Guest
Lecture: Antwerpen

Gareth Thomas
Rachid Adarghal
Focal Points

- With MATLAB/Simulink my professors think I am clever
- Knowing MATLAB/Simulink will help you get a job!
- Multi-Modeling Techniques are often needed
Motivation

- With MATLAB/Simulink my professors think I am clever
  - The tools will make your life easier.

- Knowing MATLAB/Simulink will help you get a job!
  - Put it on your CV, as you will encounter this after your degree

- Multi-Modeling Techniques are often needed
  - The real world is so complex, the solution comes from combining multiple domains
Introduction to the Speaker

Gareth Thomas

- Masters in Electronic Engineering at Instituto Superior Técnico
  Control Theory and Signal Processing
- Consultant at Altran CIS in Portugal
- Innovation Officer at Nokia Siemens Networks in Portugal
- Software Engineer at Oceanscan in Scotland
- Application Engineer at Mathworks Benelux
MathWorks Products

- How Many Toolboxes/Blocksets do you use?
  - 1 – 10
  - 10 – 15
  - 15 – 30
  - 30 – 60
  - >60

- How many toolboxes/Blockset do we offer?
  - 30 – 40
  - 40 – 50
  - 50 – 60
  - 60 – 70
  - >70
  - 96!
MathWorks Products

Fixed-Point Modeling
Event-Based Modeling
Physical Modeling
Rapid Prototyping and HIL Simulation
Verification, Validation, and Test
Simulation Graphics and Reporting
Applications
Control Systems
Signal Processing and Communications
Image Processing and Computer Vision
Test and Measurement
Computational Finance
Computational Biology
Parallel Computing
Code Generation
MATLAB®
The Language of Technical Computing
Math, Statistics, and Optimization
Application Deployment
Database Access and Reporting
MathWorks Vital Statistics

Developers of MATLAB & Simulink
2,500 staff worldwide
Support staff worldwide
Development staff in Natick, MA
30% of revenue invested in R&D
$500M annual revenue

2009 - orders from 23,000 companies in 128 countries
Revolution in Engineering Education

5000+ universities worldwide use MATLAB

> *Includes all of the Top 200 World Universities*

More than 1 million students and faculty have access to MathWorks tools through campus-wide licenses

> *More than 130 academic institutions, including 10 of the Top 20 World Universities*

Over 1400 MATLAB based books in 27 languages

* Source: Times Higher Education-QS World University Rankings 2009
Summer of 95!
Power Window Video
Power Window System

Switches

Control System

Armature Current

H-Bridge

DC Motor

Window Mechanism

Stateflow

SimMechanics

SimElectronics

SimElectronics

SimElectronics
Steps Taken

- Define Problem (Requirements)
- Model Plant (window)
- Model Controller
- Test System - Simulation
- Generate C-Code, implement it
- Test System – Real Window

- Model-Based Design
Power Window: Modeling the Plant

H-Bridge

DC Motor

Window Mechanism
Power Window: Defining the Controller

- Event-Based Control
  - For systems that change mode based on events
  - Examples
    - Automatic transmission
    - Power window
  - Best modeled in Stateflow

- Compensator Design
  - For systems where actuation is based on deviation from a commanded value (e.g. PID)
  - Examples
    - Robot position
    - Motor speed
  - Best modeled with Simulink Control Design and other control design tools
Defining the Controller: Inputs

- Input to controller are switches

1) Driver’s side switch has precedence over passenger switch
2) If no switches are closed, movement of window is defined based on history

Stateflow Truth Table

<table>
<thead>
<tr>
<th>Condition Table</th>
<th>Description</th>
<th>Condition</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>in.driver_up</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>in.driver_down</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>in.passenger_up</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>T</td>
<td>F</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>in.passenger_down</td>
<td>-</td>
<td>-</td>
<td>F</td>
<td>T</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Actions: Specify a row from the Action Table</td>
<td>1 2 1 2 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action Table</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Move Up</td>
<td>send(UP,Logic);</td>
</tr>
<tr>
<td>2</td>
<td>Move Down</td>
<td>send(DOWN,Logic);</td>
</tr>
<tr>
<td>3</td>
<td>Stay Neutral</td>
<td>send(NEUTRAL,Logic);</td>
</tr>
</tbody>
</table>
Defining the Controller: States

Stateflow Chart
Possibilities for Compensator Design

- **Linear Control Theory**
  - Linearize system using Simulink Control Design
  - Perform linear control design with Control System Toolbox
  - Test controller in nonlinear system

- **Specify System Response**
  - Specify response characteristics
  - Automatic tuning using Simulink Design Optimization
Compensator Design on Nonlinear Plants

Model:

\[ \frac{(K_p S + K_i)}{S} \]

Problem: Design and tune the controller in this system to meet system requirements

Solution: Use Simulink Design Optimization to design, tune, and test the controller

<table>
<thead>
<tr>
<th>( K_p )</th>
<th>( K_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.01</td>
<td>0.291</td>
</tr>
</tbody>
</table>
Compensator Design on Nonlinear Plants

- **Steps to Optimizing Response**

1) Identify parameters to be tuned and their ranges

2) Specify desired response

3) Perform response optimization

\[ (K_p s + K_i) \frac{s}{s} \]
Power Window: Generate Test Cases
Power Window: Generate C/C++ Code
Power Window: Test System in Real Life
Accelerating the Pace of Engineering and Science

Reduce the effort to:
- Translate your thoughts for the computer
- Interpret the results from the computer
Accelerating the Pace of Engineering and Science

Similarities between Industries:

- Matrix / vector based mathematics
- Standard and specific operations
- Specific analysis charts
Accelerating the Pace of Engineering and Science

- Direct use of matrix equations
Accelerating the Pace of Engineering and Science

- Direct use of matrix equations

\[ y = Ax + b \]
Accelerating the Pace of Engineering and Science

- Direct use of matrix equations

**MATLAB Code**

```matlab
function y=MatrixEquation(A,x,b)
y=A*x+b;
```
Accelerating the Pace of Engineering and Science

- Direct use of matrix equations
- Interactive - immediate response

MATLAB Code
```matlab
function y=MatrixEquation(A,x,b)
    y=A*x+b;
```

C Code
```c
void MatrixEquation(float A[100], float x[10], float b[10], float y[10])
{
    int32 i0;
    float d0;
    int32 i1;
    for(i0 = 0; i0 < 10; i0++) {
        d0 = 0.0;
        for(i1 = 0; i1 < 10; i1++) {
            d0 += A[i0 + (i1 << 1)] * x[i1];
        }
        y[i0] = d0 + b[i0];
    }
}
```
Accelerating the Pace of Engineering and Science

- Direct use of matrix equations
- Interactive - immediate response
- Built-in engineering functions

MATLAB Code

```
function y=MatrixEquation(A,x,b)
y=A*x+b;
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>y</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>455</td>
<td>92</td>
<td>99</td>
<td>1</td>
<td>8</td>
<td>15</td>
<td>67</td>
<td>74</td>
<td>51</td>
<td>58</td>
<td>40</td>
<td>0.5</td>
<td>-1.3</td>
</tr>
<tr>
<td>3</td>
<td>447</td>
<td>98</td>
<td>80</td>
<td>7</td>
<td>14</td>
<td>16</td>
<td>73</td>
<td>55</td>
<td>57</td>
<td>64</td>
<td>41</td>
<td>1.8</td>
<td>3.0</td>
</tr>
<tr>
<td>4</td>
<td>284</td>
<td>4</td>
<td>81</td>
<td>88</td>
<td>20</td>
<td>22</td>
<td>54</td>
<td>56</td>
<td>63</td>
<td>70</td>
<td>47</td>
<td>-2.3</td>
<td>0.7</td>
</tr>
<tr>
<td>5</td>
<td>431</td>
<td>85</td>
<td>87</td>
<td>19</td>
<td>21</td>
<td>3</td>
<td>60</td>
<td>62</td>
<td>69</td>
<td>71</td>
<td>28</td>
<td>0.9</td>
<td>-0.1</td>
</tr>
<tr>
<td>6</td>
<td>362</td>
<td>86</td>
<td>93</td>
<td>25</td>
<td>2</td>
<td>9</td>
<td>61</td>
<td>68</td>
<td>75</td>
<td>52</td>
<td>34</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>7</td>
<td>212</td>
<td>17</td>
<td>24</td>
<td>76</td>
<td>83</td>
<td>90</td>
<td>42</td>
<td>49</td>
<td>26</td>
<td>33</td>
<td>65</td>
<td>-1.3</td>
<td>-0.2</td>
</tr>
<tr>
<td>8</td>
<td>199</td>
<td>23</td>
<td>5</td>
<td>82</td>
<td>89</td>
<td>91</td>
<td>48</td>
<td>30</td>
<td>32</td>
<td>39</td>
<td>66</td>
<td>-0.4</td>
<td>-0.1</td>
</tr>
<tr>
<td>9</td>
<td>461</td>
<td>79</td>
<td>6</td>
<td>13</td>
<td>95</td>
<td>97</td>
<td>29</td>
<td>31</td>
<td>38</td>
<td>45</td>
<td>72</td>
<td>0.3</td>
<td>1.5</td>
</tr>
<tr>
<td>10</td>
<td>189</td>
<td>10</td>
<td>12</td>
<td>94</td>
<td>96</td>
<td>78</td>
<td>35</td>
<td>37</td>
<td>44</td>
<td>46</td>
<td>53</td>
<td>3.6</td>
<td>1.4</td>
</tr>
<tr>
<td>11</td>
<td>119</td>
<td>11</td>
<td>18</td>
<td>100</td>
<td>77</td>
<td>84</td>
<td>36</td>
<td>43</td>
<td>50</td>
<td>27</td>
<td>59</td>
<td>2.8</td>
<td>1.4</td>
</tr>
</tbody>
</table>
Data Analysis Tasks

Access
- Files
- Software
  - Code & Applications
- Hardware

Explore & Discover
- Data Analysis & Modeling
- Algorithm Development
- Application Development

Share
- Reporting and Documentation
  - PDF
  - .doc
  - .html
- Outputs for Design
- Deployment
  - MATLAB
  - Excel
  - .NET
  - C/C++
  - Java
  - .dll

Automate
What is the value of their/your engineering department?

- Creativity/Innovation: bringing new ideas into practice
- Knowledge/Experience: knowing what will work and what will not
Accelerating the Pace of Engineering and Science with MATLAB & Simulink

Reduce the effort to
• translate your thoughts for the computer
• interpret the results from the computer
How many Ports and Subsystem are there?

27!!
What is Simulink?

- **Explains Reality (Communication)**
  A description using basic principles that has some predictive value about behavior

- **Specifies Reality (Design)**
  A description using basic principles that specifies desired behavior

- **Replaces Reality (Simulation)**
  A description that has some predictive value about the behavior of the real thing
What is new in Simulink?

- Simulink Projects
- Data Inspector
- Comparing XML
- Concurrent execution
- Modeling Task
- Model Variants
- Subsystem Variants
- Model Explorer Improved
- Logging to Datasets

- Comparing Files/Folders
- Parallel Builds
- Model Advisor
- Export to Web
What is new in Simulink Blocksets?

- Formal Methods
  - Design Errors
  - Property Proving
  - Test Generation
- Code Generation
- Physical Modeling
- Fixed-Point word scaling
- SimEvents
- Simulink Code Inspector
- xPC Target
- SIL Performance

- Linking requirements to word via external file
- Export to web

READ Release Notes!!!
A better workflow to implementation

- **One** language
  - No multiple copies of source code
  - Integrate real-world design constraints in MATLAB

- **One** integrated design environment
  - Integrated visualization, analysis & debugging

- **Automatic** code generation
  - Path to embedded software (Embedded C)
  - Path to FPGA/ASIC (HDL)
Introduction to Physical Modeling
System-level Modeling

- Model the dynamics that matter for your analysis
- Balance cost and model fidelity
Modeling Dynamic Systems: two approaches

First-Principles Modeling

Use an understanding of the system’s physics to derive a mathematical representation

\[ J_1 \ddot{x}_1 = \sum \text{Torques} = -b_1 \dot{x}_1 - k(x_1 - x_2) - b_{12} \dot{x}_1 \dot{x}_2 + T \]

\[ J_2 \ddot{x}_2 = \sum \text{Torques} = -b_2 \dot{x}_2 + k(x_1 - x_2) + b_{12} \dot{x}_1 \dot{x}_2 \]
Modeling Dynamic Systems: two approaches

First-Principles Modeling

Use an understanding of the system’s physics to derive a mathematical representation

Data-Driven Modeling

Use system test data to derive a mathematical representation

\[ H(z) = \frac{1.22z^3 - 0.62z^2 - 0.78z - 1.27}{z^4 - 3.55z^3 + 5.08z^2 - 3.52z + 0.98} \]

Measured Signals

\[ \text{time [s]} \]

\[ \text{response [V]} \]

-0.01
-0.005
0
0.005
0.01

noise

u1

u2
Both have Advantages & Disadvantages

Advantages:
- Insight in behavior
- Physical parameters

Disadvantages:
- Friction and turbulence?
- Time consuming
- Requires expertise

Advantages:
- Fast
- Accurate

Disadvantages:
- Requires plant
- Requires data acquisition system
Tools that span both modeling approaches
Enhance Advantages, Reduce Disadvantages

Complete Modeling Environment

First-Principles
Simulink
Stateflow
Simscape
SimMechanics
SimDriveline
SimHydraulics
SimElectronics
SimPowersystems

Data-Driven
Simulink Design Optimization
System Identification Toolbox
Test & Measurement Tools
SimMechanics: accurate modeling of 3D mechanical systems

✓ 3D Multi-Body Dynamics
✓ Bodies and Joints
✓ CAD Translation
‘Simple’ example: double pendulum

Traditional approach:

Derivation of the equations of motion requires knowledge and effort.

\[
\begin{align*}
\theta_1'' &= \frac{-g (2m_1 + m_2) \sin \theta_1 - m_2 g \sin(\theta_1 - 2 \theta_2) - 2 \sin(\theta_1 - \theta_2) m_2 (\theta_2'^2 L_2 + \theta_1'^2 L_1 \cos(\theta_1 - \theta_2))}{L_1 (2m_1 + m_2 - m_2 \cos(2 \theta_1 - 2 \theta_2))} \\
\theta_2'' &= \frac{2 \sin(\theta_1 - \theta_2) (\theta_1'^2 L_1 (m_1 + m_2) + g(m_1 + m_2) \cos \theta_1 + \theta_2'^2 L_2 m_2 \cos(\theta_1 - \theta_2))}{L_2 (2m_1 + m_2 - m_2 \cos(2 \theta_1 - 2 \theta_2))}
\end{align*}
\]
Traditional approach: Derivation of the equations of motion requires knowledge and effort.

'Simple' example: double pendulum

\[ \theta_1'' = \frac{-g(2L \sin \theta_1 - L \sin \theta_2 \cos \theta_1)}{L \sin \theta_1} \]

\[ \theta_2'' = \frac{-g(L \sin \theta_1 - 2L \sin \theta_2 \cos \theta_1)}{L \sin \theta_2} \]
With SimMechanics
Power Window Example
Introduction to Modeling States
When should I use MATLAB?

- Next step for traditional programming (4th Generation)
- Quick and powerful (dedicated) visualization
- Simple C code Generation is possible
- Deployment
- Task Automation
- Data Analysis
When should I use Simulink?

- System level overview
  - Signal flow/Block diagram representation
  - Architecture/Hierarchy definition
- Multi Rate/Multi Domain System
  - Mixed Signals
  - Physical Models
- Advanced Code Generation
- Certification – Model Based Design Support
  - Model Coverage, requirement traceability, formal proving, modelling standards....
When should I use Stateflow?

- Control Logic
  - State Machine
  - Discrete Events

- Scheduling
  - Drive Simulink
  - Control flow Programming

- Mode Switching

- Fault Management
User Case – Simple filter, Image algorithm, GUI, $y = A^x + B$, visualization
User Case – Fixed Point Development

MATLAB

Simulink

Stateflow
User Case – Certification, System overview

MATLAB

Simulink

Stateflow
User Case – Multi-Domain, Mixed signals, Multi-Rate, ...
User Case – Concurrent States
User Case – Nested if then else
User Case – if then else

MATLAB

Simulink

Stateflow
Having the Choice is the Real Value!

MATLAB

VALUE

Simulink

Stateflow
What is Stateflow?

Extend Simulink with state charts and flow graphs

Design supervisory control, scheduling, and mode logic

Model state discontinuities and instantaneous events
How Does Stateflow Work with Simulink?

Simulink models **continuous** changes in dynamic systems.

Stateflow models **instantaneous** changes in dynamic systems.

Real-world systems have to respond to both continuous and instantaneous changes.

- suspension dynamics
- gear changes
- propulsion system
- liftoff stages
- manufacturing robot
- operation modes

*Use both Simulink and Stateflow so that you can use the right tool for the right job.*
Stateflow Concepts

States
- Exclusive
- Hierarchical
- Parallel

Transitions
- Default
- Conditions
- Condition Actions
- Event Triggers

Functions
- Graphical
- Truth Tables
- MATLAB

Data
- Input/Output
- Local
- Model Explorer
- Add Menu
- Symbol Wizard
What Is a Flow Graph?

A chain of logical patterns that implement a series of decision flows

- **if-else** construct
- Nested **while** loop

Can implement sequential, nested, and iterative flows
Junctions and Transitions

Default transition

Transitions

Terminating junction

Junctions
Conditions and Actions

**Conditions**

1. \( \text{failFlag} == 1 \)
2. \( \text{count} > 0 \)

**Actions**

1. \( \text{count} = 1; \)
2. \( \text{out} = \text{data}; \)
3. \( \text{out} = 0; \)
4. \( \text{count} = \text{count} + 1; \)
5. \( \text{count} = 0; \)
What Is a State Machine?

- A system that can only exist in a finite number of modes
- Can only behave in a predefined number of ways
States and Transitions

**NormalOperation**

/*output = input*/
entry:
out = data;
during:
out = data;

/*When failure occurs*/
[failFlag == 1]

/*After 5 cycles with no failure*/
[count == 5]

**FailureOccurred**

/*output = 0*/
entry:
count = 1;
out = 0;
during:
count++; out = 0;

/*Reset counter*/
[failFlag == 1]
The Concepts of Parallelism

- Parallel states enter when their parent activates.
- Transitions from or to parallel states are prohibited.
An Example of Stateflow® Events

```
digit1
  inputClick
  zero
    en: d1 = 0;
  one
    en: d1 = 1;
  two
    en: d1 = 2;

digit2
  inc
  zero
    en: d2 = 0;
  one
    en: d2 = 1;
  two
    en: d2 = 2;

digit3
  inc
  zero
    en: d3 = 0;
  one
    en: d3 = 1;
  two
    en: d3 = 2;
```
Using Events to Trigger Actions

• Guard transitions

• Perform state actions (`on` keyword)
Broadcasting Events

• Use the event name to broadcast the event.

• This can be done anywhere that actions are specified (state actions, condition actions, and transition actions)
Power Window Example
Introduction to Verification Modeling
Where does engineering go wrong…

Design Error?

Missing Test?
Why?

- Traceable
- Workflow
- Formal Methods
- Tests
- Scalable
- Version control
- Control over process
- Source code
- Style structure
- Coding standards
- Modeling standards
- Industry standards
- Formal methods
- Behavior
- Run-time
- Tests for run-time
- Tests
- Results, performance
- Model and code coverage
- Validation
- Simulation
- Certification
- Reports
- Metrics
- Certification
- Reports
- Quality
Why?

- Traceable
- Formal methods
- Verification
- Validation
- Model and code coverage
- Results, performance
- Tests
- Metrics
- Workflow
- Tests
- Simulation
- Behavior
- Run-time
- Tests for run-time
- Formal methods
- Industry standards
- Scalable
- Version control
- Control over process
- Source code
- Style structure
- Coding standards
- Modeling standards
- Certification
- Reports
- Quality
- CEO
- Senior management
- Quality organization
Why?

- Quality
  - Source code
  - Control over process
  - Scalable
  - Version control
- Certification
  - Reports
  - Model and code coverage
  - Results, performance
  - Tests
  - Formal methods
  - Metrics
  - Industry standards
- Workflow
  - Tests
  - Simulation
  - Behavior
  - Run-time
  - Tests for run-time
  - Formal methods
- Traceable
  - Verification
  - Validation
  - Results, performance
  - Reports
  - Certification

Engineer
Decision Coverage (DC)

Percentage of paths taken through decision point

if \((X \& Y)\)
\[ Z = 1; \]
else
\[ Z = -1; \]
end
Condition Coverage (CC)

Percentage of conditions exercised

if (X & Y)
    Z = 1;
else
    Z = -1;
end
Modified Condition/Decision Coverage (MC/DC)

Checks inputs independently affect output

if (X & Y) Z = 1;
else Z = -1;
end

Affects (X & Y) to be T and F?
Signal Range Coverage

Signal Ranges:

<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>LookupCovExample</td>
<td>7.00384</td>
<td>18.9998</td>
</tr>
<tr>
<td>Switch</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>Constant</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Constant1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Sine Wave</td>
<td>-0.996165</td>
<td>0.999793</td>
</tr>
<tr>
<td>Step</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Lookup Table Coverage (LUT)

Interpolation interval

Exact values

Click graph for range information
What Does Coverage Tell You

- **Useful information:**
  - How much of my system did testing explore?
  - How complete are someone else’s tests?
  - How much testing has a team done on a model?
  - Is there a part of the model that is hard or impossible to reach?
  - If using code generation, what tests are needed for the final code?
  - If I know what the expected behaviour is, did I see any violations whilst achieving coverage?

- **What it isn’t:**
  - Coverage testing helps find unintended function, but doesn’t test for correct function on its own.
  - A good starting point, but additional tests needed for full source and object code coverage.
Power Window Example

- Generate Test Cases Based on Coverage
- Generate Test Cases Based on Properties
Why?

- Traceable
- Workflow
- Formal methods
- Tests
- Scalable
- Version control
- Control over process
- Source code
- Style structure
- Coding standards
- Modeling standards
- Industry standards
- Formal methods
- Simulation
- Behavior
- Run-time
- Tests for run-time
- Model and code coverage
- Results, performance
- Tests
- Certification
- Reports
- Quality
- Project manager
Why?

- Traceable
- Workflow
- Formal methods
- Tests
- Scalable
- Version control
- Control over process
- Source code
- Style structure
- Coding standards
- Modeling standards

- Reports
- Certification
- Model and code coverage
- Results, performance
- Tests for run-time
- Tests
- Formal methods
- Industry standards

- Validation
- Simulation
- Behavior
- Run-time
- Quality

- Metrics
- Industry standards
Why?

TO KEEP EVERYONE HAPPY!

- CEO, senior management
  - Want to improve quality for product
  - Yet need to remain competitive in price, time-to-market, feature content
- Project manager
  - Has to give reports to quality engineer
  - Has to make his team comply with standards
  - Creates more work
- Engineer
  - Has more work
  - Pushes back because need is not clear
- Quality engineer
  - Needs to work with everyone and bring them on board!
  - Finds it hard
Model Transformation is Key
Model Transformation is Key

- Automatic Code Generation
  - From Simulink to:
    - C
    - C++
    - Structured Text
    - Verilog
    - VHDL
  - From C/C++/ADA/Verilog/VHDL to:
    - Simulink
Simulink Code Inspector

Model and code development

Independent code inspection

- Static verification tool that checks the generated code against model
- Automates DO-178B Table A-5 verification activities
- Technology allows seamless upgrades to new releases

IR: Intermediate representation
How can you prove that no error occurred? What is Abstract Interpretation?
Example of lattice for variables values: Signs

Levels of Abstraction

More abstract

More concrete

More concrete
Example of abstraction: Sign

```c
volatile int random;
int x=0, y=0;
if (random) {
    x++;
    y--;
} else {
    x += 2;
    y += 1;
}
assert(y > 0);
assert(x > 0);
```

- $x: =0, y: =0$
- $x: >0, y: =0$
- $x: >0, y: <0$
- $x: >0, y: =0$
- $x: >0, y: >0$
- $x: >0, y: \text{Top}$

- $y: \text{Top} \rightarrow \text{Orange}$
- $x: >0 \rightarrow \text{Green}$
PolySpace Products for Code Verification

- **Quality improvement**
  - Prove the absence of run-time errors in source code
  - Measure, improve, and control

- **Usage**
  - Simple colored source code
  - No compilation, no execution, no test cases
  - For C/C++ or Ada

- **Process**
  - Run early in development cycle
  - Use for automatically generated and handwritten code

```c
static void Pointer_Arithmetic (void)
{
    int array[100];
    int i, *p = array;

    for(i = 0; i < 100; i++, p++)
        *p = 0;

    if (get_bus_status() > 0) {
        if (get_oil_pressure() > 0)
            *p = 5;
        else
            i++;
    }

    i = get_bus_status();
    if (i >= 0) {
        *(p-i) = 10; }

    if ((0 < i) && (i <= 100)) {
        p = p - i;
        *p = 5;
    }
}
```
How can you prove that no error occurred?

Verifying $x = x / (x - y)$

- Potential run-time errors:
  - Are $x$ and $y$ initialized?
  - Could a division by 0 occur?
  - Could there be an underflow or overflow on ‘-’, ‘/’, or ‘=’?

The following slide focuses on the check for a division by 0.
No execution
No simulation
No test cases to write

Verifying $x = x // (x - y)$

Type analysis
Abstract interpretation
What color is your code today?
How do you prove code correctness?

- Number of operations x Input values
- 0% proven reliable
- PolySpace

Static analyzers

Testing

T0
unknown
unknown

T0 + 3 months
unknown
unknown

T0 + 6 months
unknown
unknown

Nothing proven

Required for functional testing…
…not suitable to prove code correctness

CODE CORRECTNESS
Challenge…

- Why is there red code here?

```c
static void Pointer_Arithmetic (void)
{
    int array[100];
    int i, *p = array;

    for(i = 0; i < 100; i++)
    {
        *p = 0;
        p++;
    }

    if(get_bus_status() > 0)
    {
        if(get_oil_pressure() > 0)
        {
            *p = 5;
        }
        else
        {
            i++;
        }
    }
}
Challenge …

- Why is there red code here?

```c
static void Square_Root_conv (double alpha, float *beta_pt)
/* Perform arithmetic conversion of alpha to beta */
{
    *beta_pt = (float)((1.5 + cos(alpha))/5.0);
}

static void Square_Root (void)
{
    double alpha = random_float();
    float beta;
    float gamma;

    Square_Root_conv (alpha, &beta);

    gamma = (float)sqrt(beta - 0.75);
}
```

-1 <= Cos(alpha) <= 1

Worst case: 0.5/5 = 0.1

Worst case: sqrt(0.1 - 0.75) is a run time Error
Takeaways

- **Challenge**
  - Prove absence of run time errors
  - Code reviews takes a long time
  - Coding standards
  - Testing is an ambiguous word, depends how it is implemented

- **Suggestion**
  - PolySpace can help
  - Formally prove absence of run time errors
  - Create reports on how well your code is tested
Putting it all together: Model-Based Design
History of Software Development

- Challenge appears… and so do proposed solutions.

Code Base Grows

Version Control

Coding Standards

Industry Standards
History of Software Development - Challenges

- Multiple domains come together
- Software teams grow
- Code base grows
- Processing power increases
- New programming languages
- Industries grow
- Finding the right people
- Time pressure
- Auditing to protect consumer
  Technology grows faster than population
- Quality is more important
- New hardware platforms
- Companies grow
- Budget restrictions
- Companies grow
Software Development - Concepts

- **Multiple domains come together**
- **Industries grow**
- **Code base grows**
- **Processing power increases**
- **New programming languages**
- **Software teams grow**
- **Quality is more important**
- **Finding the right people**
- **Time pressure**
- **Auditing to protect consumer**
- **Technology grows faster than population**
- **New hardware platforms**
- **Companies grow**
- **Budget restrictions**
- **Coding standards**
- **Industry standards**
- **Translation between languages**
- **Version control**
- **Task automation**
- **Traceability and validation**
- **Separate hardware from software**
- **Simulation**
- **Certification**
- **Reports**
- **Model-Based Design**
- **Task automation**
- **Traceability and validation**
- **Simulation**
- **Certification**
- **Reports**
- **Model-Based Design**
Expensive to fix errors found late in the process

“…each delay in the detection and correction of a design problem makes it an order of magnitude more expensive to fix…”

Source: “Migration from Simulation to Verification with ModelSim®” by Paul Yanik. EDA Tech Forum, 2004 Mar 11, Newton MA

Clive Maxfield and Kuho Goyal
“EDA: Where Electronics Begins”
TechBites Interactive, October 1, 2001
ISBN: 0971406308
Moore’s Law

CPU Transistor Counts 1971-2008 & Moore’s Law

Curve shows Moore’s Law: transistor count doubling every two years.
Moore’s Law

Number of Employees: Increase 10% every year
Problems with Traditional Development

Requirements and Specs

Text-based
Prevents rapid iteration

Prototypes
Incomplete and expensive

Manual coding
Introduces human error

Traditional testing
Errors found too late in the process

Design with prototypes

Manual Implementation

Test and Verification

MS Word
LaTeX
Visio
MS PowerPoint

MATLAB Code
C/C++ Code
High-Level Languages

VHDL
Verilog
C/C++

Simulation Test Bench
Hardware Test Environment
Advantages of Model-Based Design

**Executable models**
- unambiguous
- only “one truth”

**Simulation**
- Reduces physical prototypes
- systematic “what-if” analysis

**Test with Design**
- detects errors earlier

**Automatic code generation**
- minimizes coding errors

**Continuous Test and Verification**

**Design with Simulation**

**Models**

**Executable Specifications**

- Reduces physical prototypes
- systematic “what-if” analysis

- detects errors earlier

- minimizes coding errors
Adopting Model-Based Design

Requirements and Specs

Design with Simulation

Automatic code generation

Executable models
- Unambiguous
- Only “one truth”

Design with Simulation
- Reduces “real” prototypes
- Systematic “what-if” analysis

Automatic code generation
- Prevents coding errors

Test with Design
- Detects errors earlier

Model Elaboration

Continuous Verification

Test and Verification
Workflow for Model-Based Design

- Executable Specifications
- Continuous Test and Verification
- Design with Simulation
- Automatic Code Generation
Model-Based Design
Focal Points

- With MATLAB/Simulink my professors think I am clever

- Knowing MATLAB/Simulink will help you get a job!

- Multi-Modeling Techniques are often needed
Available Resources
Visit MathWorks Web Site

- Learn about MathWorks products
- Discover resources for learning, teaching, and research
- Learn how MathWorks products are used in academia and industry

Visit www.mathworks.com/academia
The MATLAB and Simulink product families are fundamental computational tools at the world's educational institutions. Adopted by more than 5000 universities and colleges, MathWorks products accelerate the pace of learning, teaching, and research in engineering and science. MathWorks products also help prepare students for careers in industry, where the tools are widely used for research and development.

Find out more in The MathWorks in Academia brochure.
Hardware for Project-Based Learning

Use MATLAB and Simulink with a variety of student-owned hardware to incorporate real-time processing in classroom labs.

**Arduino**
Student-priced microcontroller for introducing electrical engineering, motors, and mechatronics.

```matlab
>> a=arduino('COM5');
>> a.servoWrite(1,45);
>> a.motorSpeed(3,100);
>> a.motorRun(3,'backward');
```

**BeagleBoard**
Low-cost, single-board designed for audio and digital signal processing.

```matlab
>> a.digitalRead(4)
>> a.digitalWrite(5,0)
>> a.analogRead(6)
>> a.analogWrite(9,50)
```

**BEST Robotics**
Platform for high school competition based on Cortex microcontroller.

**dSPACE ACE Kit**
Controller boards ideal for developing and testing control systems.

Integrated platform for teaching hardware-in-the-loop, with analog I/O, digital I/O, and optional FPGA.

Visit www.mathworks.com/academia/hardware-resources/
MATLAB Central

- Open exchange for the MATLAB and Simulink user community
  - 1.2 million visits per month
- File Exchange
  - Upload/download free files including MATLAB code, Simulink models, and documents
  - Rate files, comment, and ask questions
- Newsgroup
  - Web forum and newsgroup for technical discussions about MATLAB and Simulink
- Blogs
  - Read posts from key MathWorks developers who design and build the products
Learning Resources

- **Interactive Video Tutorials** – Students learn the basics outside of the classroom with self-guided tutorials provided by MathWorks
  - MATLAB
  - Simulink
  - Signal Processing
  - Control Systems
  - Computational Mathematics

Visit www.mathworks.com/academia/student_center/tutorials
Recorded Webinars

Learn more about MathWorks products and how they help solve complex technical issues through these online recorded webinars. To view a free webinar, select a language and topic, and then click on the link and complete the request form.

Webinar RSS Feed

Language of Webinar: [English]

<table>
<thead>
<tr>
<th>Most Recent</th>
<th>By Application</th>
<th>By Product</th>
<th>Most Popular</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introduction to Curve Fitting for Nonprogrammers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introduction to Econometrics Toolbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developing a Financial Market Index Tracker with MATLAB OOP and Genetic Algorithms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Analysis with Statistics and Curve Fitting Toolboxes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Practices for Verification, Validation, and Test in Model-Based Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What’s New for Object-Oriented Programming in MATLAB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tips &amp; Tricks: Getting Started Using Optimization with MATLAB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Student Version R2011a

- MATLAB
- Simulink
- 7 popular add-on products
  - Control System Toolbox
  - Signal Processing Toolbox
  - DSP System Toolbox
  - Statistics Toolbox
  - Optimization Toolbox
  - Image Processing Toolbox
  - Symbolic Math Toolbox
MATLAB Mobile

MATLAB Mobile is a lightweight desktop on your iPhone that connects to a MATLAB session running on the MathWorks Computing Cloud or on your computer. From the convenience of your iPhone, you can run scripts, create figures, and view results.

Features
- Command-line access to MATLAB
- Access to MATLAB workspace
- Ability to view MATLAB figures on your iPhone
- Record of commands typed on the iPhone in your command history
- Custom keyboard
- MathWorks Computing Cloud connectivity
- Windows, Mac, and Linux connectivity

Limitations
- MATLAB Mobile does not support
  - Graphical user interfaces, such as SPTool and Curve Fitting Tool
  - MATLAB Editor
  - Simulink® graphical environment, but the `sim` command is supported at the MATLAB Mobile command line
  - Interaction with 2D and 3D figures

Don't see the download buttons?
Log in to your MathWorks Account or create an account now.

What's New in MATLAB Mobile 2.0
- MathWorks Computing Cloud connectivity

Resources
- Documentation
- MATLAB Answers
- Blog: Mike on the MATLAB Desktop
- Enhancement Request
MATLAB Mobile
Thank you for attention

Please fill out your evaluation form