Introduction to Simulink and Stateflow

GianCarlo Pacitti
Senior Application Engineer, MathWorks

MATLAB EXPO 2015
UNITED KINGDOM
Topics we will address this session

- Why do organisations use Simulink and Stateflow?
- Getting to grips with the basics of Simulink and Stateflow through a worked example
Why model a system?
OHB Develops Satellite Guidance, Navigation, and Control Software for Autonomous Formation Flying

Challenge
Develop low-cost satellite GNC systems to enable autonomous formation flying, rendezvous, and close-proximity operations with a small team

Solution
Use MathWorks tools for Model-Based Design to model GNC algorithms, perform real-time simulations, and generate production flight code

Results
- Development time cut by 50%
- Early verification and test reuse enabled
- Interagency collaboration simplified

“Traditionally, control engineers specify requirements using text and diagrams and someone else codes the software. With Model-Based Design, we have eliminated that step. We work with models from concept to implementation, and we have the automatically generated code flying in space.”

Ron Noteborn
OHB

Prisma’s Mango and Tango satellites in the sunbeam of the space simulator.
Modelling & Simulation gives you insight
Traditional Development Process

- **RESEARCH**
- **REQUIREMENTS**
- **SPECIFICATIONS**
- **DESIGN**
- **IMPLEMENTATION**
- **INTEGRATION AND TEST**

- **Requirement Documents**
  - Difficult to analyze
  - Difficult to manage as they change

- **Paper Specifications**
  - Easy to misinterpret
  - Difficult to integrate with design

- **Physical Prototypes**
  - Incomplete and expensive
  - Prevents rapid iteration
  - No system-level testing

- **Manual Coding**
  - Time consuming
  - Introduces defects and variance
  - Difficult to reuse

- **Traditional Testing**
  - Design and integration issues found late
  - Difficult to feed insights back into design process
  - Traceability
What is Model-Based Design?

- RESEARCH
- REQUIREMENTS
- DESIGN
  - Environmental Models
    - Mechanical
    - Electrical
  - Control Algorithms
  - Supervisory Logic
- IMPLEMENTATION
  - C, C++
  - VHDL, Verilog
  - Structured Text
  - MCU, DSP, FPGA, ASIC, PLC
- INTEGRATION
- TEST & VERIFICATION
Why use Simulink?
**Challenge**
Simplify the operation of forage harvesters by automating the process of filling trailers with corn, grass, and other crops

**Solution**
Use Model-Based Design to develop, test, and implement an automated control system that uses 3D camera data to position the filler spout

**Results**
- Development times halved
- Generated code immediately operational
- Industry innovation award won

---

**Link to user story**

"With Model-Based Design we spent most of our time developing and optimizing the system. Almost no time was spent implementing it in C or debugging code. There’s no difference in performance between the Simulink model running on a laptop and production code."

Karel Viaene
CNH
Using Simulink & Stateflow
Introduction to Simulink

- Block-diagram environment
- Model, simulate, and analyze multidomain systems
- Design, implement, and test:
  - Control systems
  - Signal processing systems
  - Communications systems
  - Other dynamic systems
- Platform for Model-Based Design
Model-Based Design Application

- Control the actuation of part of a robotic system
- Electric motor with speed control
What questions do we want to answer?

- Can I get the closed loop response I need?
- What current will my motor draw during operation?
- Does my system still work if component values change?
- What if…?
Steps in the process

1. Model the motor
2. Model the speed controller
3. Model the supervisory logic

At each stage: **Simulate the model**
PID Control of a DC Motor

\[ V = K \cdot \omega + i \cdot R + L \frac{di}{dt} \quad \Rightarrow \quad i = \frac{1}{L} \int (V - K \cdot \omega - i \cdot R) dt \]

\[ -T = K \cdot i - b \cdot \omega - J \frac{d\omega}{dt} \quad \Rightarrow \quad \omega = \frac{1}{J} \int (T + K \cdot i - b \cdot \omega) dt \]
Steps in the process

- Model the motor
- Model the speed controller
- Model the supervisory logic

At each stage: Simulate the model
Decision Flows and State Machines
Stateflow Overview

- Extend Simulink with a design environment for developing state machines and flow charts
- Design systems containing control, supervisory, and mode logic
- Describe logic in a natural and understandable form with deterministic execution semantics
What are State Machines?

- Represent reactive systems that have states or modes
- States change based on defined conditions and events

What are Flow Charts?

- Represent an algorithm or process
Modelling the system with Simulink and Stateflow
Next steps in the process

- Model the motor
- Model the speed controller
- Model the supervisory logic

4. Refine the motor model parameters using measured data
5. Deploy the control model to hardware

- Simulate the model

Visit the Demo Stations!
Conclusions

- Modelling and simulation gives you insight to make smarter decisions, earlier

- Simulink allows you to model the complete system in a single environment

- Accelerate your simulation work with the power of MATLAB