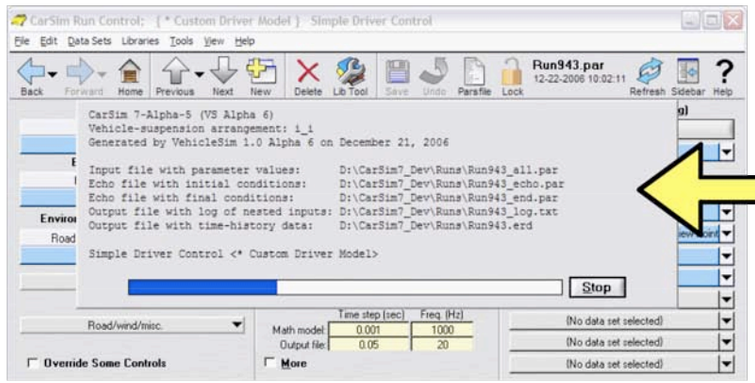
- Combine with “native variable” by add, replace, multiply
- Machine-generated documentation (Excel)

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**

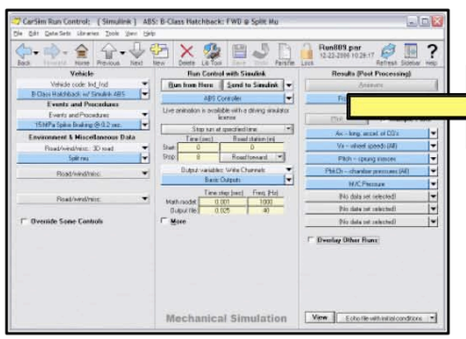
VehicleSim API

- One set of DLLs
  - VehicleSim Browser (SGUI)
  - Simulink
  - LabView
  - ASCET
  - Custom EXE
  - More…
- VS API provides 3D road, tire
- Extend with custom C/C++

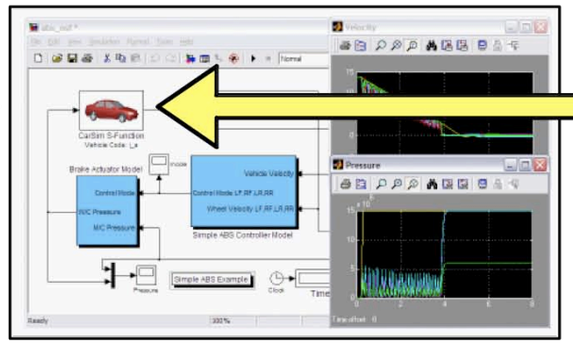


VS Solver DLL

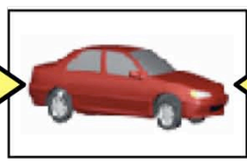
SGUI



SGUI

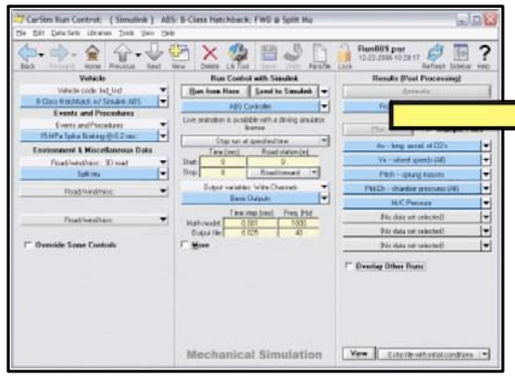


MATLAB/Simulink MDL

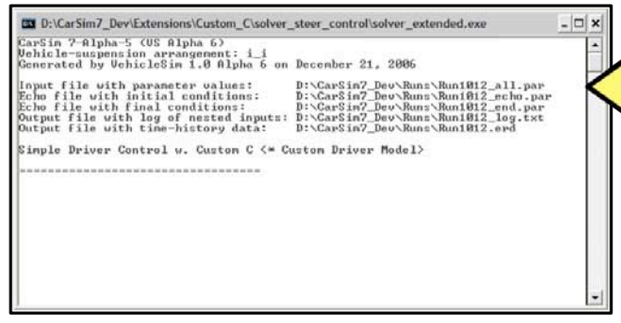


Wrapper DLL

VS Solver DLL



SGUI



Wrapper EXE program

VS Solver DLL

VS API: You control the simulation

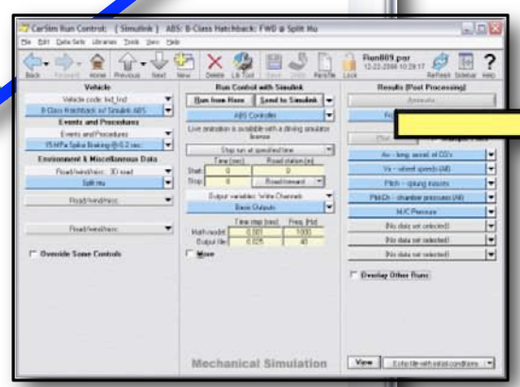
solver_extended.c - Microsoft Visual Studio

File Edit View Debug Tools Window Community Help

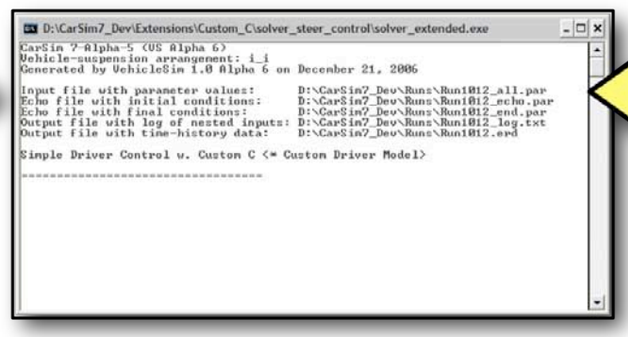
solver_extended.c

```
/* ------------------------------------------------------------------
    Main program to run built-in vehicle model gVsDLL.
------------------------------------------------------------------ */
void main(int argc, char **argv)
{
  double t;
  char   pathDLL[FILENAME_MAX], simfile[FILENAME_MAX]={"simfile"}, *prir
  int    ibarg = 0;

  // get simfile from argument list and load DLL
  if (argc > 1) strcpy(simfile, &argv[1][0]);
  if (vs_get_dll_path (simfile, pathDLL)) return;
  if (vs_api_load_dll (pathDLL)) return;
  printMsg = gAPI.vs_get_output_message(); // pointer to text from DLL

  // Initialize VS model
  t = gAPI.vs_setdef_and_read(simfile, external_setdef, external_scan);
  vss_check_for_dll_error ();
  gAPI.vs_initialize (t, external_calc, external_echo);
  vss_print_message (printMsg);
  vss_check_for_dll_error ();

  // Run. Each loop advances time one full step.
  while (!gAPI.vs_stop_run())
    {
    gAPI.vs_integrate (&t, external_calc);
    vss_check_for_dll_error ();
    gAPI.vs_bar_graph_update (&ibarg); // update ba
    vss_print_message (printMsg);
    }

  // Terminate
  external_calc (t, VS_EXT_EQ_END);
  gAPI.vs_terminate (t, external_echo);
  vss_print_message (printMsg);
  vss_opt_pause ();
  vs_api_unload_dll();
  return;
}
```

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



SGUI



Wrapper EXE program

VS Solver DLL

external.c - Microsoft Visual Studio

File Edit View Debug Tools Window Community Help

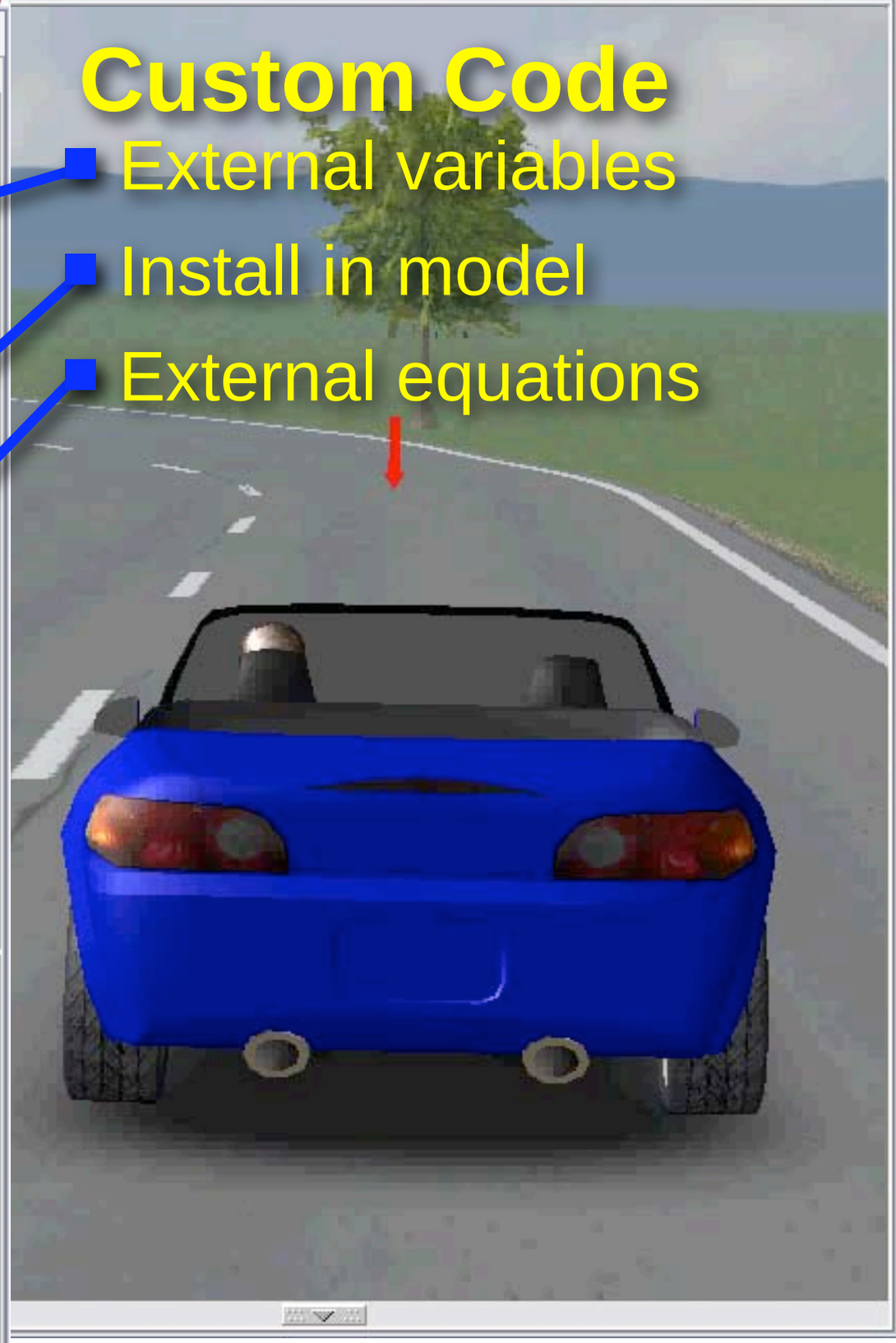
external.c

```
/* ---------------------------------------------------------------------
   Set up variables for the model extension. For the steering controller
   define new units and parameters and set default values of the paramet
   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;
  sLfwd = 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 vehicl
                            &sLfwd, "M");
  gAPI.vs_define_parameter ("LAT_TRACK",
                            "Lateral offset (to driver's left) for targe
                            &sLatTrack, "M");
```

```
case VS_EXT_EQ_IN: // calculations at the start of a time step

  // calculate X and Y coordinates of preview point
  sXprev = *sXcg + sLfwd*cos(*sYaw);
  sYprev = *sYcg + sLfwd*sin(*sYaw);

  if (!sUseExternal) ; // no effect if sUseExternal is FALSE
  else if (t <= *sTstart) *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;
```

Ready Ln 43 Col 2 Ch 2 INS

Help



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

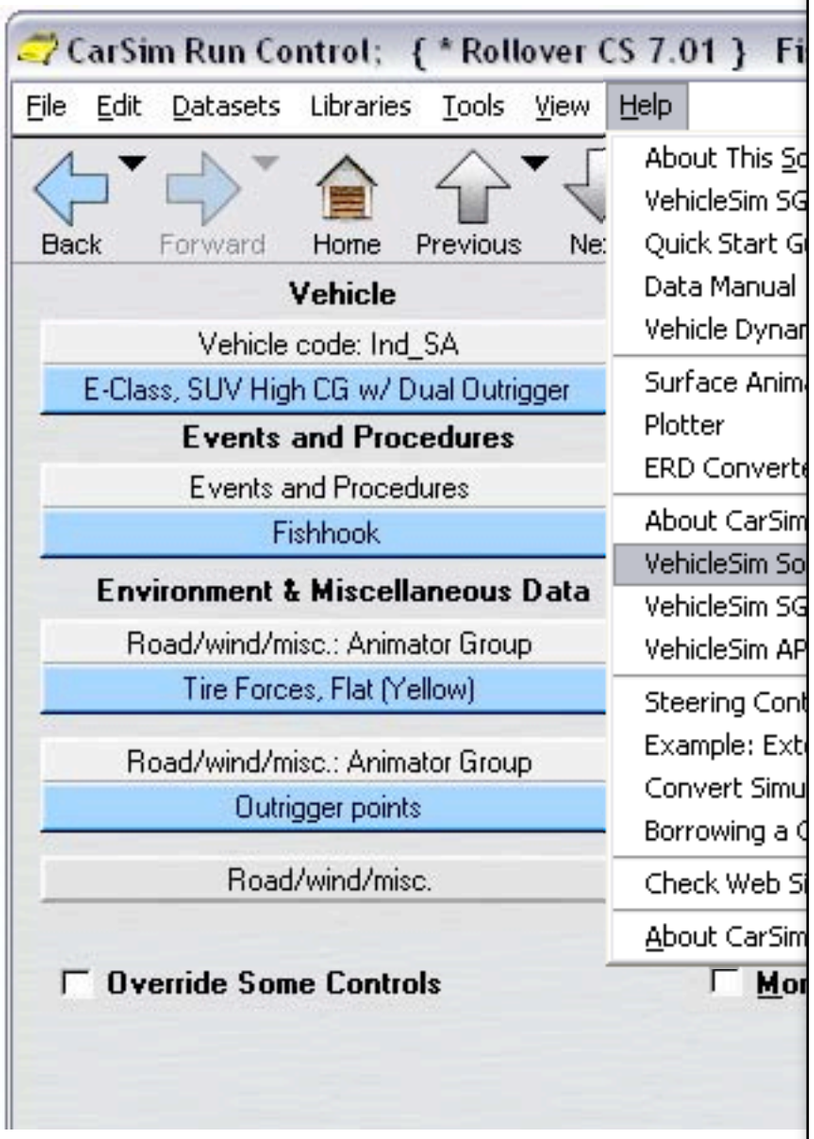


Table 5. VehicleSim Commands.

Command	Action
DEFINE_EVENT	Define new event.
DEFINE_IMPORT	Define a new potential import variable (to pass-through data).
DEFINE_OUTPUT	Define a new output variable for export, plotting, and animation.
DEFINE_VARIABLE	Define a new variable available to the solver.
DEFINE_UNITS	Install new units in the VS solver.
EQ_DIFFERENTIAL	Add an equation to calculate the derivative of a new variable.
EQ_END	Add an equation that is applied when the run terminates.
EQ_IN	Add an equation that is applied at the start of a time step.
EQ_INIT	Add an equation applied before initial outputs are calculated.
EQ_INIT2	Add an equation applied after initial outputs are calculated.
EQ_OUT	Add an equation that is applied at the end of a time step.
EQ_PRE_INIT	Add an equation that is applied just before initialization.
EVENT_SET_GT	Define new event (old version, not recommended).
EVENT_SET_LT	Define new event (old version, not recommended).
REDEFINE_UNITS	Redefine existing units in the VS solver.
RESET_EVENTS	Clear any pending events.
RESET_EXPORTS	Disable all export variables.
RESET_IMPORTS	Disable all import variables.
RESET_LIVE_ANI	Disable all live animator broadcast variables.
SET_OUTPUT_COMPONENT	Set the 32-character component name for a new output variable.
SET_OUTPUT_GENERIC	Set the 32-character generic name for a new output variable.
SET_OUTPUT_LONG_NAME	Set the 32-character long name for a new output variable.
SET_UNITS	Set the units for a variable.

VehicleSim Commands

- Only a few commands
- Yet, powerful options

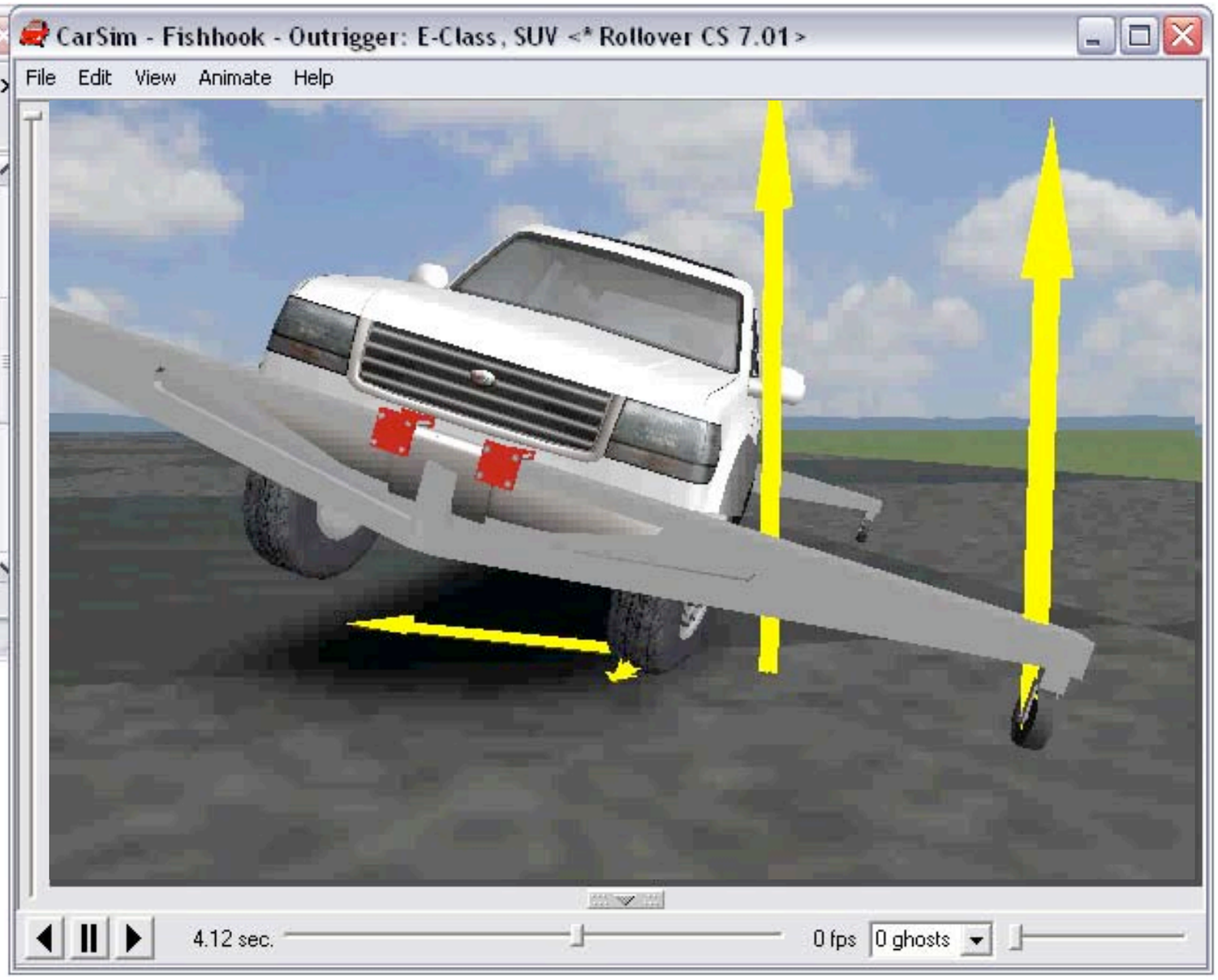
ConTEXT - [c:\CarSim703_B2_Data\Models\External\Ext11...

File Edit View Format Project Tools Options Window Help

Ext117.par

```
#MiscYellow1
define_units N/mm 0.001

define_variable K_RIG 1000; units = N/mm

eq_out IMP_FZ_EXT_7 = max((.09 -Z_RP7)*K_RIG,0);
eq_out IMP_FZ_EXT_8 = max((.09 -Z_RP8)*K_RIG,0);
eq_out IMP_FZ_EXT_9 = max((.09 -Z_RP9)*K_RIG,0);
eq_out IMP_FZ_EXT_10 = max((.09 -Z_RP10)*K_RIG,0);

IMP_FZ_EXT_7 replace 0;
IMP_FZ_EXT_8 replace 0;
IMP_FZ_EXT_9 replace 0;
IMP_FZ_EXT_10 replace 0;
```

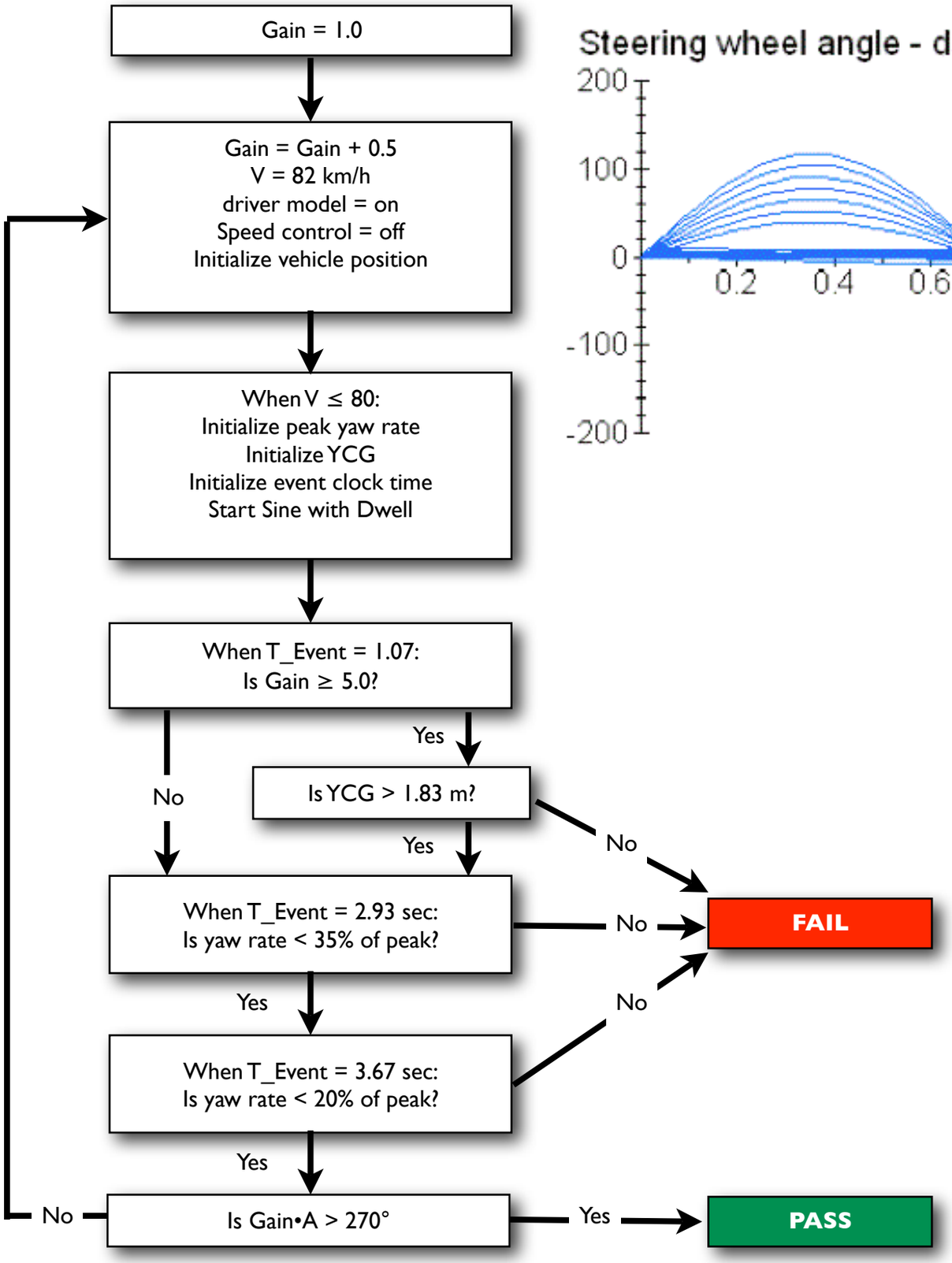
Ln 1, Col 1 | Insert | Sel: Normal

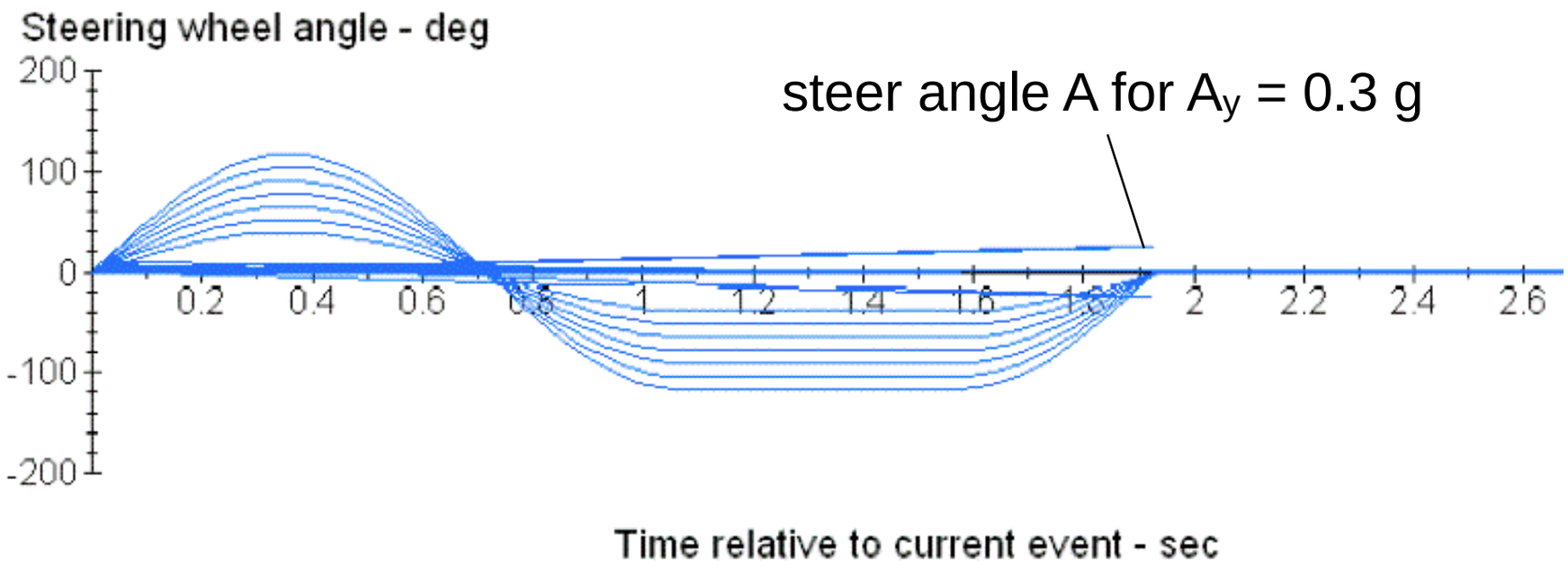


Outrigger Example

- 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

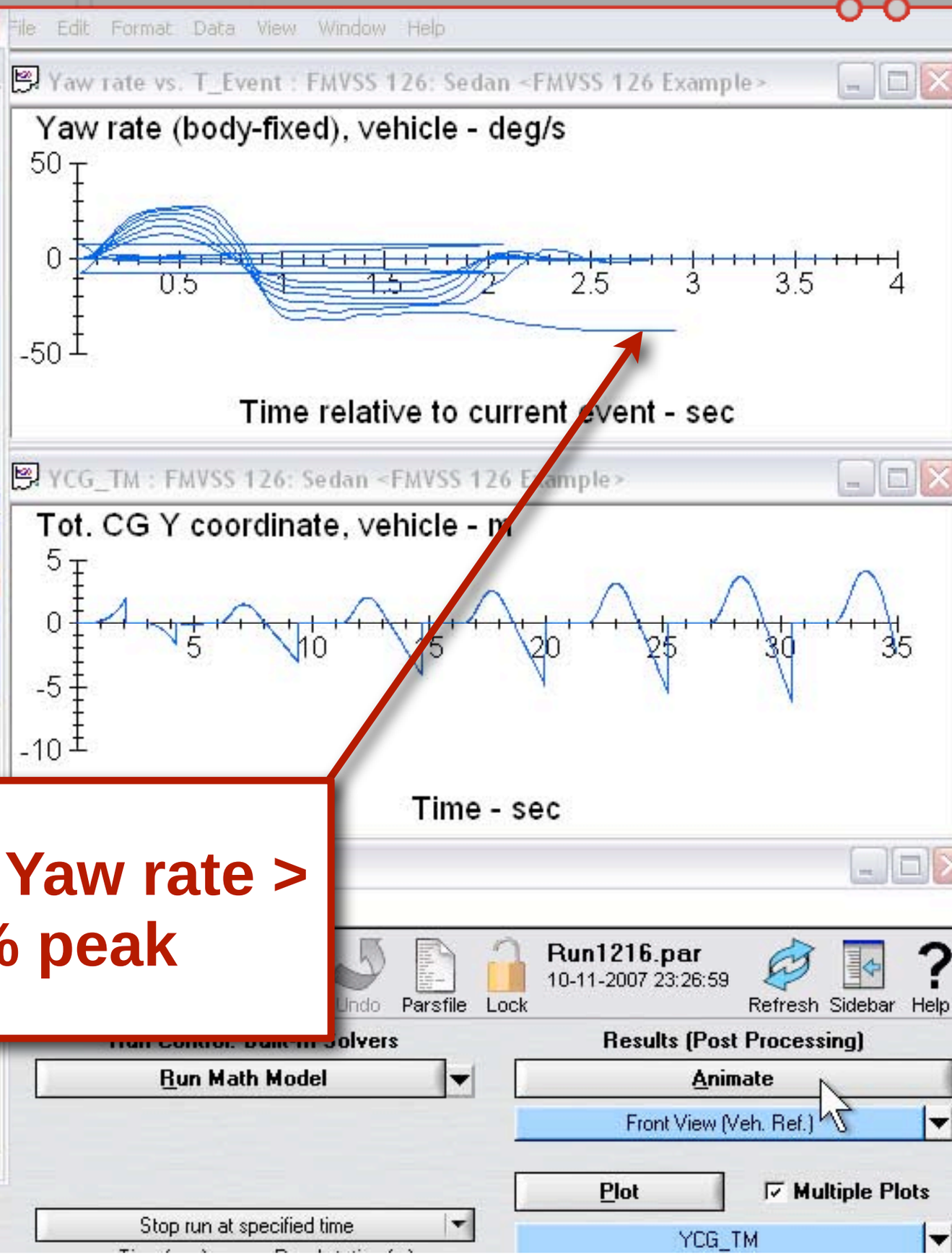




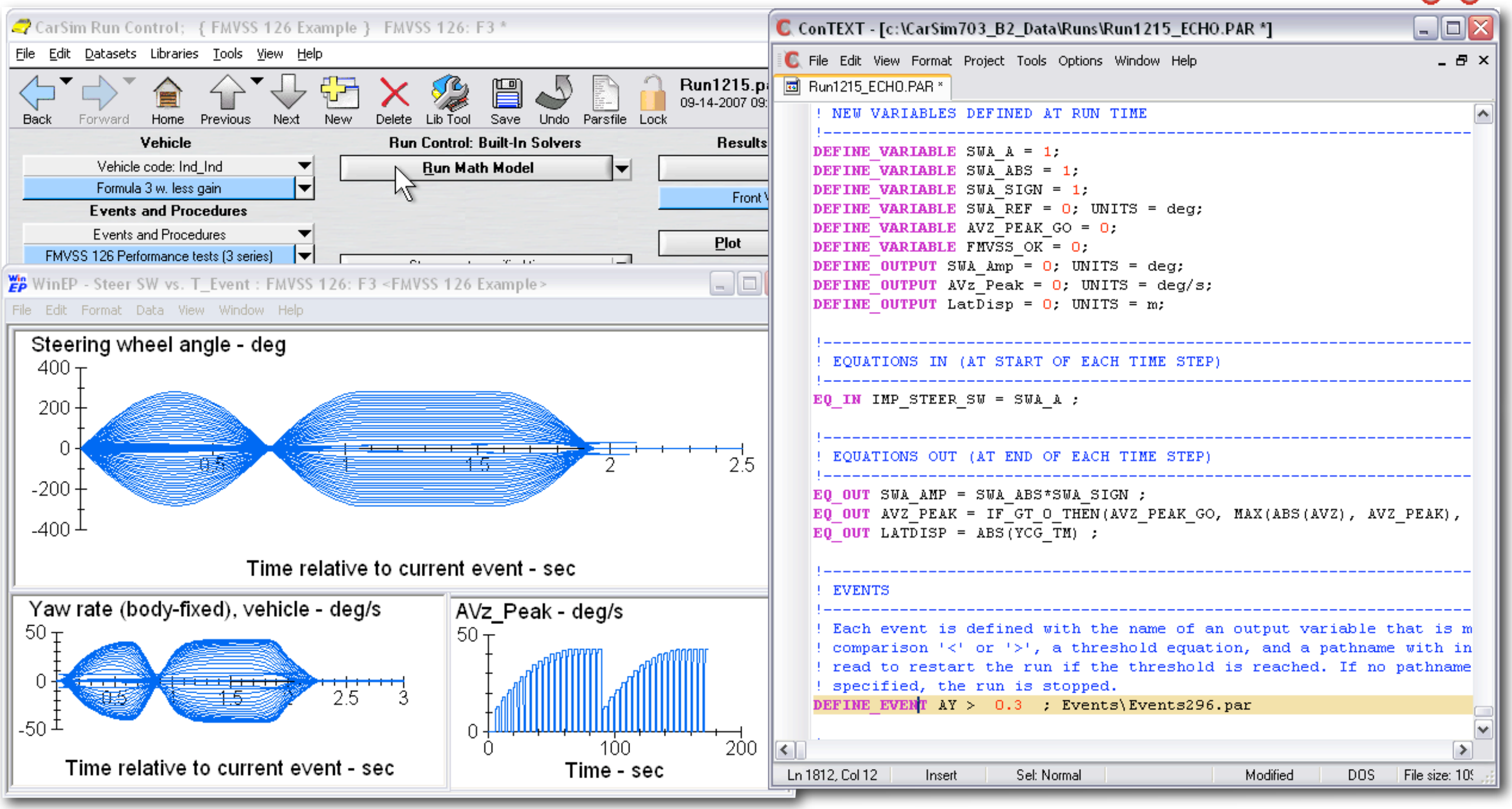
ESC Test: FMVSS 126

- Run tests to find steer for A_y = ±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



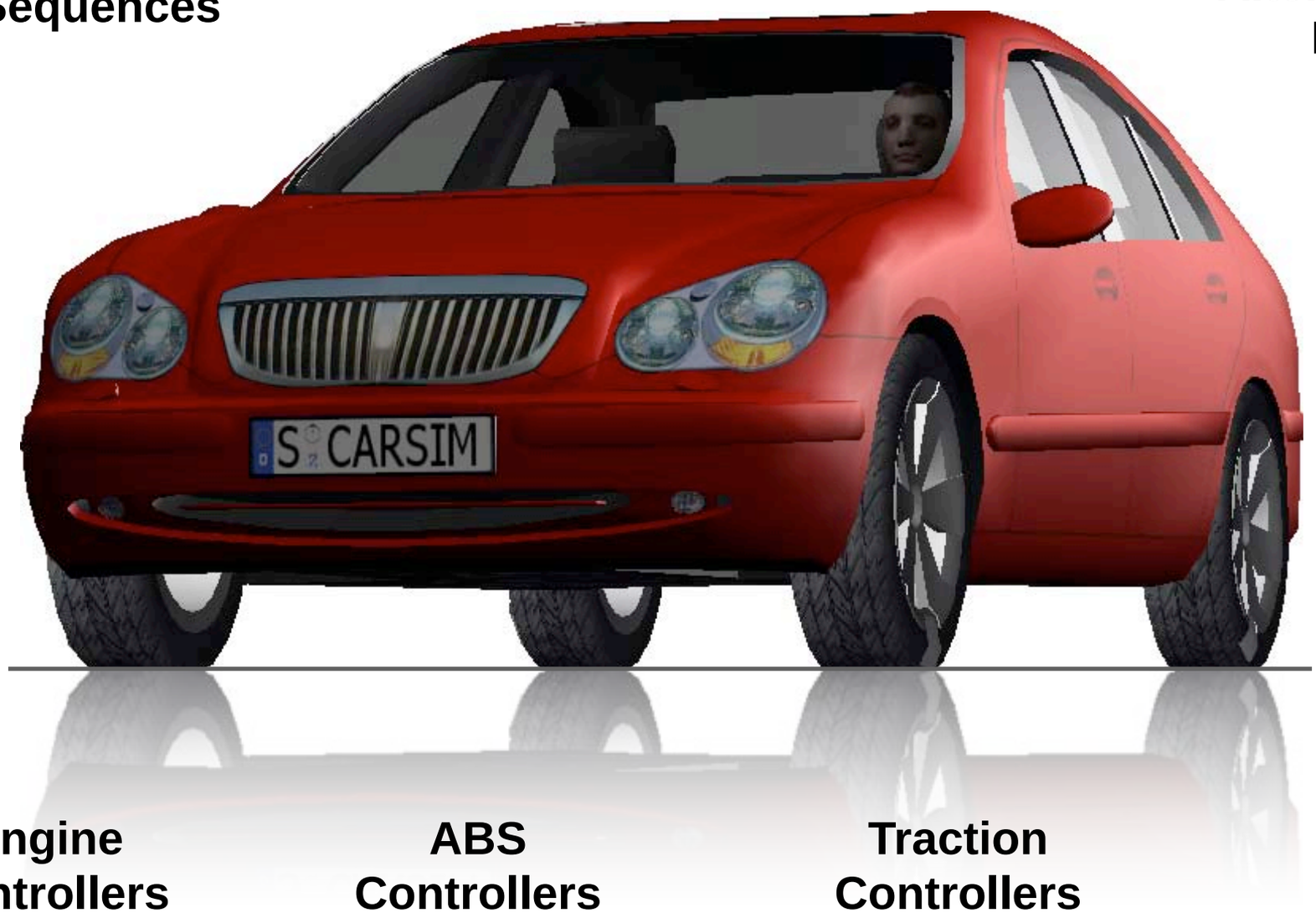
VS Commands extend the model

- Add new variables
- Add equations
- Define events

Complex Testing Sequences

Alternate Driver Models

Alternate Powertrain Models



Alternate Steering Models

Active Suspension Models

Drive Train Component Controllers

Alternate Tire Models

Engine Controllers

ABS Controllers

Traction Controllers

ESC

CarSim Has the Core Vehicle Model

Extend the core model as needed

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