MATLAB-Automatisierung von Dymola-Simulationen und Ergebnisauswertung

Holger Dittus

Modelica User Group BaWü, Stuttgart, 13.06.2013

DLR

Inhalt

- Motivation
- Dymola .mos-Skripte
- MATLAB .m-Skripte
- Kombination von MATLAB und Dymola-Skripten

Motivation
Warum Automatisierung?

Automatisierung ermöglicht

- Beschleunigung häufig wiederkehrender Vorgänge
- Schnelle Anpassung von Modellen und Modellparametern
- Nutzung einfacher grafischer Benutzeroberflächen
- Parametervariationen mit geringem Aufwand
- Erzeugung von Standard-Ausgabeformaten und –grafiken
- Reproduzierbarkeit

Automatisierungsvarianten

1. Direkte Automatisierung in Dymola: .mos-Skripte
2. Automatisierung in MATLAB: .m-Skripte mit Zugriff auf *dymosim.exe*
3. Kombination von .mos und .m-Skripten

Dymola-Simulation

Jeder Dymola-Simulationslauf erzeugt:

- *dymosim.exe*: ausführbare Datei des Simulationsmodells
- *dsin.txt* oder *dsin.mat*: Modell- und Löserparameter für *dymosim.exe*
- *dsres.mat*: Ergebnisdatei
- *dslog.txt*: Logfile mit Status- und Fehlermeldungen

Für die Automatisierung kann auf diese Dateien zurückgegriffen werden, z.B. um die im Modell verwendeten Parameter auszulesen.

Dymola .mos-Skripte
Anwendungsbeispiele

- Start-Skript zum automatischen Laden von Bibliotheken und einstellen des gewünschten Arbeitsverzeichnisses

```
// define working directory
mdcwork = "C:/HD/Dymola_Work";

// open libraries
openModelFile("VehicleInterfaces","c:/VehicleInterfaces 1.1/package.mo");
openModelFile("AlternativeVehiclesExt","c:/AlternativeVehiclesExt 1.0/package.mo");
openModelFile("RailwayVehicles","c:/RailwayVehicles 1.0/package.mo");

// go to working directory
cd(mdcwork);
```

Dymola .mos-Skripte
Anwendungsbeispiele

- Automatische Erzeugung von Grafiken
- Speichern in Standard-Formaten

```
// Plot erzeugen
createPlot(id = 4,
 position = {-7, -30, 1436, 362},
 y = Par11,
 range = {timeMin, timeMax, 50.0, 20.0},
 autoscale = auto,
 autoerase = true,
 autoreplot = true,
 description = false,
 grid = gridOn,
 color = true,
 online = false,
 legend = false,
 subPlot = 2,
 leftTitleType = 1,
 bottomTitleType = 1,
 colors = OneColor);
// Plot speichern als .png
ExportPlotAsImage(actFile+"_Plot4.png", 4);
```

Dymola .mos-Skripte
Verfügbare Funktionen

listfunctions() in Dymola-Befehlszeile zeigt die verfügbaren Funktionen:

```
listfunctions()
function fill "fill an array";
function size "size of array";
function array "construct array of arrays";
function ReadAnimationSocket;
function importFMU "imports an FMU";
function Dymola_AST_ClassShownInBrowser;
function BaseFontSize "Base font size";
function RestoreWindow "Restore window setup";
function switchToModelingMode "switch to modeling mode";
function switchToSimulationMode "switch to simulation mode";
function messageDialog "wait for user input";
function messageBox "show message box";
```

document(„NAME“) zeigt Kurzbeschreibung:

```
document("saveSettings")
function saveSettings "Stores current setting on a file"
  input String fileName "File to store in";
  input Boolean storePlot := false "Store plot commands";
  input Boolean storeAnimation := false "Store animation commands";
  input Boolean storeSettings := false "Store global flags";
  input Boolean storeVariables := false "Store current parameter setting";
  input Boolean storeInitial := true "Store variables at initial point";
  input Boolean storeAllVariables := true "Store all variables";
  input Boolean storeSimulator := true "Store simulator setup";
  output Boolean ok;
end saveSettings;
```

Dymola .mos-Skripte
Vor- und Nachteile

Vorteile:

- Direkt in Dymola ausführbar
- Modelle compilieren und starten möglich
- Modelländerungen möglich
- Modularer Aufbau möglich
- Teilweise automatisch erzeugt
- Gut geeignet für wiederkehrende Aufgaben
- Befehle intuitiv verständlich

Nachteile:

- Eingeschränkte Funktionalität
- Doku sehr kurz gehalten
- Gesuchte Funktionen schwer zu finden, Suchfunktion unzureichend
- Eigene Programmiersprache
- Datenexport nur in wenige Formate möglich
- Komplexe Grafiken nicht möglich
- Vergleich / Zusammenfassung mehrerer Simulationsläufe schwierig

MATLAB .m-Skripte
Anwendungsgebiete

- Erzeugung von Eingabedaten
- Von Dymola mitgelieferte .m-Skripte (*dymbrowse, dymload, dymget*)
- Postprocessing von Signalen, z.B. Einheitenumrechnung, Mittelwert, Maximum, Minimum, Curve Fitting, …
- Darstellung und Speichern von Ergebnissen mit umfangreichen Visualisierungsmöglichkeiten, z.B. Flächen- und Balkendiagramme, …
- Aufruf von weiteren Windows-Programmen und Datenübergabe
- Bedingte Operationen, komplexe Schleifen,
- Weitergabe an Dritte einfach möglich (sowohl Ergebnis als auch Auswerte-m-file)
- Datenexport auch in andere Dateiformate (z.B. xls)

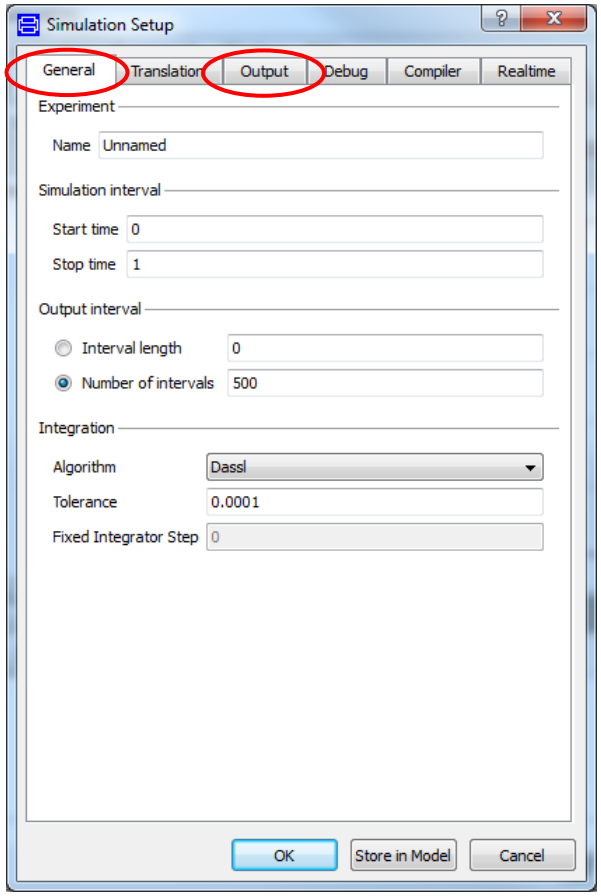
MATLAB .m-Skripte

SARA – Simulation Automation & Result Analysis

- generate_dymo_varlist.m
- plotmaster_template_period.m
- plotmaster_template_raw.m
- plotmaster_template_sum.m
- sim_00_init.m
- sim_10_simulation_dymola.m
- sim_20_parameter.m
- sim_30_raw.m
- sim_31_raw_calc.m
- sim_32_raw_plot.m
- sim_40_period.m
- sim_41_period_calc.m
- sim_42_period_plot.m
- sim_50_sum.m
- sim_51_sum_calc.m
- sim_52_sum_plot.m
- sim-log.txt

- Strukturierter Satz von MATLAB-Skripten gegliedert in:
  - Initialisierung
  - Aufruf der Simulation(en)
  - Auswertung der Roh-Ergebnisse
  - Aufbereitung von Grafiken

MATLAB .m-Skripte
fkdymosim.m - Übersicht



- zentrale Funktion zur Durchführung von Dymola-Simulationen
- basiert auf einer existierenden *dymosim.exe*
- erzeugt / speichert die Eingabeparameter der *dymosim.exe* in *dsin.mat*
- lädt *dsin.mat* in MATLAB-workspace
- überschreibt entsprechend user-Vorgaben:
  - Modell- und Löserparameter
  - Ausgabeeinstellungen
- Startet Simulation mit geänderten Parametern
- Wertet Simulationsstatus aus

MATLAB .m-Skripte
fkdymosim.m - Einschränkungen

- *dymosim.exe* muss manuell erzeugt werden
- nur in *dsin.mat* vorhandene Parameter können geändert werden
- array-Größen sind nicht veränderbar ➔ Probleme bei Daten, die dynamisch aus mat-Files o.ä. eingelesen werden
- replaceable Modells können nicht ausgetauscht werden

➔ Für einfache Parametervariationen geeignet, komplexe Modellmanipulationen erfordern aber direkten Zugriff auf die .mo-Dateien

MATLAB .m-Skripte
Vor- und Nachteile

Vorteile

- Umfangreiche Funktionalität, sehr gute Dokumentation, große Community
- Postprocessing von Signalen, Erstellung von Grafiken, Datenexport
- Aufruf von weiteren Windows-Programmen und Datenübergabe
- Erzeugung/Aufruf von .mos-Skripten möglich
- GUI-Anbindung an Dymola über *dymbrowse, dymload, dymget*

Nachteile

- Vorkenntnisse notwendig
- MATLAB ist Zusatzsoftware für die weitere Kosten entstehen
- Keine direkte Modellanpassung möglich
- Zugriff auf einzelne Simulations-Parameter manchmal nicht möglich

Komplexe Modellmanipulationen
Kombination von MATLAB und .mos-Skripten

1. In Dymola: Manuelle Erstellung eines Packages mit gewünschten Modellen
2. MATLAB erstellt automatisch eines Instanz des Packages über Windows-Befehle
3. Direkte Anpassung der .mo-Datei durch Text-Manipulation, z.B.
   - Austausch replaceable models
   - Ersetzen von Arrays, auch mit anderen Größen
   - …
4. Automatische Erzeugung eines .mos-Skripts aus MATLAB:
   - Dymola und Bibliothek(en) öffnen
   - Modell(e) compilieren
   - *dymosim.exe* erzeugen
5. Simulationen / Parametervariationen durchführen mit *fkdymosim.m*

Kontakt:
Holger Dittus
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Fahrzeugkonzepte
Pfaffenwaldring 38-40
70569 Stuttgart