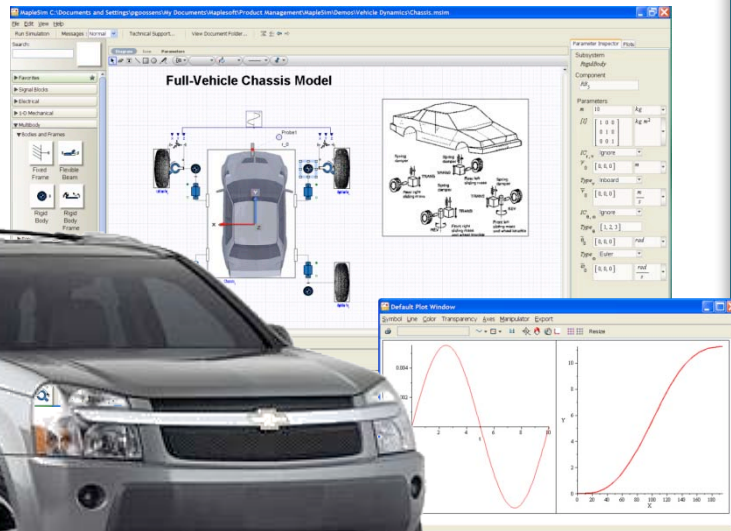




High-Performance Multi-Domain Modeling and Simulation



Next-Generation Modeling and Simulation Tools for Stability Control Development

Paul Goossens, Maplesoft



© 2008 Maplesoft, a division of Waterloo Maple Inc.

Outline

- Maplesoft's Role in Vehicle Dynamics and Control Development
- Introduction to MapleSim
- Real-time Applications

Physical Modeling creates better products - faster

Toyota and Maplesoft enter a multi-year partnership to produce new tools for knowledge-rich physical modeling
Contract part of new engineering tool

Waterline, November 2007. Maplesoft™, the leading provider of high-performance software tools for engineering, science, and mathematics today announced a multi-year contract with Toyota Motor Corporation, the world's largest automobile company. The partnership will produce advanced physical modeling tools to help Toyota move to a new product development process called the Model-Based Development (MBD). Key features of the new process include Control System Design and Physical ("Plant") Modeling based on a symbolic approach.

Toyota has been one of the earliest among industrial companies to embrace Model-Based Design, the concept of creating a computer-based model of a system to analyze, test, improve, and optimize it before building the physical system. In the initial stages, this was used in the design, simulation and implementation of control systems using tools from companies such as The MathWorks™. Toyota is now expanding its scope with the development of the new Model-Based Development process.

Forbes.com
"Tools developed by Maplesoft will provide the fundamental mathematical framework for physical modeling within the Model-Based Development throughout Toyota... Toyota has recognized Maple™ as an important part of this framework, providing the ideal mathematical environment for Physics-based modeling."

Computer Business Review
"Key features of the model-based development (MBD) process include control system design and physical modeling based on a symbolic approach. The goal of MBD is to improve time-to-market, quality, and reliability, while reducing cost."

Yahoo Auto
"The partnership will produce advanced physical modeling tools to help Toyota move to a new product development process called the Model-Based Development (MBD). Key features of the new process include Control System Design and Physical ("Plant") Modeling based on a symbolic approach."

Digital 50
"Mathematical environments such as Maple have many advantages," said Jim Cooper, President and CEO of Maplesoft. "By describing the complex, causal relationships of a physical model in a clear and efficient way, Maple enables simplification and optimization, taming the complexity of large models and reducing development and testing time."

CBC Canada
"The partnership will produce advanced physical modeling tools to help Toyota move to a new product development process. The goal of (the new process) is to improve time-to-market, quality, and reliability, while reducing cost."

The goal of MBD is to improve time-to-market, quality and reliability, while reducing cost. Physical Modeling requires a symbolic approach to computations in order to accurately and efficiently represent real-world physical systems. Toyota has recognized Maple™ as an important part of this framework, providing the ideal mathematical environment for Physics-based modeling. Maple is an advanced software tool from Maplesoft that relies on a powerful computation engine to derive and solve complex sets of equations, simplify large sets of equations, develop advanced mathematical models, and create user-friendly technical applications.

"Mathematical environments such as Maple have many advantages," said Jim Cooper, President and CEO of Maplesoft. "By describing the complex, causal relationships of a physical model in a clear and efficient way, Maple enables simplification and optimization, taming the complexity of large models and reducing development and testing time. Furthermore, the Maple environment makes solutions more readable and understandable, rendering knowledge capture and re-use simple and effective. We are delighted to partner with Toyota in this innovative venture."

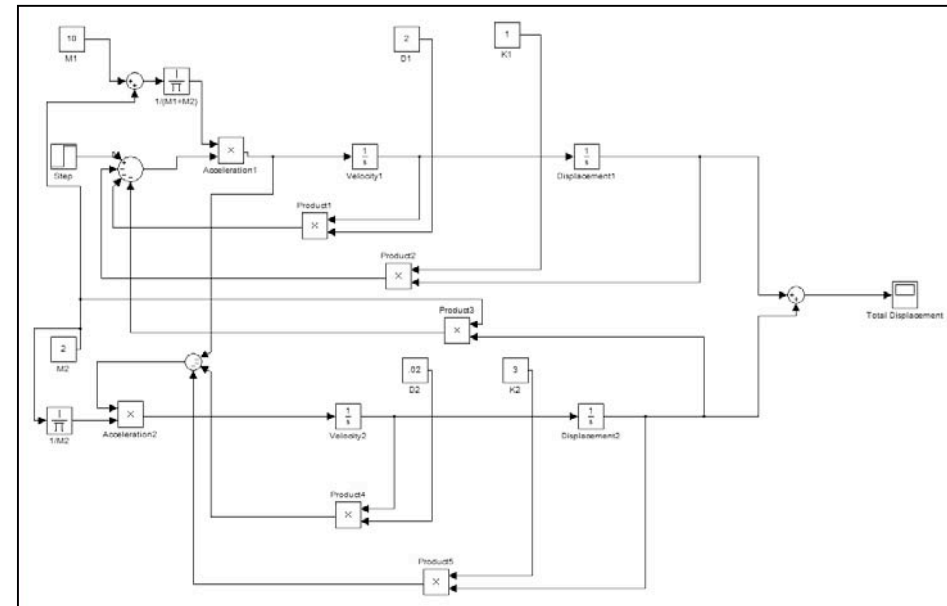
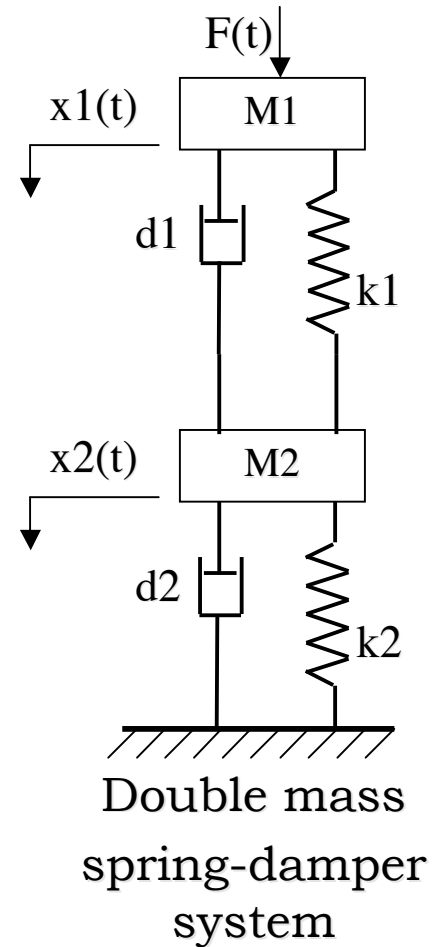
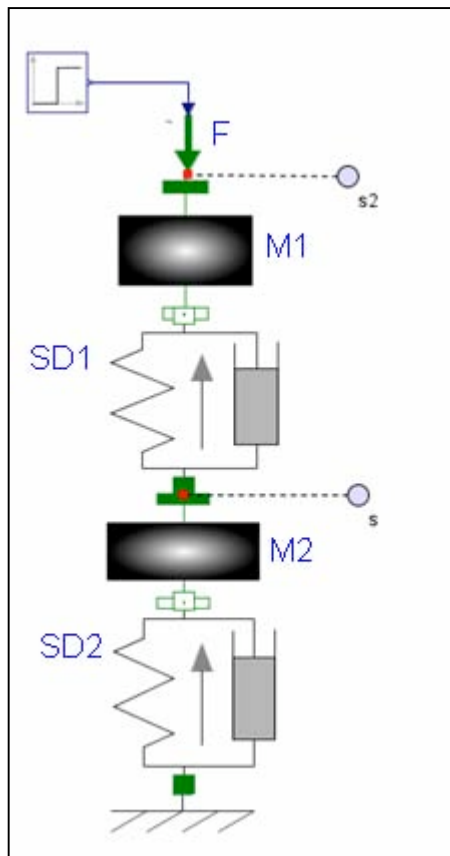
"Automotive companies like us will realize further improvements in cycle times, cost optimization, and smoother implementation of extremely complex systems."

Mr. Akira Ohata: Project General Manager
Toyota Motor Corporation

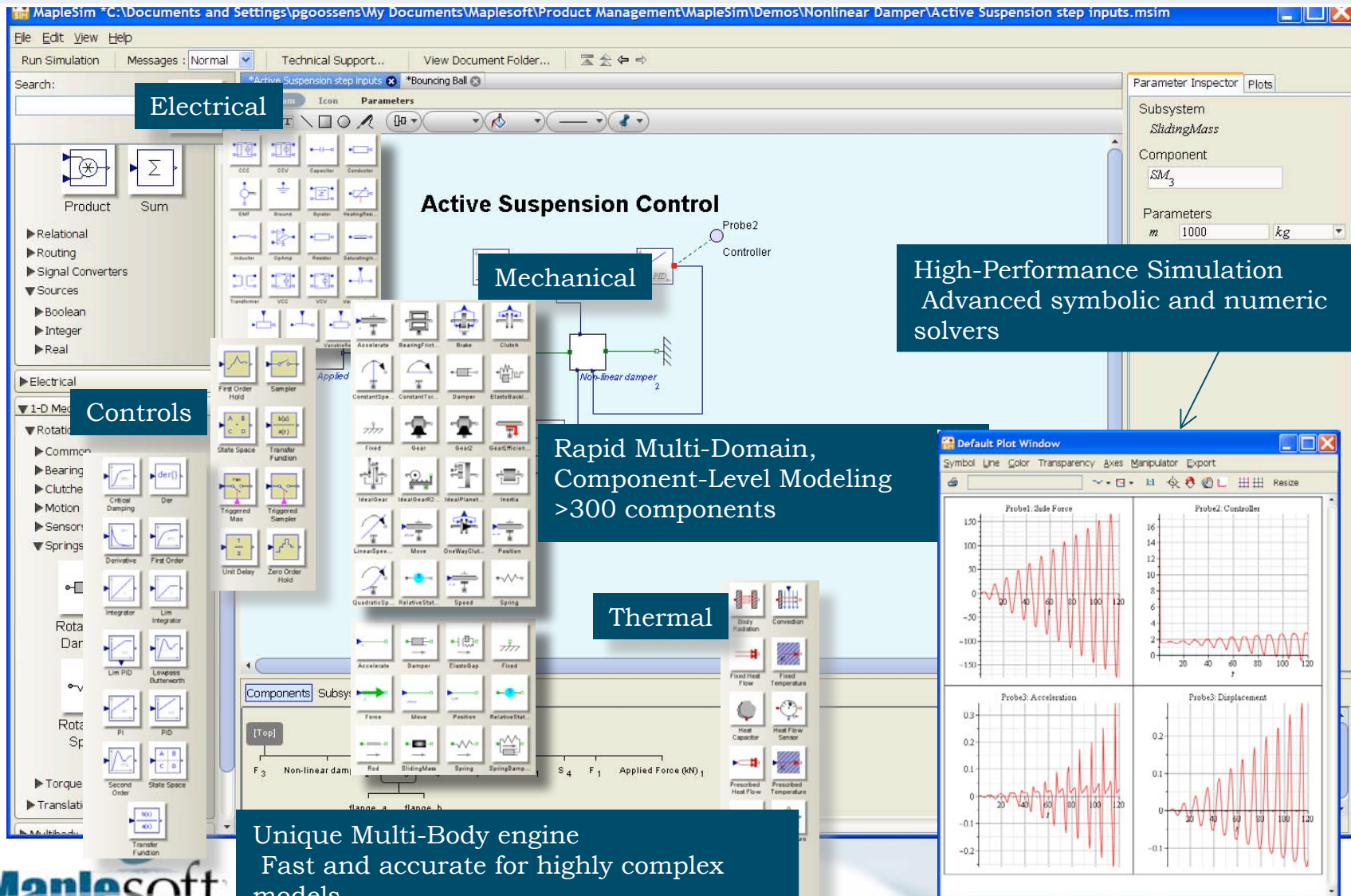
- Speed-to-market for new products is their main growth factor
- Physical modeling produces better products and reduces time to market



Physical Modeling – Faster and Intuitive



- Model maps directly to physical components of system
- Automatically generates equations of motion



The screenshot displays the MapleSim software interface for a simulation titled "Active Suspension Control". The main workspace shows a mechanical model of a suspension system with a mass, springs, and a damper. A control loop is implemented, featuring a PID controller and a non-linear damper. The interface includes a menu bar, a search bar, and several toolbars for component selection and simulation control. A "Parameter Inspector" window on the right shows the parameters for a "SlidingMass" component, with the mass m set to 1000 kg. A "Default Plot Window" in the bottom right corner displays four time-series plots: "Probe1: Side Force", "Probe2: Controller", "Probe3: Acceleration", and "Probe3: Displacement".

Electrical

Active Suspension Control

Mechanical

High-Performance Simulation
Advanced symbolic and numeric solvers

Controls

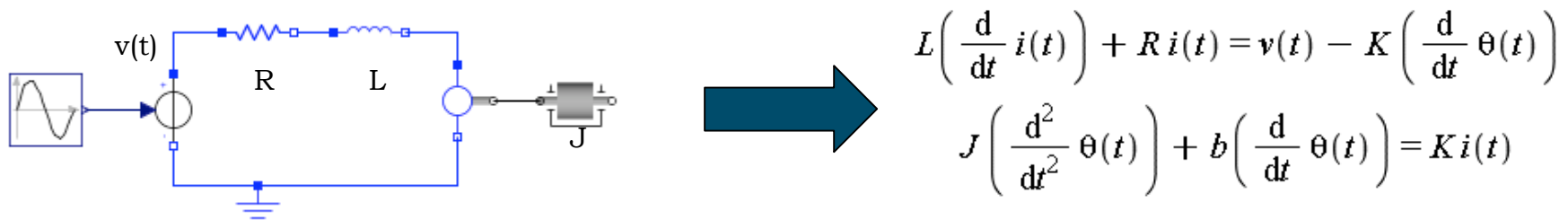
Rapid Multi-Domain,
Component-Level Modeling
>300 components

Thermal

Unique Multi-Body engine
Fast and accurate for highly complex models

Automatic Equation Generation

Symbolic Computation automatically converts physical system representations to mathematical models



- Rapid, error-free model formulation
- Concise and numerically efficient
- Parametric math model

Symbolic Simplification

$TorqueProbe1.flange_a_phi(t) - TorqueProbe1.flange_b_phi(t), TorqueProbe1.flange_a_tau(t) - TorqueProbe1.tau(t), TorqueProbe1.flange_b_tau(t) - TorqueProbe1.tau(t), AngularVelocityProbe1.phi(t) - AngularVelocityProbe1.flange_a_phi(t), D,$
 $- AngularVelocityProbe1.flange_a_tau(t), DCmotor1.Resistor1.r(t) - DCmotor1.Resistor1.p(t) - DCmotor1.Resistor1.n.r(t), D, - DCmotor1.Resistor1.p(t) + DCmotor1.Resistor1.n.t(t), DCmotor1.Resistor1.t(t) - DCmotor1.Resistor1.p(t), DCmotor1.Inertia1.flange_a_phi(t)$
 $- DCmotor1.Inertia1.phi(t), DCmotor1.Inertia1.flange_b_phi(t) - DCmotor1.Inertia1.phi(t), DCmotor1.Ground1.p.r(t) - D, DCmotor1.SignalVoltage1.r(t) - DCmotor1.SignalVoltage1.p.r(t) - DCmotor1.SignalVoltage1.n.r(t), D, - DCmotor1.SignalVoltage1.p(t)$
 $+ DCmotor1.SignalVoltage1.n.t(t), DCmotor1.SignalVoltage1.t(t) - DCmotor1.SignalVoltage1.p(t), DCmotor1.Inductor1.r(t) - DCmotor1.Inductor1.p.r(t) - DCmotor1.Inductor1.n.r(t), D, - DCmotor1.Inductor1.p(t) + DCmotor1.Inductor1.n.t(t), DCmotor1.Inductor1.t(t)$
 $- DCmotor1.Inductor1.p(t), DCmotor1.EMF1.r(t) - DCmotor1.EMF1.p.r(t) - DCmotor1.EMF1.n.r(t), D, - DCmotor1.EMF1.p(t) + DCmotor1.EMF1.n.t(t), DCmotor1.EMF1.t(t) - DCmotor1.EMF1.p(t), DCmotor1.Inertia1.flange_b_tau(t) - DCmotor1.flange_b_tau(t) - D,$
 $DCmotor1.Inertia1.flange_b_phi(t) - DCmotor1.flange_b_phi(t), DCmotor1.SignalVoltage1.r(t) - DCmotor1.r(t), DCmotor1.EMF1.flange_b_tau(t) + DCmotor1.Inertia1.flange_a_tau(t) - D, DCmotor1.EMF1.flange_b_phi(t) - DCmotor1.Inertia1.flange_a_phi(t), DCmotor1.EMF1.p(t)$
 $+ DCmotor1.Inductor1.n.t(t) - D, DCmotor1.EMF1.p.r(t) - DCmotor1.Inductor1.n.r(t), DCmotor1.EMF1.n.t(t) + DCmotor1.SignalVoltage1.n.t(t) + DCmotor1.Ground1.p(t) - D, DCmotor1.EMF1.n.r(t) - DCmotor1.SignalVoltage1.n.r(t), DCmotor1.SignalVoltage1.n.p.r(t)$
 $- DCmotor1.Ground1.p.r(t), DCmotor1.SignalVoltage1.p(t) + DCmotor1.Resistor1.p(t) - D, DCmotor1.SignalVoltage1.p.r(t) - DCmotor1.Resistor1.p.r(t), IdealGear1.bearing_phi(t) - D, D, - IdealGear1.flange_a_tau(t) + IdealGear1.flange_b_tau(t) + IdealGear1.tau_support(t),$
 $IdealGear1.phi(t) - IdealGear1.flange_a_phi(t) - IdealGear1.bearing_phi(t), IdealGear1.phi(t) - IdealGear1.flange_b_phi(t) - IdealGear1.bearing_phi(t), Inertia2.flange_a_phi(t) - Inertia2.phi(t), Inertia2.flange_b_phi(t) - Inertia2.phi(t), SpringDamper1.phi_rel(t)$
 $- SpringDamper1.flange_b_phi(t) - SpringDamper1.flange_a_phi(t), SpringDamper1.flange_b_tau(t) - SpringDamper1.tau(t), SpringDamper1.flange_a_tau(t) - SpringDamper1.tau(t), TorqueProbe2.flange_a_phi(t) - TorqueProbe2.flange_b_phi(t), TorqueProbe2.flange_a_tau(t)$
 $- TorqueProbe2.tau(t), TorqueProbe2.flange_b_tau(t) - TorqueProbe2.tau(t), AngularVelocityProbe2.phi(t) - AngularVelocityProbe2.flange_a_phi(t), D, - AngularVelocityProbe2.flange_a_tau(t), D, - SpeedSensor1.flange_a_tau(t), Feedback1.y(t) - Feedback1.u(t)$
 $- Feedback1.u(t), Inertia2.flange_b_tau(t) + SpeedSensor1.flange_a_tau(t) - D, Inertia2.flange_b_phi(t) - SpeedSensor1.flange_a_phi(t), Feedback1.u(t) - SpeedSensor1.w(t), ZZ1.y(t) - DCmotor1.r(t), Feedback1.y(t) - ZZ1.u(t), Solp1.y(t) - Feedback1.u(t),$
 $AngularVelocityProbe2.flange_a_tau(t) + SpringDamper1.flange_b_tau(t) + TorqueProbe2.flange_a_tau(t) - D, AngularVelocityProbe2.flange_a_phi(t) - SpringDamper1.flange_b_phi(t), SpringDamper1.flange_b_phi(t) - TorqueProbe2.flange_a_phi(t), TorqueProbe2.flange_b_tau(t)$
 $+ Inertia2.flange_a_tau(t) - D, TorqueProbe2.flange_b_phi(t) - Inertia2.flange_a_phi(t), IdealGear1.flange_b_tau(t) + SpringDamper1.flange_a_tau(t) - D, IdealGear1.flange_b_phi(t) - SpringDamper1.flange_a_phi(t), TorqueProbe1.flange_b_tau(t) + IdealGear1.flange_a_tau(t)$
 $- D, TorqueProbe1.flange_b_phi(t) - IdealGear1.flange_a_phi(t), AngularVelocityProbe1.flange_a_tau(t) + DCmotor1.flange_b_tau(t) + TorqueProbe1.flange_a_tau(t) - D, AngularVelocityProbe1.flange_a_phi(t) - DCmotor1.flange_b_phi(t), DCmotor1.flange_b_phi(t)$
 $- TorqueProbe1.flange_a_phi(t), IdealGear1.bearing_tau(t) - D, Solp1.y(t) - D, $\begin{cases} t < 0 \\ otherwise \end{cases}$ AngularVelocityProbe1.w(t) - $\frac{d}{dt}$ AngularVelocityProbe1.phi(t), DCmotor1.Inertia1.w(t) - $\frac{d}{dt}$ DCmotor1.Inertia1.phi(t), DCmotor1.Inertia1.t(t) - $\frac{d}{dt}$ DCmotor1.Inertia1.w(t),$
 $DCmotor1.EMF1.w(t) - $\frac{d}{dt}$ DCmotor1.EMF1.flange_b_phi(t), Inertia2.w(t) - $\frac{d}{dt}$ Inertia2.phi(t), Inertia2.t(t) - $\frac{d}{dt}$ Inertia2.w(t), SpringDamper1.w_rel(t) - $\frac{d}{dt}$ SpringDamper1.phi_rel(t), AngularVelocityProbe2.w(t) - $\frac{d}{dt}$ AngularVelocityProbe2.phi(t), SpeedSensor1.w(t)$
 $- $\frac{d}{dt}$ SpeedSensor1.flange_a_phi(t), DCmotor1.Resistor1.n.t(t) + DCmotor1.Inductor1.p.t(t) - D, DCmotor1.Resistor1.n.r(t) - DCmotor1.Inductor1.p.r(t), SpringDamper1.phi_rel(t) - D, DCmotor1.Inertia1.w(t) - D, ZZ1.t(t) - D, Inertia2.w(t) - D, AngularVelocityProbe2.phi(t)$
 $- D, DCmotor1.Inertia1.phi(t) - D, Inertia2.phi(t) - D, DCmotor1.Inductor1.t(t) - D, SpeedSensor1.flange_a_phi(t) - D, AngularVelocityProbe1.phi(t) - D, DCmotor1.EMF1.flange_b_phi(t) - D, DCmotor1.Resistor1.t(t) - DCmotor1.Resistor1.r(t), D, DCmotor1.Inertia1.t(t)$
 $- DCmotor1.Inertia1.flange_a_tau(t) + DCmotor1.Inertia1.flange_b_tau(t), DCmotor1.EMF1.w(t) - DCmotor1.EMF1.r(t),$
 $EMF1.flange_b_tau(t) + IdealGear1,$
 $- \frac{1}{5} ZZ1.u(t)$

Example: 2,300 equations simplified to 150

Speedup: 10 x

$- Inertia2.phi(t) + AngularVelocityProbe2.phi(t) = 0, - TorqueProbe1.flange_a_tau(t) + TorqueProbe2.flange_a_tau(t) = 0, DCmotor1.Inertia1.phi(t) - Inertia2.phi(t)$
 $+ SpringDamper1.phi_rel(t) = 0, - AngularVelocityProbe1.phi(t) + DCmotor1.Inertia1.phi(t) = 0, - DCmotor1.EMF1.flange_b_phi(t) + DCmotor1.Inertia1.phi(t) = 0, Inertia2.phi(t)$
 $- SpeedSensor1.flange_a_phi(t) = 0, TorqueProbe2.flange_a_tau(t) + SpringDamper1.phi_rel(t) + $\frac{d}{dt}$ SpringDamper1.phi_rel(t) = 0, DCmotor1.Inertia1.w(t)$
 $- \left(\frac{d}{dt} DCmotor1.Inertia1.phi(t) \right) = 0, $\frac{d}{dt} DCmotor1.Inductor1.i(t) + DCmotor1.Inductor1.i(t) - 10 \left(\frac{d}{dt} I11.x(t) \right) - 2 I11.x(t) + \frac{d}{dt} DCmotor1.EMF1.flange_b_phi(t) = 0,$$
 $- 10 \left(\frac{d}{dt} I11.x(t) \right) - 2 \left(\frac{d}{dt} SpeedSensor1.flange_a_phi(t) \right) + 2 \left(\begin{cases} 0, & t < 0 \\ 1, & otherwise \end{cases} \right) = 0, AngularVelocityProbe2.w(t) - \left(\frac{d}{dt} AngularVelocityProbe2.phi(t) \right) = 0,$
 $- AngularVelocityProbe1.w(t) + \frac{d}{dt} AngularVelocityProbe1.phi(t) = 0, - Inertia2.w(t) + \frac{d}{dt} Inertia2.phi(t) = 0, - TorqueProbe2.flange_a_tau(t) + \frac{d}{dt} Inertia2.w(t) = 0,$
 $- 2 DCmotor1.Inductor1.i(t) + 2 TorqueProbe1.flange_a_tau(t) + \frac{d}{dt} DCmotor1.Inertia1.w(t) = 0, SpringDamper1.phi_rel(t) = 0, DCmotor1.Inertia1.w(t) = 0, I11.x(t) = 0,$
 $Inertia2.w(t) = 0, AngularVelocityProbe2.phi(t) = 0, DCmotor1.Inertia1.phi(t) = 0, Inertia2.phi(t) = 0, DCmotor1.Inductor1.i(t) = 0, SpeedSensor1.flange_a_phi(t) = 0,$
 $AngularVelocityProbe1.phi(t) = 0, DCmotor1.EMF1.flange_b_phi(t) = 0$



Real-time Simulation

Motivation: **Real-time simulation** and control of engineering models of **arbitrary topology**

- **Real-time simulation**

MapleSim is built on top of Maple's symbolic engine. All models are generated symbolically, yielding efficient simulation code.

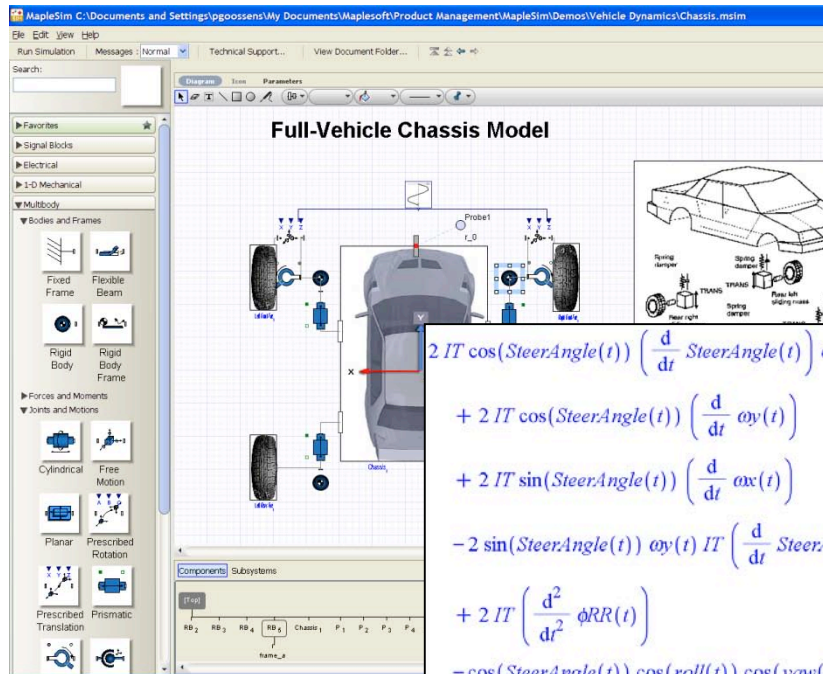
- **Arbitrary topology**

MapleSim is a general systems modeling tool. The user is not restricted to predefined topologies when generating simulation code.

Modeling Tool-chain

MapleSim

Full-Vehicle Chassis Model
- 14 DoF plus 8 DoF Tires



$$\begin{aligned}
 & 2IT \cos(\text{SteerAngle}(t)) \left(\frac{d}{dt} \text{SteerAngle}(t) \right) \text{ax}(t) \\
 & + 2IT \cos(\text{SteerAngle}(t)) \left(\frac{d}{dt} \text{ay}(t) \right) \\
 & + 2IT \sin(\text{SteerAngle}(t)) \left(\frac{d}{dt} \text{ax}(t) \right) \\
 & - 2 \sin(\text{SteerAngle}(t)) \text{ay}(t) IT \left(\frac{d}{dt} \text{SteerAngle}(t) \right) \\
 & + 2IT \left(\frac{d^2}{dt^2} \phi_{RR}(t) \right) \\
 & - \cos(\text{SteerAngle}(t)) \cos(\text{roll}(t)) \cos(\text{yaw}(t)) R3_TireR \\
 & - \cos(\text{SteerAngle}(t)) \cos(\text{roll}(t)) \sin(\text{yaw}(t)) R3_TireR \\
 & + \sin(\text{pitch}(t)) \sin(\text{SteerAngle}(t)) R1_TireRR FxTireRR \\
 & - \cos(\text{pitch}(t)) \sin(\text{yaw}(t)) \sin(\text{SteerAngle}(t)) R3_TireR \\
 & - \cos(\text{SteerAngle}(t)) \sin(\text{roll}(t)) \sin(\text{pitch}(t)) \cos(\text{yaw}(t)) \\
 & - \cos(\text{pitch}(t))
 \end{aligned}$$

I	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	kg m^2
$I_{C_{x,y}}$	ignore	
r_0	$[0, 0, 0]$	m

Maple 12

Analysis/Validation/Documentation
- Rapid Kinematic and Dynamic
Equation Generation

```

if (0.0e0 <= Alpha)
  signAlpha = 0.10e1;
else
  signAlpha = -0.10e1;
end if;
EY = (RBy1 + RBy2 * dFz) * (0.1e1 - (RBy3 + RBy4 * Gamma) * signAlpha) * LY;
RY = RBy1 * FzPrime * sin(0.2e1 * atan(Fz / RBy2 / FzPrime)) * (0.1e1 - RBy3 * Iabs(Gamma)) * LY;
BY = BY / Cy / By;
PY0 = By * sin(Cy * atan(By * Alpha - By * (By * Alpha - atan(By * Alpha)))) + SY;
DYkappa = My * Fz * (RBy1 + RBy2 * dFz * RBy3 * inclination) * cos(atan(RBy4 * slipangle));
SYkappa = DYkappa * sin(RBy5 * atan(RBy6 * longslip)) * LYkappa;
SHykappa = RBy1 + RBy2 * dFz;
Kappa = longslip + SHykappa;
Bykappa = RBy1 * cos(atan(RBy2 * (slipangle - RBy3)));
Cykappa = RBy1;
Eykappa = RBy1 + RBy2 * dFz;
Oykappa = cos(Cykappa) * atan(RBykappa * Kappa) - Eykappa;
Py = Py0 * Oykappa + SYkappa;
Rk = R0 * Fz * (Qk1 + LYk0 - Qk2 * inclination + Qk3);
if (0.0e0 <= spinrate)
  BC = (Qk1 + Qk2 * dFz + Qk3 * dFz * dFz) * (0.1e1 +
  CC = Qk1;
  DC = Fz / Fz0 * R0 * (Qk1 + Qk2 * dFz) * (0.1e1 + Qk3);
  EC = (Qk1 + Qk2 * dFz + Qk3 * dFz * dFz) * (0.1e1 +
  trail = DC * cos(CC) * atan(tit * AlphaTeq - EC * (BC * A
  sinf = siny + sy / By;
  AlphaR = slipangle + shf;
  
```

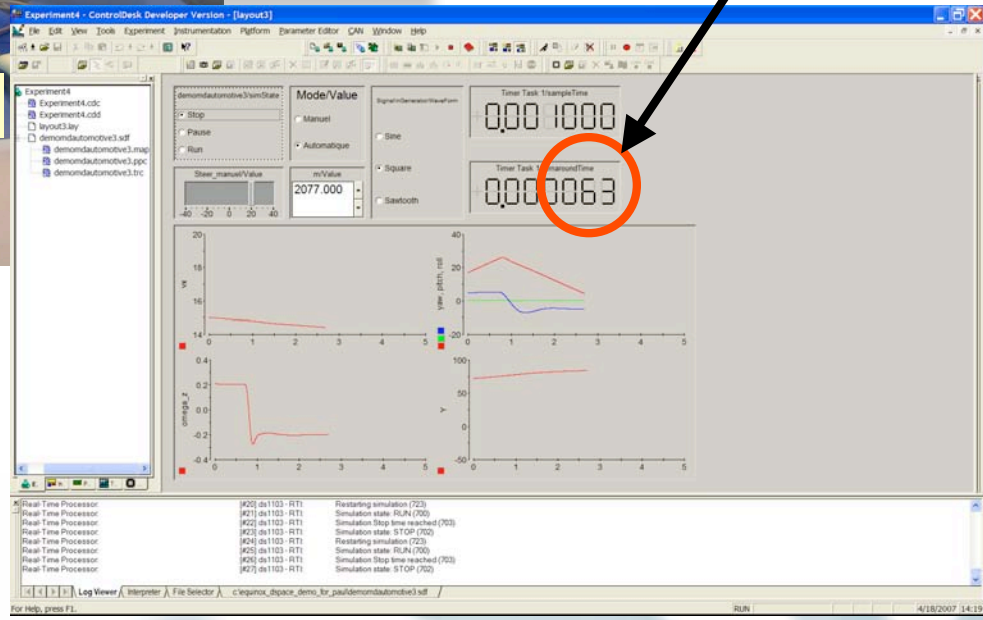
MapleSim Connectivity Toolbox
“One-Click” S-function Generation



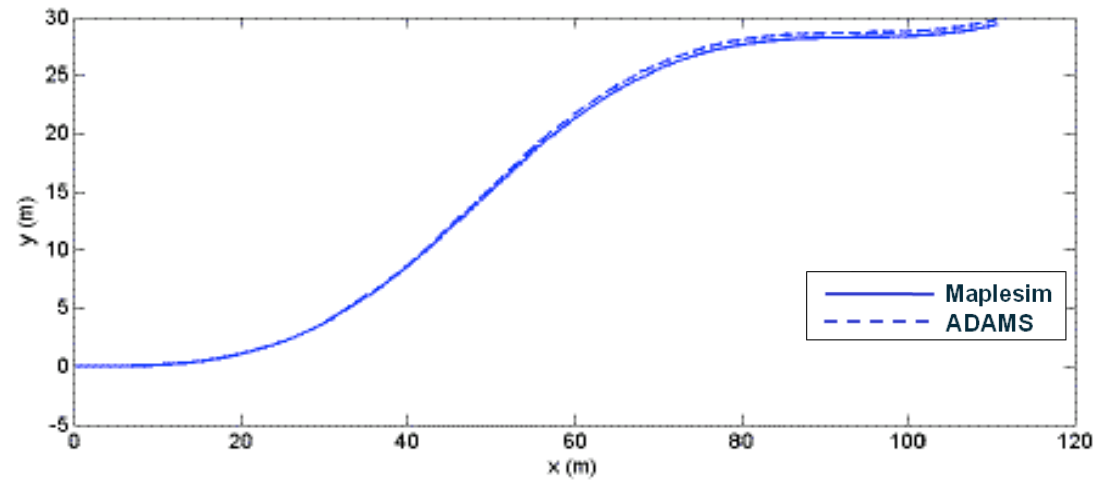
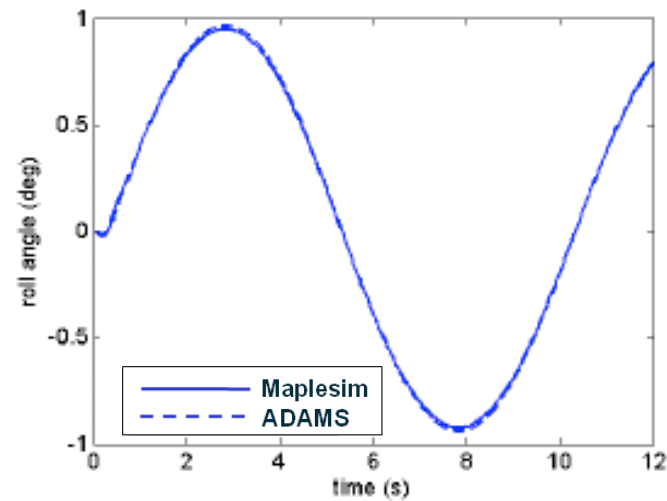
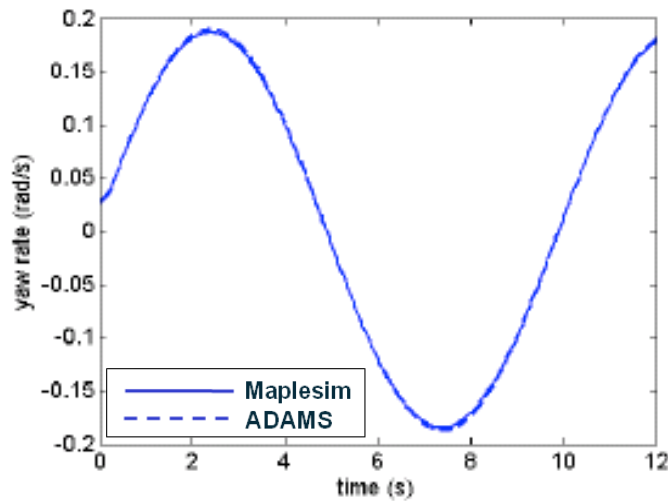
Real-Time Execution



63 μ s Cycle Time



Real-time Performance, with no loss of Fidelity



High-Performance Multi-Domain Simulation and Modeling



High-Performance Multi-Domain Modeling and Simulation

- Represent both physical system and controller
- Significant advantages over traditional systems:
 - Model diagram maps directly to the real system
 - Symbolic model formulation delivers greater numeric efficiency
 - Powerful, integrated analysis environment
- Shorten product development cycle

