Comparison of model simulations

Anton de Villiers*

April 18, 2016

1 Introduction

This document contains the most recent results and findings of five distinct complex models programmed in OpenModelica [5]. The models are built to simulate physical complex systems (*e.g.* systems containing mechanical, electronic, or process-oriented subcomponents) and are described by means of a combination of differential, algebraic and discrete equations where the number of unknowns and the number of equations is identical.

OpenModelica's "gold-standard" solver for numerically solving ordinary differential equations (ODEs) is DASSL [6]. The DASSL solver can simulate all of the models in this document accurately within the OpenModelica framework. In addition to the DASSL solver, numerous other solvers are provided by OpenModelica including Runge-Kutta and Euler based solvers. Finally, OpenModelica also provides a Quantised State Systems (QSS) [3] method, which is still in development. The QSS solvers differs fundamentally from traditional ODE solvers which is based are on the notion of state quantization, ignoring the traditional approach of time discretization when numerically solving ODEs. OpenModelica's version of a QSS solver currently has very limited functionality.

QSS Solver[1] [7] is a modeling and simulation environment for continuous and hybrid systems. This simulation tool describe models using a subset of the Modelica language called MicroModelica [2]. QSS Solver supports the entire family of QSS related solvers, including a family of Linearly Implicit QSS (LIQSS) [3] solvers which were created to deal with stiff mathematical systems. The LIQSS solvers may be considered to be more comprehensive solvers than that of the normal QSS methods. In order to convert Modelica models to MicroModelica models, ModelicaCC [1] is used which consists of four conversion stages. ModelicaCC has four steps to convert models from Modelica syntax to MicroModelica syntax. These steps are:

- flatter
- antialias
- mmo
- causalize

HealthQ has undertaken to incorporate a QSS solver as well as a first order LIQSS solver in OpenModelica. These solvers aim to build on the current implementation details of the current QSS solver in OpenModelica.

*HealthQ Technologies, Office 9, First Floor, The Woodmill Lifestyle, Vredenburg Road, Devon Valley, Stellenbosch, 7600, South Africa

[1]QSS Solver is an application that contains a family of QSS implemented solvers for addressing ODE systems.

2 The five models

Numerical results of five distinct models will be considered in this document. These five models are:

- A stiff mathematical ODE system (Stiff)
- A pulse model of the tactile arterial palpation of the heartbeat (Pulse)
- An ideal diode system in a circuit (Ideal)
- A ball bouncing down stairs (BBall)
- A simple cardiovascular system (CardioV)

3 The solvers

All the solvers used have been mentioned in §1 of this document. These solvers include:

- OpenModelica DASSL (OM_DASSL)
- OpenModelica QSS (OM_QSS)
- QSS Solver's LIQSS solvers. This includes the first-, second- and third-order LIQSS solvers (QSS_LI1, QSS_LI2 & QSS_LI3)
- HealthQ's implementation of QSS and LIQSS in OpenModelica (HQ_QSS & HQ_LI)

4 Simulation details

For each simulation run, the following will be specified:

- duration of the simulation,
- some key state variables that will be investigated,
- the actual completion time of the simulation (including the compilation times, back-end calculations *etc.*), and
- a conclusion of the obtained results.

For the DASSL solver in OpenModelica a tolerance of `1e-6` is used and the number of intervals will be fixed at 1 000 for all simulations.

In QSS Solver and absolute tolerance of `1e-6` and relative tolerance of `1e-3` is used.

HealthQ's QSS and LIQSS solver in OpenModelica both have $\Delta Q_i = 0.001$ for each state variable i. This is constant for all models, except the Cardiovascular Circulation model and the Bouncing Ball models where $\Delta Q_i = 0.01$. For the Cardiovascular Circulation model the ΔQ_i is furthermore multiplied by a "nominal" value[2] of each state variable.

[2]This "nominal" value is calculated as an approximate average value of the state variable. To retrieve this average value a simulation is run using OpenModelica's DASSL solver. The numerical results are then aggregated for each state variable individually. This average value is then multiplied by the tolerance to provide and adequate step size for the LIQSS solver between any two time steps — this is done for each state variable.

5 Results

The numerical results and related finding will be considered in this section. The computational times reported are in seconds and are the sum of the `user` time and `sys` time provided by the UNIX `time` command, which is used in conjunction with the executable that is built for each model. The `user` time is defined as the *total number of CPU-seconds that the process used directly (in user mode) in seconds*, while the `sys` time is the *total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds*. The total time `user+sys` indicates the total amount of actual CPU time a simulation uses.

5.1 A stiff mathematical system

The first model is a simple ODE system that is mathematically stiff. The model is:

$$\begin{aligned} \dot{x}_1(t) &= 0.01x_2(t) \\ \dot{x}_2(t) &= -100x_1(t) - 100x_2(t) + 2020 \end{aligned}$$

with initial solutions $x_1(0) = 0$ and $x_2(0) = 20$.

This first model was specifically chosen to highlight the consistency between solvers (and their applications) when a model is simulated correctly. The results are shown in Table 5.1. The simulation was run for 500 seconds in total. In this case both the state variables $x_1(t)$ and $x_2(t)$ are considered in particular. In this case all the solvers were able to solve the model and the numerical results were very much similar, except for the QSS solvers (not the linearly implicit versions). Figure 5.1 displays some graphical results of this model.

Solver	OM_DASSL	OM_QSS	QSS_LI1	QSS_LI2	QSS_LI3	HQ_QSS	HQ_LI
Sim Time	0.009	2.521	0.256	0.027	0.008	0.109	0.014

Table 5.1: Simulation times of Stiff.

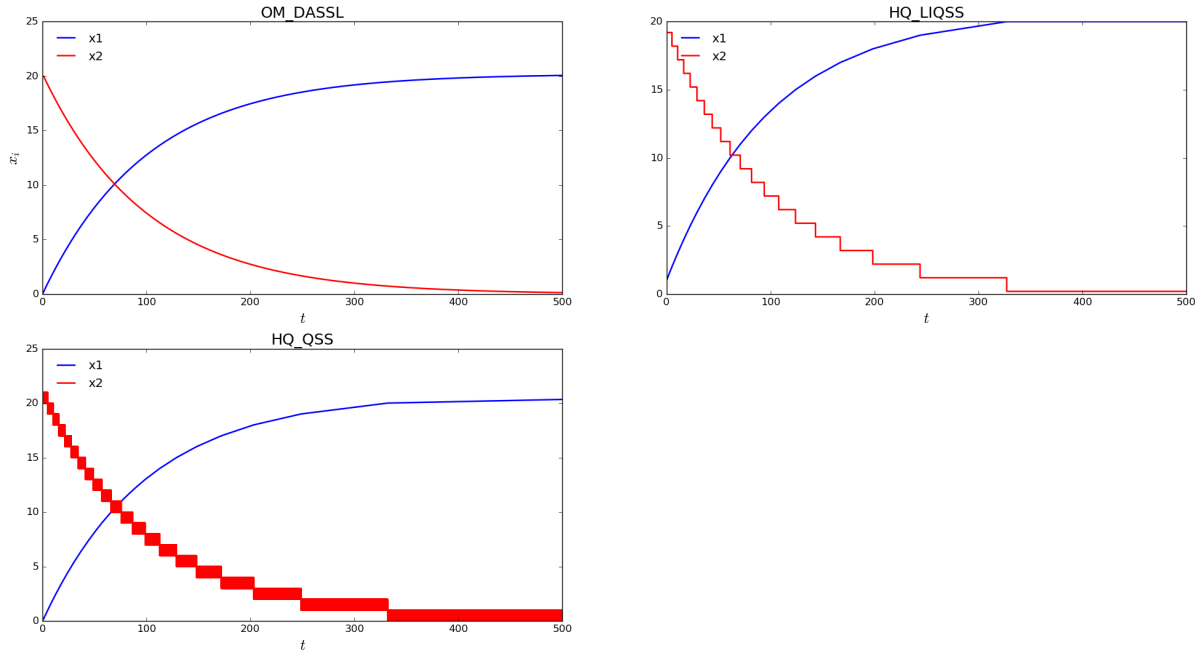

Figure 5.1: Simulation results of Stiff for the first 500 seconds of the simulation runs for (a) OM_DASSL, (b) HQ_LIQSS and (c) HQ_QSS with large quantums in (b) and (c). These artificially large quantums in HQ_LIQSS and HQ_QSS were not used in the actual simulations — it is merely for illustration purposes.

5.2 Pulse

The Pulse model simulates a pulse or heartbeat of a human. This model contains an event which changes by means of sinus oscillation – that is, the pulse is quicker when the sinus value is larger than zero and the oscillations are slower when the sinus value is less than zero. Simulation were performed for 100 seconds. It was a difficult task to convert the Pulse model to simulate successfully in QSS Solver, but it was eventually possible by creating a piecewise function in QSS Solver. The model therefore had to be adjusted for the QSS Solver application. In this case the `phi` variable is of importance and will be considered in the results.

Table 5.2 contains the simulation times of this model. Two solvers encountered problems with this model, those being OM_QSS and QSS_LI1. The remainder of the solvers had no problems simulating the model. Figure 5.2 shows how the OM_DASSL and QSS_LI3 solvers addressed the model and provides similar solutions.

Solver	Sim time	Comments
OM_DASSL	0.058	Correct simulation results obtained
OM_QSS	8.950*	Only 1 second simulation time
QSS_LI1	N/A	Unable to solve model. QSS Solver crashes
QSS_LI2	0.001	Correct simulation results obtained
QSS_LI3	0.070	Correct simulation results obtained
HQ_QSS	0.189	Correct simulation results obtained
HQ_LI	0.675	Correct simulation results obtained

Table 5.2: Simulation results of Pulse.

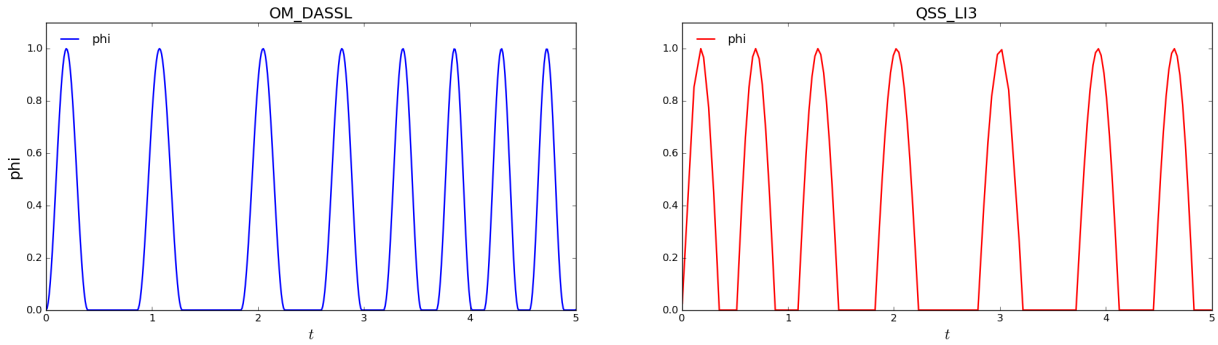

Figure 5.2: Simulation results of Pulse for the first 5 seconds of the simulation runs for (a) OM_DASSL and (b) QSS_LI3.

5.3 Ideal diode

The ideal diode or perfect diode is a two terminal device, which completely allows the electric current without any loss under forward bias and completely blocks the electric current with infinite loss under reverse bias. An ideal diode is a good example as it contains some conditional equations for when the flow is on or off.

Solver	Sim time	Comments
OM_DASSL	0.205	Correct simulation results obtained
OM_QSS	116.006	Can only simulate 321 seconds in total.
QSS_LI1	None	No feasible conversion
QSS_LI2	None	No feasible conversion
QSS_LI3	None	No feasible conversion
HQ_QSS	None	Segmentation fault thrown
HQ_LI	34.889	Correct simulation results obtained

Table 5.3: Simulation results of Ideal.

This model cannot be reduced to MicroModelica by means of ModelicaCC as the causalize step suggests that the number of variables and equations do not match. Therefore simulation of this model is not possible in QSS Solver.

The simulation time is set at 1 000 seconds. The `resistance.q` variable is considered in this model as a key variable. Table 5.3 contains a summary of simulation run results, while Figure 5.3 displays the OM_DASSL and HQ_LI solver results, which were the only two solvers that could solve the model successfully for 1 000 seconds.

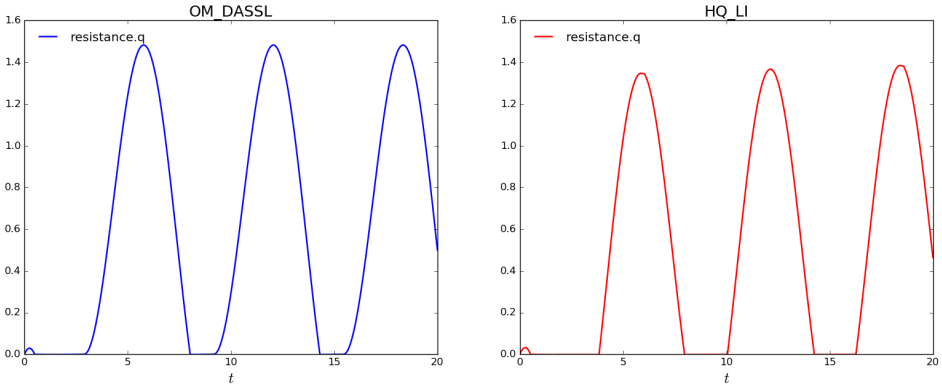

Figure 5.3: Simulation results of Ideal for the first 20 seconds of the simulation runs for (a) OM_DASSL and (b) OM_LI.

5.4 Bouncing ball

The bouncing ball is an example of a hybrid system, containing events and conditional expressions. We considered two variations on this model, one where the ball is released in a forward direction and bounces on a flat surface. The alternative model contains stairs and the ball bounces down these stairs.

5.4.1 Flat surface bouncing

The model was simulated for 25 seconds in each run. Only the OM_DASSL solver was able to successfully solve this model as expected. Table 5.4 contains a summary of the simulation runs, while Figure 5.4 contains the graphical results for each simulation run. From these results it is clear that the solvers contain fundamental differences, especially with respect to this model.

Solver	Sim time	Comments
OM_DASSL	0.025	Correct simulation results obtained
OM_QSS	141.846	No events are triggered
QSS_LI1	0.259	Ball continually bounces higher
QSS_LI2	0.105	Erratic bouncing
QSS_LI3	0.011	Erratic bouncing
HQ_QSS	4.216	The ball bounces to the same approximate height
HQ_LI	7.245	The ball does not bounce high enough

Table 5.4: Simulation results of BBall on a flat surface.

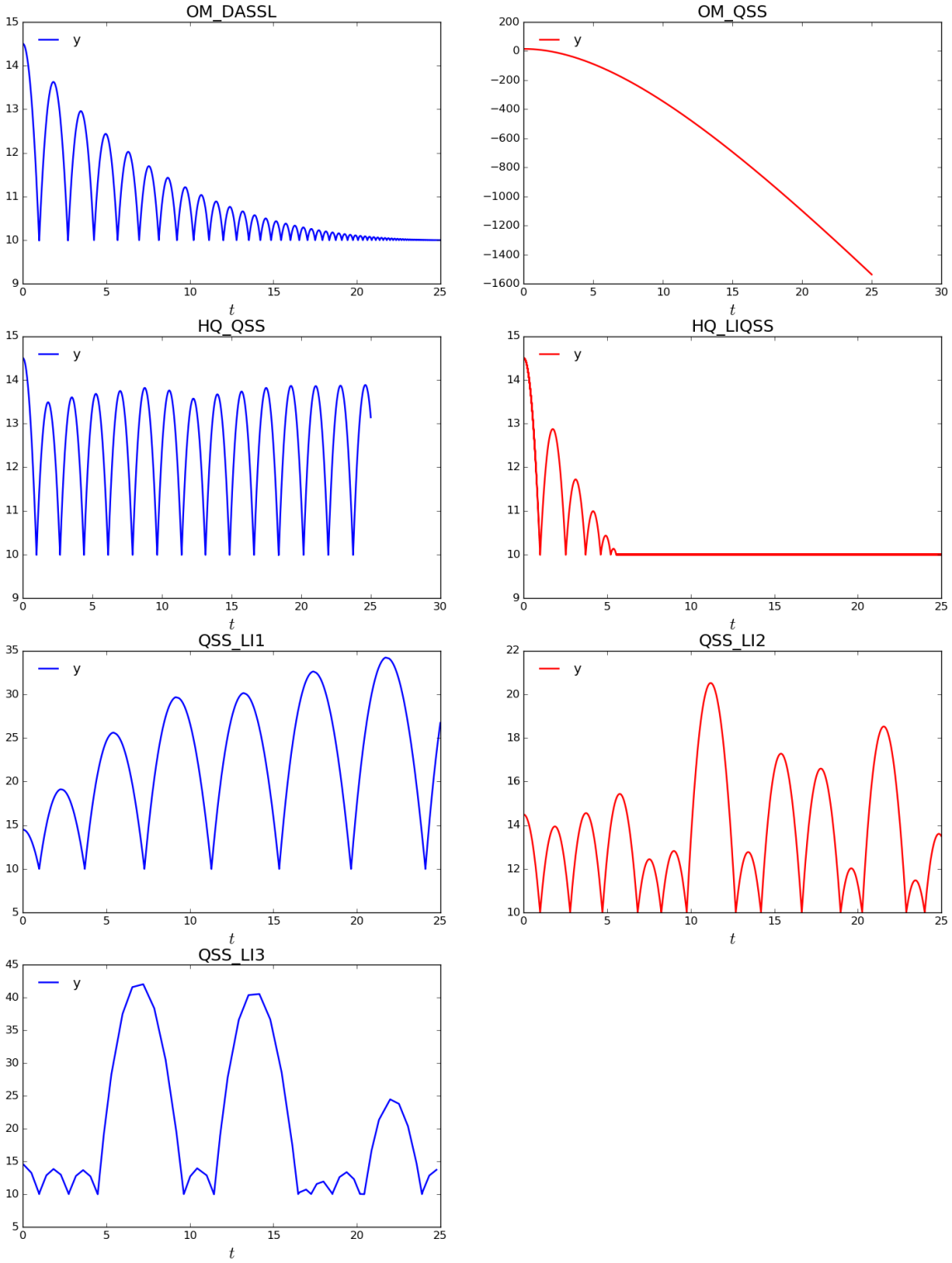

Figure 5.4: Simulation results of BBall on a flat surface for (a) OM_DASSL, (b) OM_QSS, (c) HQ_QSS, (d) HQ_LIQSS, (e) QSS_LI1, (f) QSS_LI2 and (g) QSS_LI3. Only the OM_DASSL solver provides the expected and correct simulation results.

5.4.2 Descending stairs

This model is further complicated by the addition of adding stairs. The simulation is run for 40 seconds and the important state variables considered are `y` and `stair`.

This model is particularly difficult to simulate. The only model that was able to simulate this model correct was OM_DASSL. All the other solvers had some form of inaccuracies or faults that renders the simulation results as not useful. Table 5.5 contains a summary of the simulation runs, while Figure 5.5 contains the graphical results

Solver	Sim time	Comments
OM_DASSL	0.017	Correct simulation results obtained.
OM_QSS	190.237	No events are triggered
QSS_LI1	None	QSS Solver crashes
QSS_LI2	0.156	Erratic bouncing
QSS_LI3	0.015	Erratic bouncing
HQ_QSS	5.015	Ball sometimes bounces too high
HQ_LI	9.514	The ball does not bounce high enough

Table 5.5: Simulation results of BBall.

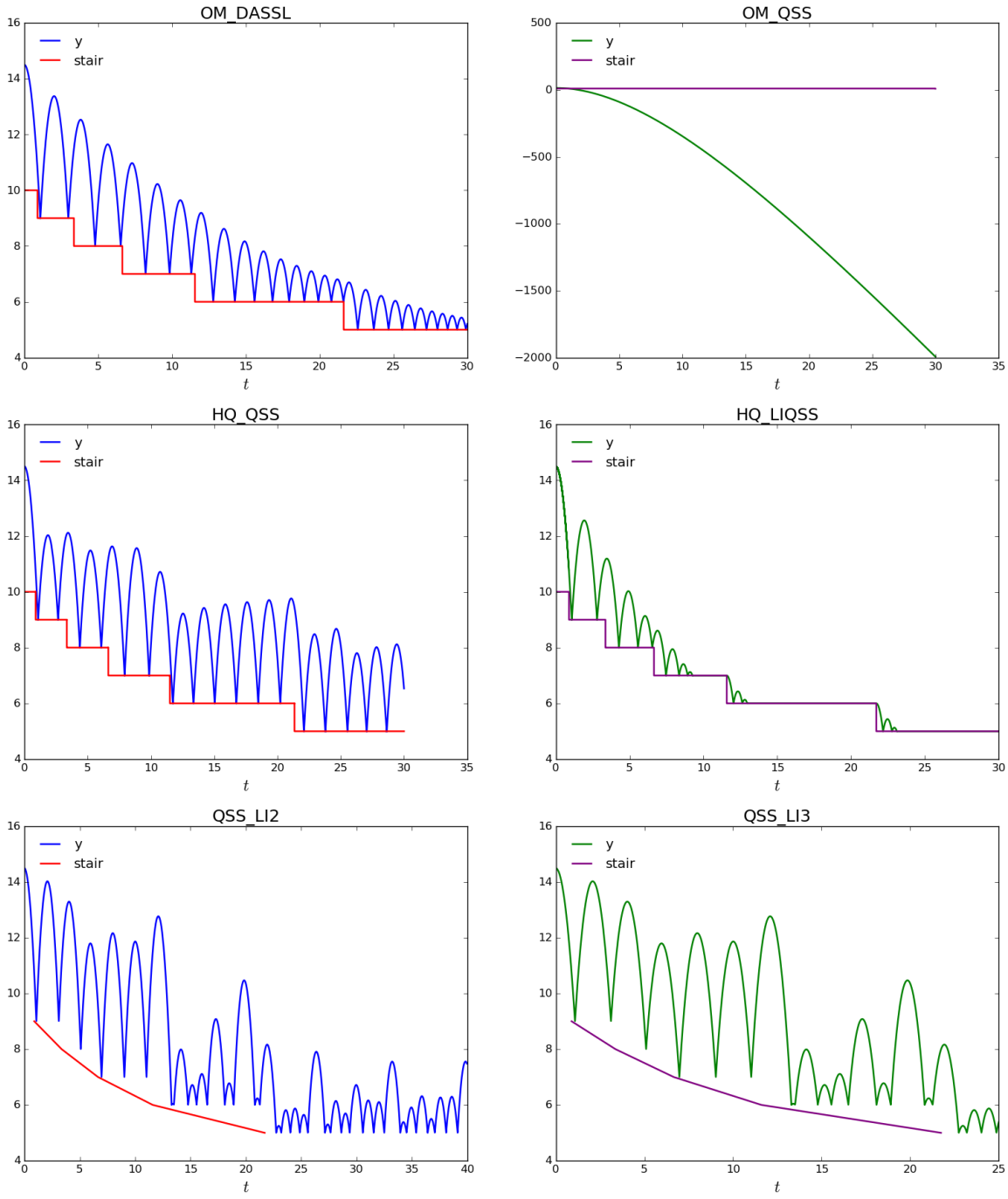

Figure 5.5: Simulation results of BBall down a set of stairs for (a) OM_DASSL, (b) OM_QSS, (c) HQ_QSS, (d) HQ_LIQSS, (e) QSS_LI2 and (f) QSS_LI3. Only the OM_DASSL solver provides the expected and correct simulation results.

5.5 Cardiovascular Circulation

The cardiovascular circulation model is a simple system of the cardiovascular system of the human body. This model entails elements from all the other models presented in this document. The simulation time is set at 100 seconds, but only OM_DASSL could solve the model successfully for this time period. The HQ_LI solver can solve the model for shorter periods of time such as 10 seconds. In this simulation the two state variables that were investigated are `heart.leftHeart.ventricle.compartment.P` and `heart.leftHeart.characteristicResistor.flow_out.P`.

Table 5.6 contains of summary of the simulation runs, while Figure 5.6 displays the graphical results of OM_DASSL and HQ_LIQSS for the first three seconds of a simulation run.

Solver	Sim time	Comments
OM_DASSL	2.503	Solves the model ideally.
OM_QSS	None	Unable to solve the model for more than 0.01 seconds.
QSS_LI1	None	Unable to simulate in the MicroModelica framework.
QSS_LI2	None	Unable to simulate in the MicroModelica framework.
QSS_LI3	None	Unable to simulate in the MicroModelica framework.
HQ_QSS	None	Unable to simulate the model for more than a second. The model is too stiff and the QSS produces very short increments in time steps.
HQ_LI	117.846*	Slow simulation speed. Cannot simulate the full 100 seconds as too many time steps are created and the memory becomes overloaded. We successfully simulated 10 seconds.

Table 5.6: Simulation results of CardioV.

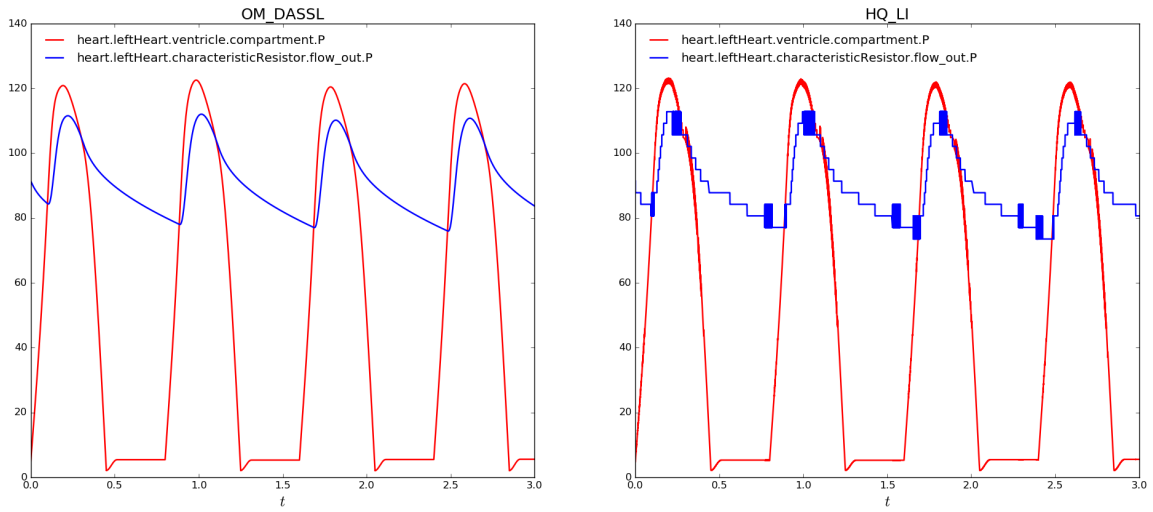

Figure 5.6: Simulation results of CardioV for the first 3 seconds of the simulation runs for (a) OM_DASSL and (b) HQ_LI.

6 Conclusion

This document illustrates some of the fundamental differences in simulation results for complex systems using different modeling applications and various solvers within these applications. OpenModelica's DASSL solver is very robust and provides numerically stable results for all the models considered in this document. OpenModelica's QSS solver is computationally very expensive and algorithmically struggles to handle and support event detection and updates. QSS Solver's LIQSS solvers have the ability to solve some models in very short periods of time. The shortcomings with QSS Solver's methods include inaccurate simulation results, incorrect event handling and the fact that the MicroModelica framework must be used, instead of Modelica's environment. HealthQ's QSS solver is outperformed by its LIQSS solver which has concerns related to event handling and slower simulation times but is able to simulate larger, complex models.

The family of QSS Solvers are fundamentally different from traditional solvers (*e.g.* DASSL, Runge-Kutta and Euler) as they do not discretize the total simulation times. The times at which the ODEs are updated is not known in advance and not all ODEs are necessarily required to be updated at the same time. OpenModelica's framework is created with the aim of discretizing time steps and updating all ODEs simultaneously. This causes considerable inefficiencies with the current QSS methods implemented in OpenModelica. QSS Solver overcomes these obstructions, but the application is still in development and more complex models are often difficult or not possible to convert to the MicroModelica framework. There is thus a need to create an ideal framework to enable the full potential of the QSS solvers.

References

[1] Bergero F, Botta M, Campostrini E & Kofman E, [Online], Cited 5th April 2016, Available from `http://www.ep.liu.se/ecp/118/048/ecp15118449.pdf`

[2] Fernández J & Kofman E, *μ-Modelica Language Specification*, [Online], Cited 5th April 2016, Available from `http://www.fceia.unr.edu.ar/control/modelica/micromodelicaspec.pdf`

[3] Migoni G & Fofman E, 2009, *Linearly implicit discrete event methods for stiff ODE's*, Latin American Applied Research, **39(3)**, pp. 245–254.

[4] OMCompiler, 2016, *HealthQ OMCompiler submodule repository*, [Online], Cited 15th March 2016, Available from `https://bitbucket.org/antonpdv/omcompiler`

[5] OpenModelica, 2016, *Open Source Modelica Consortium*, [Online], Cited 15th March 2016, Available from `https://openmodelica.org/`

[6] Petzold LR, 1982, *A description of DASSL: A differential/algebraic system solver*, Technical Report, Applied Mathematics Division, Sandia National Laboratories, Livermore (CA).

[7] QSS Solver, 2016, *Modeling and simulation tool for continuous and hybrid systems*, [Online], Cited 15th March 2016, Available from `https://sourceforge.net/projects/qssengine/`