System Modeling and Simulation in the Age of Internet of Things

Requirements and Altair's Offering

Altair Engineering – August 2017
The Old Development Approach

BUILD IT
PAINT IT
ADD CONTROLS

Design errors
Requirement misses
Longer development times
Higher development costs
Today’s Products are Complex
Today’s Products are Connected
Today’s Products Require Model Based Development Process
Models: From Components to System of Systems
Models: Multi-Disciplinary
# Models: From 0D to 3D

<table>
<thead>
<tr>
<th>Suspension</th>
<th>E-Motor</th>
<th>Electronics</th>
</tr>
</thead>
</table>
| 0D \[
x = \begin{bmatrix} A & B \\ C & D \end{bmatrix} x + u \\
y = \begin{bmatrix} x 
| \end{bmatrix} 
| \] | \[
\dot{x} = f(x,u,t) \\
y = g(x,u,t) 
| \] | \[
\Theta(s) = \frac{\alpha}{E_n(s) = J R s^2 + (B R + \alpha^2) s} 
| \] |

## 1D

- Quarter Car Ride Model with Active Suspension
- Permanent Magnet Synchronous Motor (PMSM)
- Vector Field-Oriented Control - Speed or Torque

## 3D

- Suspension E-Motor Electronics
Modeliis Acquisition (formerly part of EASii IC)
Model Based Development with Altair
Model Based Development with Altair

- Openness
- Usability
- 0D to 3D
- Business Model
Compose

Activate

Embed

Concept Studies | Control Design | System Performance Optimization | Controller Implementation
solidThinking Compose

\[
\begin{bmatrix}
a_0 & a_{-1} & a_{-2} & \ldots & a_{-n+1} \\
a_1 & a_0 & a_{-1} & \ddots & \vdots \\
a_2 & a_1 & a_0 & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
a_{n+1} & a_n & a_{n-1} & \ldots & a_0
\end{bmatrix}
\]

\[
\frac{dy_1}{dx} = f_1(x, y_1, y_2, \ldots, y_n)
\]

\[
\frac{dy_2}{dx} = f_2(x, y_1, y_2, \ldots, y_n)
\]

\[
\vdots
\]

\[
\frac{dy_n}{dx} = f_n(x, y_1, y_2, \ldots, y_n)
\]

Matrix Analysis  Diff. Equations  Signal Analysis  Control Design

Unified visual math environment for all kind of math.
solidThinking Compose

IDE
(Integrated Development Environment)

OML
Matlab
Octave

Python

SciPy

HW-
Integration
Material Modeling for Crash Analysis
```matlab
if (y1 == y2)
    x_val = x;
    break
else
    dy = y1 - y2;
    dya = abs(dy);
    if (dya > dyn)
        if (dx > dyn)
            y_val = k1*x;
            x_val = x;
            break
        else
            dyn = dy;
        end
    end
end
RF = sqrt(x_val^2 + y_val^2)/sqrt(Rs^2+Rz^2)
end

function RF = HS8FastenerSizing(Fx,Fy,Fz,FC_allowable,Fa_allowable,k,curve)
    % Main function
    Fs = (Fy^2+Fz^2)^.5;
    Rs = abs(Fs/Fa_allowable);
    Rz = abs(Fs/(FC_allowable*k));
    % Curves are
    RF = 8.42224733
    ans = 8.42224733
end
```

Develop Function in Compose
Aerospace Certification
Why Compose?

• Best of multiple worlds
  • Leverages past investment in Matlab
  • CAE readers from HyperWorks
  • Supports Python, as well

• Superior business model
  • Available under HWUs
  • Accessible to everybody at your organization
  • One module for various applications

• Non-disruptive adoption
  • Can co-exist with Matlab
solidThinking Activate

Model & Simulate Dynamic Systems

Signal Based Modeling

Physical Modeling

Block diagram environment for Model Based Development & Multi-domain simulation
System Integration Platform

- Controls
- Modelica
- Multibody Dynamics
- Electro-magnetics
Why Activate?

• Signal and physical modeling in one environment
  • Leverages past investment in Simulink
  • Leverages the current trend towards physical modeling with Modelica
  • Provides superior simulator coupling via Functional Mock-Up Interface (FMI)

• Seamless transition to/from 3d-modeling

• Superior business model
  • Available under HWUs
  • Accessible to everybody at your organization
  • One module for various applications

• Non-disruptive adoption
  • Can co-exist with Simulink
Optimizing Acoustic Performance of Mufflers

1 iteration per day
1D Transfer Matrix Method

\[
\begin{bmatrix}
 p_0 \\
 v_0
\end{bmatrix} = 
\begin{bmatrix}
 A_1 & B_1 \\
 C_1 & D_1
\end{bmatrix} 
\begin{bmatrix}
 A_2 & B_2 \\
 C_2 & D_2
\end{bmatrix} 
\begin{bmatrix}
 A_3 & B_3 \\
 C_3 & D_3
\end{bmatrix} 
\begin{bmatrix}
 A_4 & B_4 \\
 C_4 & D_4
\end{bmatrix} 
\begin{bmatrix}
 A_5 & B_5 \\
 C_5 & D_5
\end{bmatrix} 
\begin{bmatrix}
 p_6 \\
 v_6
\end{bmatrix}
\]

\[
TL = 10 \log_{10} \left( \frac{1}{4} \left|\begin{array}{cc}
 A + B \frac{S}{\rho c} + C \frac{\rho c}{S} + D
\end{array}\right|^2 \right).
\]
Realization in Activate

Library of fundamental blocks

```
10;0.1 --> Pipe --> Extended_Inlet --> Pipe
```

Extended_Inlet

TL

Pipe

Sud. Discntuity

Get My T

Scope

50

533

50
Validation

**Validation**

- **seconds**
- **1 day**

![Graph showing transmission loss](image)

**Math Code (TMM)**
**Simulation - BEM**

![Graph showing numerical and experimental results](image)

**Experimental**
Typical Configurations Solved

Transmission Loss (dB)

Freq (Hz)

Math Code: TMM  Simulation: BEM

Transmission Loss (dB)

Freq (Hz)
Optimization of Muffler Configurations

After optimization

After optimization
Case Study: e-Mobility
Series-Parallel HEV Powertrain
Engine Modeling Methods

- Analytic approaches (Park model) – Activate only (no Flux)
  - FAST! Least accurate...

**Electrical equations:**

\[
V_d = R_s * I_d + \frac{d\psi_d}{dt} - \omega * \psi_d \\
V_q = R_s * I_q + \frac{d\psi_q}{dt} + \omega * \psi_d
\]

**Magnetic equations:**

\[
\psi_q = I_q * L_q \\
\psi_d = I_d * L_d + \psi_{PM}
\]

**Magnetic Torque:**

\[
\Gamma = \frac{3}{2} p (\psi_d * I_q - \psi_q * I_d)
\]

Ld, Lq and Phi_PM are calculated using Flux or by physical testing
Engine Modeling Methods

• Full co-simulation Flux + Activate
  • BEST ACCURACY! But slowest...

Output:
• Current
• Speed
• Torque…

Voltage to fed the motor

Link to FE Motor model
Engine Modeling Methods

- Table-based model from finite element results—FAST and IMPROVED ACCURACY vs. Park

Flux in a, b, c axis and magnetic torque are computed in term of Id, Iq and rotor position using Flux 2D.
Engine Modeling Methods – Motor Torque Comparison
Summary

• Open System Approach
  • Octave/Matlab, Tcl/Tk & Python scripting
  • Modelica
  • FMI
  • CAE Readers/Writers
  • Support of Legacy Models (Matlab/Simulink)

• Driving Innovation through Simulation
  • From Concept Design to HIL and SIL
  • Combining Math, Signal-Based, Physical & 3D Modeling

• Unique Business Model
  • Flexibility
  • Affordability
Getting Started - Functions
Defining functions, variable length argument lists; use of nargout, nargin, varargout, global, and feval.

Getting Started - Basic Plot Commands
Plot, subplot, grid, linewidth

Getting Started - Line Attributes
Color and marker codes; line types

Getting Started - Plot Types
2D and 3D plots: plot, scatter, plot3, surf, contour, contour3, waterfall, polar, semilog, loglog, area, bar

Getting Started - Loops
For and While loops; illustrated by calculating the Vandermonde Matrix and the Kronecker Product

Getting Started - Flow of Control
Controlling which statements should be executed by using break, switch, try/catch.
Accelerometer modeling

Project Files

Download Accelerometer.scm (107 KB)

View all episodes
Model-Based Development Webinar Series

In this age of the internet of Things, Big Data, analytics and mechatronics it is the multi-disciplinary, multi-control systems that provide the product and experience opportunities the model-based design. Altair’s solidThinking modeling solutions can be applied right from concept studies through full simulation to controller implementation & testing.

Design, analyze and develop smart systems with Altair’s solidThinking model-based engineering tools to math with signal-based, physical and 3D modeling to offer a complete toolset that supports the latest trends in electro-mechanical system development.

Join one of our live webinars

- **JULY 26**
  - Model-Based Development with Altair

- **AUGUST 2**
  - Modeling Dynamic Systems in Activate 2017

- **AUGUST 9**
  - Compose 2017: A Multi-Language Math Environment

http://web2.altairhyperworks.com/2017-model-based-development