Repeatability Evaluation Submission for HSCC 2018

Akshay Rajhans*, Srinath Avadhanula, Alongkrit Chutinan, Pieter J. Mosterman, Fu Zhang

January 12, 2018

Abstract

This article outlines the Repeatability Evaluation (RE) package accompanying the tool paper "Graphical Modeling of Hybrid Dynamics with Simulink and Stateflow" [RAC+] accepted at the 2018 HSCC conference.

1 Getting access to release R2017b of MATLAB

The paper requires using the R2017b release of MATLAB and requires access to Simulink and Stateflow products. In case any member of the repeatability evaluation committee does not have access to MATLAB R2017b via their institution, a free 30-day trial of MATLAB can be obtained[1] via this URL: `https://www.mathworks.com/programs/trials/trial_request.html?prodcode=SF`.

2 What should be evaluated for reproducibility?

Technically, the only two computational artifacts from the paper that need to be evaluated for repeatability are the plots in Figures 4 and 7. As part of this RE package, we provide instructions not just for reproducing these two Figures, but also for opening model views of which the other Figures are screenshots, as well as for reproducing the Simulink-to-Stateflow copy-paste workflow and library-link reuse workflow mentioned in the context of the pole vault example and powertrain controller example respectively.

3 Computational contents of the RE package

The .zip archive contains the following computational content.

3.1 Models

1. Pole vault models `mPoleVaulter.slx` and `mPoleVaulter_SimulinkOnly.slx`
2. Powertrain controller model `mPowertrainAFController.slx` and the library for open loop feed-forward control `lib_PowertrainFFDynamics.slx` that enables reuse
3. Cardiac cell model `mCardiacCell.slx`

*Contact: Akshay.Rajhans@mathworks.com

[1] Please note that creating a MathWorks account or signing in with an existing MathWorks account is necessary in order to receive the trial. If the RE evaluation committee runs into any trouble obtaining access, please contact the first author.

3.2 Figures

These are provided as a baseline for reproducibility.

1. Clutch plot Figure file `clutchPlot.fig` (Figure 4 in [RAC+])
2. Pole vaulter plot Figure file `poleValuterPlot.fig` (Figure 7 in [RAC+])

The RE committee reviewers should be able to reproduce similar plots on their own and compare against these. Instructions for reproduction are provided in this document as well as in the code scripts.

3.3 Code

For the convenience of the RE committee, instead of having to copy-paste code from this PDF file, we provide MATLAB scripts that they can run. The code is organized in sections corresponding to each model, and running each section will generate the appropriate Figures and/or open model views of which the Figures are screenshots. Each script is complete, so running either script should be sufficient.

1. Script `GHA_HSCC2018_RE_script.m`
2. Live script `GHA_HSCC2018_RE_live_script.mlx`

4 Clutch model

The running example of a mode-switching clutch model from the paper is available to all users as part of Stateflow. This model is not re-distributed as part of the RE package, but rather can be accessed simply by typing `sf_clutch` at the MATLAB command line.

4.1 Figure 1

Figure 1 shows the graphical hybrid automaton of the Clutch example. The view shown in Figure 1 can be accessed by running the following code[2]. (Note that the Simulink Functions `detectLockup` and `detectSlip` are cropped out of the screenshot in Figure 1 in the paper because of space constraints.)

```
>> open_system sf_clutch
>> open_system sf_clutch/Clutch
```

4.2 Figure 2

Figure 2 shows the Simulink state `Slipping`.

```
>> open_system sf_clutch/Clutch/Slipping
```

Note that if this or any subsystem opens such that the contents do not fit on your screen, pressing space bar is a quick way to resize the contents to automatically fit the screen.

4.3 Figure 3

Figure 3 shows the Simulink state `Locked`.

```
>> open_system sf_clutch/Clutch/Locked
```

[2] Throughout this document, '>>' is intended to stylistically indicate the MATLAB command prompt, and should not be interpreted as being part of the command to be entered.

4.4 Figure 4

Figure 4 can be reproduced by following the steps below.

1. Simulate the model `sf_clutch` and double-click on the Scope block titled '`Speeds`' at the topmost (root) level of the model, or run the following code.

   ```
   >> sim sf_clutch
   >> open_system sf_clutch/Speeds
   ```

   This should reproduce the correct values for the plot.

2. Export this Simulink plot to a MATLAB Figure (File → Print to Figure)

3. Display the Legend
   (a) Show Legend (Insert → Legend)
   (b) Double click on the text in the Legend box to set it to '`Engine Speed`' and '`Vehicle Speed`' respectively.

4. Edit Figure properties (Edit → Figure Properties)
   (a) Click on the background, then select fill color as White, and line color as Black
   (b) Click on the `Engine Speed` plot, select line color as Blue, line width as 3
   (c) Click on the `Vehicle Speed` plot, select line color as Orange, line width as 3.
   (d) Click on the Legend, select Fill Color as White, Font Color as Black.

4.5 Figure 5

Figure 5 shows the contents of the Simulink Function `detectLockup`.

```
>> open_system sf_clutch/Clutch/detectLockup
```

5 Pole vault example

5.1 Figure 6

Figure 6 shows the GHA of the pole vault model

```
>> open_system mPoleVaulter
>> open_system mPoleVaulter/PoleVaulter
```

5.2 Figure 7

1. The data for the plot is generated by simulating the model.

   ```
   >> sim mPoleVaulter
   ```

2. The formatting is achieved by running the corresponding code provided in the MATLAB scripts ('Step 2' under 'Reproducing Figure 7').

5.3 Simulink to Stateflow Copy-paste workflow

In the context of the pole vault example, we introduce the Simulink-to-Stateflow copy-paste workflow where copying an existing Action subsystem from a Simulink canvas when pasted onto a Stateflow canvas automatically creates a corresponding Simulink state. To reproduce this workflow, follow these steps

1. Open the source Simulink-only model `mPoleVaulter_SimulinkOnly.slx`[3]

   ```
   >> open_system mPoleVaulter_SimulinkOnly
   ```

2. Open the destination Stateflow chart

   ```
   >> open_system mPoleVaulter
   >> open_system mPoleVaulter/PoleVaulter
   ```

3. Copy any of the Action subsystems (the `Run-up`, `Take-off` and the `Fly` subsystems) from the source Simulink canvas

4. Paste onto the destination Stateflow canvas to see a corresponding Simulink state created

6 Powertrain Controller example

6.1 Figure 8

Figure 8 shows the GHA of the controller

```
>> open_system mPowertrainAFController
>> open_system mPowertrainAFController/Chart
```

6.2 Figure 9

Figure 9 shows the library that enables reuse of the open-loop feedforward dynamics

```
>> open_system lib_PowertrainFFDynamics
```

6.3 Library link workflow

In the context of the powertrain control example, we introduce the library-link-based reuse workflow where, instead of creating duplicate Simulink states with same dynamics, one can create links to a library subsystem. To reproduce this workflow, follow these steps

1. Open the source library subsystem `lib_PowertrainFFDynamics.slx`

   ```
   >> open_system lib_PowertrainFFDynamics
   ```

2. Open the destination Stateflow chart

   ```
   >> open_system mPowertrainAFController
   >> open_system mPowertrainAFController/Chart
   ```

3. Copy the Action subsystems `OpenLoopControlLaw` from the source Simulink library canvas

4. Paste onto the destination Stateflow canvas to see a corresponding Simulink state created with a library link[4].

[3] This model is taken directly from [Rou16] and can also be downloaded from the blog post.

[4] To confirm that this is a library link, ensure that double-click on this new Simulink state opens up a Simulink canvas with a lock icon at the bottom left corner. Hovering on this icon will show a tooltip that says '`Locked: library link`'.

7 Cardiac cell example

7.1 Figure 10

Figure 10 shows the GHA of the cardiac cell model

```
>> open_system mCardiacCell
>> open_system mCardiacCell/Chart
```

References

[RAC+] A. Rajhans, S. Avadhanula, A. Chutinan, P. J. Mosterman, and F. Zhang. Graphical modeling of hybrid dynamics with simulink and stateflow. In *HSCC 2018*. Accepted.

[Rou16] Guy Rouleau. Olympic 2016 – pole vault. *Guy on Simulink*, (https://blogs.mathworks.com/simulink/2016/08/19/olympic-2016-pole-vault), 2016.