Advanced Rapid Prototyping in Small Research Projects with Matlab/Simulink

Rob Toulson
Anglia Ruskin University, Cambridge, UK
rob.toulson@anglia.ac.uk
www.robtoulson.com
Welcome

Summary:

1. Using Matlab for data analysis and algorithm development

2. Model based design with Simulink

3. Scripting efficiency with Matlab/Simulink

4. Advanced Matlab scripting

5. Auto-code generation for rapid prototyping

6. Conclude and recap
1. Using Matlab for data analysis and algorithm development

1.1 Matlab introduction

1.2 Data management and sharing data between Matlab and MS Excel (ex1)

1.3 The advantages of using Matlab for algorithm development

1.4 Case study: Development of an intelligent agricultural crop sorting system
1.2 Data management and sharing data between Matlab and MS Excel

Matlab’s Direct Data Exchange (dde) functions:

- Open communications with Excell with the ddeinit command
- Use the ddeinit command to specify a channel name for communication
  - eg. channel=ddeinit('excel','waveform_data.xls');

- Import data into Matlab with the ddereq command
  - eg. waveforms=ddereq(channel,'r1c1:r256c4');

- Export data from Matlab with the ddepoke command
  - eg. waveforms=ddereq(channel,'r1c1:r256c4');

Data can also be saved in Matlab’s .mat file formats.
1.3 The advantages of using Matlab for algorithm development

- High level language
  - quick visualisation of data
  - algorithm functions are simpler to code
  - quick visualisation and evaluation of results

- Algorithms can therefore be tested and evaluated in Matlab on test data, before being coded (C or Assembler) for an embedded device

- Integration with other programs and other Mathworks products

- Matlab toolboxes
1.4 Case study: Development of an intelligent agricultural crop sorting system
2. Model based design with Simulink

2.1 Simulink introduction

2.2 Physical system simulation (ex2)

2.3 Advantages of the model based design approach

2.4 Case study: Hybrid vehicle simulation research project
2.2 Physical system simulation (ex2)
2.3 Advantages of the model based design approach

- Obvious advantages of simulation to enable ‘right-first-time’ design

- A single engineer can work in design, coding, deployment and testing and move quickly between each process

- Less design documentation to keep up to date – the model IS the design flowchart

- Simulation models can be auto-coded (with Real Time Workshop) and deployed as hardware-in-the-loop plant models

- Control code can be auto-coded to C or embedded code for immediate deployment to an electronic control unit (ECU)
2.4 Case study: Hybrid vehicle simulation research project
3. Scripting efficiency with Matlab/Simulink

3.1 Using Matlab to auto-generate Simulink models (ex3)
   Case study: Integrated vehicle chassis control architecture

3.2 Developing generic and re-useable ‘masked’ subsystems
   Case study: Dynamically configured down counter (ex4)

3.3 Implementing Matlab and C code in Simulink s-functions
   Example CAN message converter (ex5)
3.1 Using Matlab to auto-generate Simulink models (ex3)

new_system and open_system commands open a Simulink model from the Matlab workspace, eg.
```matlab
new_system('SimulinkBuilderOutput');
open_system('SimulinkBuilderOutput');
```

add_block command creates Simulink blocks from the workspace, eg.
```matlab
add_block('built-in/PulseGenerator', 'SimulinkBuilderOutput/PulseInput')
```

set_param allows block values, position, appearance etc to be manipulated, eg.
```matlab
set_param('SimulinkBuilderOutput/PulseInput','Position',[50 50 100 100])
```

add_line command adds lines and data flow to the Simulink model, eg.
```matlab
add_line('SimulinkBuilderOutput','PulseInput/1','ScopeOutput/1');
```

replace_block and delete_block commands can also be useful for dynamically created models, eg.
```matlab
replace_block('SimulinkBuilderOutput','name','PulseInput','Constant','noprompt');
delete_block('SimulinkBuilderOutput/ScopeOutput');
```
3.1 Using Matlab to auto-generate Simulink models

Case study: Integrated vehicle chassis control architecture
3.2 Developing generic and re-useable subsystems

Case study: Dynamically configured down counter (ex4)
3.3 Implementing Matlab and C code in Simulink s-functions

- Bespoke Simulink subsystems can be created using scripted code such as Matlab, C or Fortran

- This can be useful when Simulink flow doesn't lend itself to performing particular coding applications

- This can also be useful for creating low level interface and driver functions in embedded systems (when using RTW - to be discussed!)

- See the CAN message converter example (ex5)

- For a C language example of Simulink s-functions, try The Mathworks’ ‘timestwo’ example available from:
4. Advanced Matlab scripting

4.1 Using handle graphics (ex6)

4.2 Generating executable applications with the Matlab Compiler

4.3 Case study: Acoustic analysis system research project
4.1 Using handle graphics

- Handle graphics allow development of interactive tools from within the Matlab environment.

- Figures and axes are objects that can be used to display user control features and computed data.

- Uicontrol objects allow the user to make selections and input data.

- Uiget and Uiput commands allow the user to save and recall Matlab data.

- Callbacks allow functions to be called and implemented in real-time when control buttons are activated.

- Data management can be incorporated to allow data rendering to a preferred format, eg a text file.
4.1 Using handle graphics

Example: Seminar questionnaire (ex6)
4.2 Generating executable applications in Matlab

- The Matlab Compiler allows Matlab scripts to be packaged as a standalone .exe application

  deploytool opens the Matlab Compiler

- This means that Matlab applications can be distributed to people who do not own Matlab software

- A Matlab runtime wrapper must be installed for the .exe file to function, this however is freely available
4.3 Case study: Acoustic analysis tool
5. Auto-code generation for rapid prototyping

5.1 Advantages of auto-code generation within a model based design project

5.2 Simulink Real-Time-Workshop auto-code generation (ex7)

5.3 Case study: Vehicle control system development project
5.1 Advantages of auto-code generation within a model based design project

- Obvious advantages of simulation to enable ‘right-first-time’ design
- A single engineer can work in design, coding, deployment and testing and move quickly between each process
- Less design documentation to keep up to date – the model IS the design flowchart
- Simulation models can be auto-coded (with Real Time Workshop) and deployed as hardware-in-the-loop plant models
- Control code can be auto-coded to C or embedded code for immediate deployment to an electronic control unit (ECU)
5.2 Simulink Real-Time-Workshop auto-code generation (ex7)
5.3 Case study: Vehicle control system development

Embedded C file for microprocessor target
6. Conclude and recap

Summary refresh:

1. Using Matlab for data analysis and rapid algorithm development

2. Model based design with Simulink

3. Scripting efficiency with Matlab/Simulink

4. Advanced Matlab scripting

5. Auto-code generation for rapid prototyping
All example code denoted by (ex*) can be downloaded from:

http://www.robtoulson.com

(follow link to ‘Tools and Downloads’)