

OpenModelica Solvers for VHM

27 January 2016
Anton de Villiers

In an attempt to reduce the execution time during simulations of the VHM it was proposed that a novel solver be implemented in order to significantly reduce the computational execution times of these simulation runs. Quantised State Systems (QSS) methods are a family of numerical integration solvers based on the idea of state quantization, leaving the time variable continuous. QSS are numerically stable and provide good accuracy control.

OpenModelica currently has a basic QSS solver able to simulate simple ODE systems. This solver is currently still under development and is unable to perform any sort of event handling – any events are essentially ignored. Also no documentation is available on the implementation of this and other solvers. The OpenModelica forums also do not contain programmatic details to aid developers and contributors with in-depth queries.

To get a better understanding of the various QSS solvers, a basic QSS solver and a Linearly Explicit QSS (LIQSS) solver were implemented in Python as a proof of concept that the algorithms can address stiff ODE systems.

Most of the work in the literature done on QSS solvers involve the authors Ernesto Kofman and Federico Bergero. These authors have developed their own application to simulate ODE systems with a variety of QSS solver. The application is called QSS Solver. In order to utilise these stand-alone solvers, regular Modelica models must be translated into a simpler language called Micro-Modelica. To facilitate the transformation of an OpenModelica model to a Micro-Modelica model, Modelica C Compiler (ModelicaCC) is used as a four-step process to convert the Modelica model to a “simpler” equivalent one, eventually completely described within the Micro-Modelica framework and thereby enabling the use of QSS Solver to simulate the ODE system.

The four step process of ModelicaCC can be described as follows:

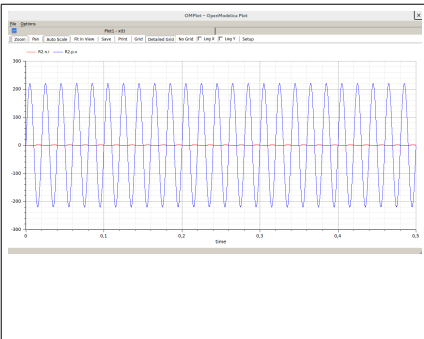
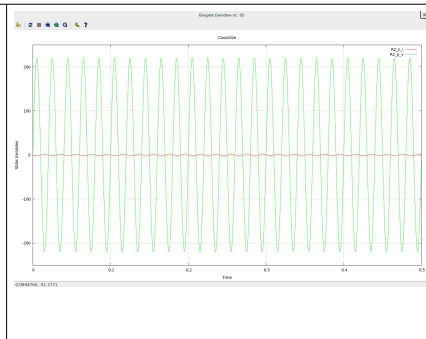
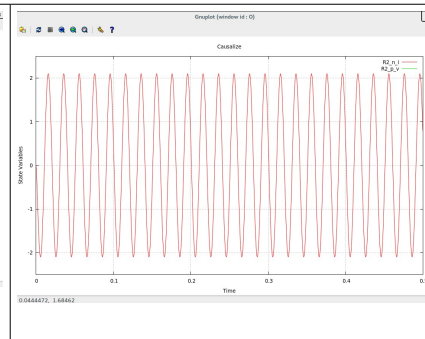
1. Flattening stage. This stage of the flattening algorithm deals with class flattening leaving connections (and connectors) untouched. The algorithm follows the principles of most Modelica compilers, but with a special treatment for arrays of variables. Thereafter the connectors are replaced throughout the model by means of a unique vectorised connection graph, which is briefly explained.
2. Alias elimination. Trivial equations are removed, i.e. $a[i] = b[i]$.
3. Reduction. This implies the reduction to Micro-Modelica syntax. Semantically equivalent expressions and operators are used in this here.
4. Causalization. Certain sets of algebraic for-loop are unrolled and expanded, while other arrays and for-loops are preserved. The preserved set of arrays which are not unrolled are represented in a bipartite graph structure, where each for loop can be modelled as a single node in a vectorised graph representation. Tarjan's algorithm is appended to utilise this new causalization approach.

After implementing these four steps, one should be able to simulate a model in QSS Solver using any of the available QSS solvers. The QSS Solver application contains a number of demo models, which simulate correctly. However, since this software is still under development, we have not been able to model the Handbook VHM Model (Danielsen Ottesen Cardiovascular model) successfully in QSS Solver. In fact we have not been able to pass the reduction step of the ModelicaCC process.

In an attempt to test ModelicaCC, we created two electrical circuits in OpenModelica – the one being the simple electrical circuit found in the OpenModelica textbook and another smaller model containing only a AC source (battery), a resistor and a ground. OpenModelica can successfully simulate these models. None of these two models can pass the flattening stage of ModelicaCC. To resolve this issue, we discovered that one can use OpenModelica's `omc` function to flatten the model. The resulting (flattened) model can then pass all the ModelicaCC steps. Contrary to what was expected, QSS Solver was able to simulate the larger electrical system, but could not simulate the smaller electrical system. It is currently unclear where any of the problems lie that we are experiencing. Without delving into the code (and understanding the fundamental implementation of ModelicaCC), ascertaining the origin of the problems we are experiencing will be difficult to find.

We have also been in contact with Federico Bergero around the conversion of our Handbook model in ModelicaCC as well as the results of the two electrical circuit models described above. Communication from

Federico has been brief as he is currently on vacation. He has, however, indicated that he believes the reduction stage of ModelicaCC may contain the bug, since we cannot successfully pass that particular stage when reducing the Handbook model. No meaningful feedback has yet been received on the strange occurrences experienced when attempting to model the electrical circuits. In the simple electrical circuit model it was found that the QSS solvers in the QSS Solver application was not able provide the correct numerical results.

		
OpenModelica's DASSL solver provides correct numerical results.	QSS Solver's DASSL solver provides the same results.	QSS Solver's QSS4 solver provides different numerical results.

Federico has also indicated that they are working on an FMI extension to allow QSS simulation. I have enquired on the status of this implementation, but have not received any feedback to date. He also indicated that the OpenModelica developers are attempting to bind their tool with the QSS solvers implemented in the QSS Solver application, but this is still very experimental. Similarly, an OpenModelica developer mentioned that they are collaborating with Federico to incorporate the solvers in OpenModelica, but that resources are currently limited.

In the meantime we have decided to investigate the current solvers in OpenModelica and attempt to create our own solver. An initial investigation into the current QSS solver implemented in OpenModelica was done. In particular, all the data that is available for used was examined. Changes to the OpenModelica implementation of the QSS solver was also made and the code of the solver then compiled. A test model was simulated to evaluate the effect the changes made had on the solver and its execution. We have a relatively good idea of what data is available to the solver and the algorithmic implementation of the solver is clear at this point.

The current investigation is situated around understanding the various kinds of events that may occur in OpenModelica and how OpenModelica handles these events. Furthermore, the manner in which information around events are handled by the solvers are still unclear. We will have to add the event-handling functionality to our novel QSS solver.

The idea is currently aimed at implementing a simple QSS solver with event handling and apply it to the Handbook VHM model.