Efficient Simulation of Hybrid Renewable Energy Systems

G. Migoni[a,b], P. Rullo[a], F. Bergero[a], E. Kofman[a,b]

[a]*French- Argentine International Center for Information and Systems Sciences (CIFASIS–CONICET), 27 de Febrero 210 bis, S2000EZP, Rosario, Argentine. Phone: +54-341-4237248-304. Fax: +54-341-482-1772*

[b]*Facultad de Ciencias Exactas, Ingeniera y Agrimensura - Universidad Nacional de Rosario, Argentina*

Abstract

This article explores the usage of novel tools for realistic modeling and efficient simulation of Hybrid Renewable Energy Systems (HRES). Using the object oriented Modelica language, a new library providing component models such as photovoltaic (PV) cells, proton exchange membrane (PEM) fuel cells, electrolyzers, hydrogen storage tanks, batteries and electronic converters is developed and used to build different HRES models. Since the components are represented under realistic assumptions, the resulting models exhibit frequent discontinuities, strong non-linearities and combinations of slow and fast dynamics (i.e. stiffness). As these features impose severe limitations to classic numerical simulation solvers, we analyze the use of a new family of numerical algorithms called Quantized State Systems (QSS) that overcome most of those difficulties. The results obtained show that these algorithms applied to realistic HRES are more than one order of magnitude faster than the most efficient classic solvers, allowing to simulate these systems in reasonable times.

Keywords: Hybrid renewable energy systems, Modelica, Quantized State System Simulation

Email addresses: `migoni@cifasis-conicet.gov.ar` (G. Migoni), `rullo@cifasis-conicet.gov.ar` (P. Rullo), `bergero@cifasis-conicet.gov.ar` (F. Bergero), `kofman@cifasis-conicet.gov.ar` (E. Kofman)

1. Introduction

Standard HRES consist of arrays of photovoltaic panels and/or wind generators powering AC, DC or mixed loads [1]. Since energy supplied by renewable sources depend mainly on environmental conditions, it is necessary to use energy storage systems to reduce the consequent power variations. For this reason, batteries are usually a necessary component of HRES [2], that can be complemented by fuel cell (FC)–electrolizer systems [3].

The design of some of the control units, particularly those that act on the switching power supplies that connect the different elements, strongly affect the efficient operation of these systems. Due to the complexity of the resulting mathematical models, it is necessary to use numerical simulations for dimensioning the different components, and for designing and tuning the controllers.

The presence of switching elements in the DC–DC converters operating at high frequencies impose several difficulties to classic numerical integration methods. The reason is that, in order to obtain decent results, the algorithms must perform several calculations to compute the time of each discontinuity [4], restarting the simulation after the occurrence of each event. Moreover, realistic representation of the switching elements (diodes and transistors) may results in stiff models (i.e., with simultaneous slow and fast dynamics) requiring the use of implicit numerical solvers that perform expensive iterations and matrix inversions. Consequently, the simulation of a few minutes of a realistic HRES can take several hours of CPU time even in modern powerful computers.

To overcome this problem, switching converters are usually represented by time–averaged models [5, 6, 7]. This simplification, which is adequate to solve many problems, is limited to particular operating conditions and hide some real phenomenons such as transient discontinuous conduction in the converters, the harmonic content they introduce, the presence of failures in some switching components, etc. Thus, in cases where these phenomenons are relevant, the replacement of power electronic converters by their time–averaged models is not possible and the simulation with conventional numerical solvers experiences the aforementioned problems. In order to make simulations suitable, only a few seconds of the system evolution is actually simulated [8]

However, there is a new family of numerical integration algorithms called Quantized State System (QSS) [4]. These methods replace the time discretization of classic solvers by the quantization of the state variables. A remarkable feature of QSS methods is that they are very efficient in the simulation of ordinary differential equations (ODEs) with frequent discontinuities. There are also Linearly Implicit QSS (LIQSS) methods that are very efficient to simulate some stiff systems [9]. Thus, it can be expected that LIQSS algorithms can efficiently simulate HRES without making use of time–averaged models. In fact, it has been already shown that LIQSS algorithms are very efficient to simulate different topologies of DC-DC converters [10], which constitute a critical component of HRES as well as smart–grid models [11].

A limitation of the QSS methods was that its implementation required the use of specific software tools that were unfriendly to describe complex models such as HRES. However, an autonomous QSS solver [12] was recently developed that can simulate models previously translated from Modelica representations making use of a novel compiler [13]. Modelica [14], is a standard object oriented modeling language where models can be easily defined and composed to form complex systems making use of different available graphical user interfaces and existing multi-domain component libraries. That way, a DC-DC converter, for instance, can be easily modeled by connecting the corresponding electrical components from the existing Modelica electrical library and then it can be used as part of the HRES model. Making use of the mentioned Modelica compiler, the resulting model can be then automatically simulated by the QSS solver.

In this work we first developed a Modelica library of realistic HRES components, including models of PEM fuel cells, electrolyzers, hydrogen storage tanks, batteries, converter controllers and switched models of most typical DC-DC converters. Then, we used the library to build different configurations of HRES and simulated the models using classic solvers and LIQSS methods. The analysis of the simulation results shows that LIQSS can simulate these complex systems in reasonable CPU times (near to real–time, in fact), speeding up more than 10 times the results of classic ODE solvers.

The paper is organized as follows: Section 2 introduces the modeling and simulation tools used along the rest of the work, then Section 3 describes the HRES, their components and the corresponding Modelica library. Section 4 shows and discusses the simulation results, and finally, Section 5 concludes the article.

2. Modeling and Simulation Tools

In this section we introduce the tools used along the article. We first describe the Modelica language used to define the libraries and models. Then, we recall the main features of classic ODE solvers and their difficulties to simulate HRES models and we introduce the QSS family of numerical algorithms. Finally, we present a tool chain that allows to simulate Modelica models using the QSS methods.

2.1. Modelica

Modelica [14] is an open object–oriented declarative modeling language that allows the combination of models coming from different technical domains in a unified way. In Modelica, elementary mathematical relationships between variables are described by non–causal equations to form basic subsystems, that are then connected together to compose more complex systems. Then, for simulation purposes, the resulting models are processed by Modelica compilers in order to produce the simulation code.

For instance, an electrical connector can be defined by the following Modelica class:

```
connector pin
  Real v;        //potential
  flow Real i;   //current
end pin;
```

Here, `pin` is a class of type connector characterized by two real variables representing the potential and current. This new class can be used to define a generic one–port element composed by two pins as follows:

```
model oneport
  pin p;         //positive pin
  pin n;         //negative pin
  Real v;        //element voltage
  Real i;  //element current
equation
  i=p.i;
  p.i+n.i=0;
  v=p.v-n.v;
end oneport;
```

This generic one–port model can be used to derive specific elements like resistors, inductors, etc:

```
model resistor
  extends oneport;
  parameter Real R=1;
equation
  v-R*i=0;       //Ohms law
```

```
end resistor;

model inductor
  extends oneport;
  parameter Real L=1;
equation
  L*der(i)=v;    //Faraday law
end inductor;
```

An RL circuit can be then constructed as follows:

```
model RL_Circuit
  resistor R;
  inductor L;
equation
  connect(R.p,L.p);
  connect(R.n,L.n);
end RL_circuit
```

This last stage of the modeling task consisting in connecting together the components is usually done with the help of Modelica graphical user interfaces, where the modeling practitioner only has to drag and drop the elements and connect them. Figure 1, for instance, shows a Boost converter circuit built in a Modelica software tool called Dymola [15] using components like those described above.

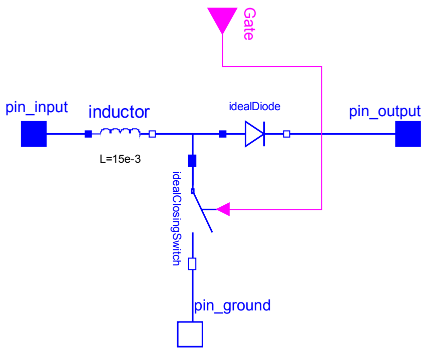

Figure 1: DC-DC Boost converter

The non–causal formulation of the model equations together with the object–oriented paradigm enable the reuse of code and has been used to develop the Modelica Standard Library (MSL), a repository of model components from different technical domains (mechanical, electrical, electronic, hydraulic, thermal, etc.) that can be used to build models by dragging, dropping and connecting them. The MSL is an open library maintained by the Modelica Association, a non–profit organization in charge of developing the language. Besides the MSL there are several other libraries (both, free and proprietary) comprising many technical domains and industrial applications.

Regarding renewable energy models, there are some previous works reporting the use of Modelica to the field [16, 17, 18, 19, 20] showing that the language is appropriate for modeling these systems.

Once a model like the `RL_circuit` is built, it can be simulated. For that goal, a Modelica compiler collects all the equations involved in the model obtaining a set of *Differential–Algebraic Equations* (DAEs) that are then sorted and processed to form a set of ODEs which is simulated by an ODE solver.

Currently, there are various available Modelica compilers, both commercial (like Dymola [15] and Wolfram SystemModeler) and open source (OpenModelica [21], JModelica [22]). Most of them have also graphical user interfaces allowing to build models in a drag and drop fashion.

2.2. Classic ODE Solvers and HRES Simulation

Given an ODE of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{1}$$

where $\mathbf{x}(t)$ is the vector of state variables, classic numerical integration algorithm solve this equation based on time discretization. That is, they compute an approximate solution at certain time points t_0, t_1, $\cdots$, t_N. These time points can be equidistant (fixed step methods) or they can be adjusted to fulfill error tolerance settings (variable step methods).

The approximation performed by a numerical algorithm coincides with the Taylor series expression of the solution of Eq.(1) up to certain power defining the *order* of the algorithm. Higher order solvers usually require more calculations at each time step, but they can perform longer steps without increasing the numerical error. In most engineering applications such as HRES, the optimal balance between computational load and numerical accuracy is given by algorithms of order between 3 and 5 [4]. For this reason, the fifth order algorithms of DOPRI [23] and DASSL [24] are the most efficient and popular solvers for this type of problems.

DOPRI is an explicit fifth–order variable step Runge-Kutta algorithm, while DASSL is an implicit variable step Backward Difference Formulae (BDF) method. Due to stability reasons, DOPRI (as any other explicit algorithm) cannot efficiently integrate stiff systems, i.e., systems with simultaneous slow and fast dynamics. Since stiffness is a very common phenomenon in multi–domain applications, DASSL is a priori the preferred solver of Modelica tools. Implicit solvers like DASSL have advantages regarding stability which are essential to simulate stiff systems, but they add an extra computational load as they must invert the Jacobian matrix of the system and iterate at each step.

Besides stiffness, realistic HRES models also exhibit frequent discontinuities produced by the switched power converters. Since numerical algorithms cannot integrate across discontinuities without provoking unacceptable errors, the solvers must detect their occurrence finding the exact time point restarting the simulation after each event. The process of event detection and discontinuity handling usually requires iterations and, together with the stiffness issues, imply that the simulation of realistic HRES models becomes very inefficient.

2.3. Quantized State System Methods

Given the ODE of Eq.(1) the first order Quantized State System method (QSS1) [25] approximates it by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), t) \tag{2}$$

Here, $\mathbf{q}$ is the *quantized state vector*. Its entries are component-wise related with those of the state vector $\mathbf{x}$ by the *hysteretic quantization function*, so that the components $q_j(t)$ only change when they differ from $x_j(t)$ in a quantity ΔQ_j called *quantum*.

The QSS1 method has the following features:

- The quantized states $q_j(t)$ follow piecewise constant trajectories, and the state variables $x_j(t)$ follow piecewise linear trajectories.

- The state and quantized variables never differ more than the quantum ΔQ_j. This fact ensures stability and global error bound properties [25, 4].

- Each step is local to a state variable x_j (the one which reaches the quantum change), and it only provokes calculations on the state derivatives that explicitly depend on it.

- The fact that the state variables follow piecewise linear trajectories makes very easy to detect discontinuities. Moreover, after a discontinuity is detected, its effects are not different to those of a normal step. Thus, QSS1 is very efficient to simulate discontinuous systems [4].

However, QSS1 has some limitations as it only performs a first order approximation, and it is not suitable to simulate stiff systems. The first limitation was solved with the introduction of higher order QSS methods like the second order accurate QSS2, where the quantized state follow piecewise linear trajectories.

Regarding stiff systems, a family of Linearly Implicit QSS (LIQSS) methods of order 1 to 3 was proposed in [9]. LIQSS methods have the same advantages of QSS methods, and they are able to efficiently integrate many stiff systems, provided that the stiffness is due to the presence of large entries in the main diagonal of the Jacobian matrix. Unlike classic stiff solvers, LIQSS methods are explicit algorithms

In the context of realistic HRES simulation, the explicit treatment of stiff systems and the efficient handling of discontinuities constitute the main advantages of the QSS methods.

2.4. QSS Stand Alone Solver

The first implementations of QSS methods were based on the DEVS formalism [4]. Recently, the complete family of QSS methods was implemented in a *stand–alone QSS solver* [12] that improves DEVS–based simulation times in more than one order the magnitude. In addition, the QSS Solver implements very efficient versions of DASSL and DOPRI.

The stand–alone QSS solver requires that the models are described in a subset of the Modelica language called μ-Modelica [12], where the equations are given in its ODE form.

2.5. QSS Simulation of Modelica Models

In spite of some preliminary attempts to include QSS methods in Open-Modelica [26], none of the popular Modelica software tools allow to simulate using these algorithms.

Recently, the group developed ModelicaCC [13], a Modelica compiler which has some unique features (vectorized flattening and equation sorting) and generates code specially targeted for the QSS Solver, i.e., it translates a generic Modelica model into μ-Modelica.

3. HRES Modelica Library

In this section we describe the HRES components that constitute the new Modelica library. We first present a possible HRES configuration and then we introduce the models corresponding to the different sub–systems. Finally, using this library, we built a complete HRES model.

3.1. HRES Scheme

HRES are composed by various types of power sources and energy storing devices that are able to supply a load. Primary power sources are generally photovoltaic (PV) modules and/or wind power generators, while the combination PEM fuel cells, electrolyzer, hydrogen storage tanks and batteries are used as backup and storage systems. All these elements are usually connected to a direct-current bus through power converters.

Figure 2 shows a possible HRES configuration.

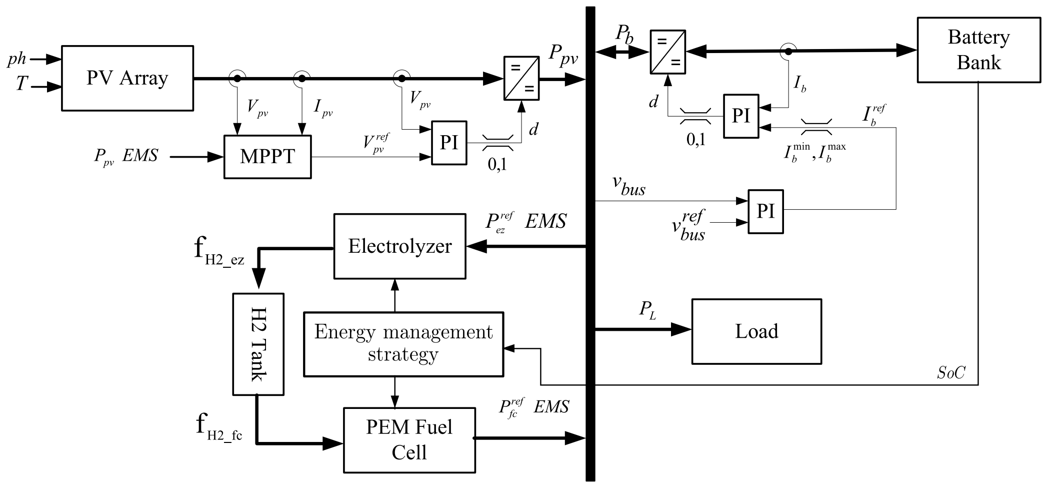

Figure 2: HRES scheme

3.2. DC-DC Converters

DC-DC converters are electronic devices that allow to isolate the voltage changes produced in the power sources from the constant bus voltage. The voltage conversion is made by high frequency switching components implemented with transistors or diodes. There exist multiple converters topologies, the most typical are the *Buck* or reducer, *Boost* or elevator and *Buck-Boost* or reducer-elevator.

The Modelica HRES library contains models of the different converter topologies, built using electrical components of the Modelica Standard Library (inductors, diodes, switches, etc). Figure 1 shows the Boost converter (used to control the unidirectional power flow of the PV arrays, electrolyzer and PEM fuel cell) and Figure 3 shows the bidirectional Buck-Boost converter used to control the power flow that charges and discharges the batteries.

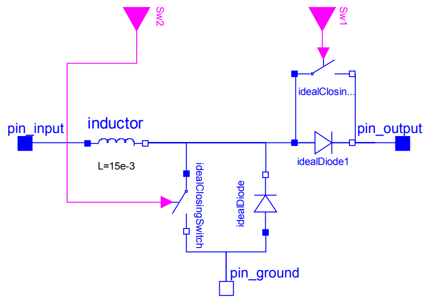

Figure 3: DC–DC Buck–Boost bidirectional converter

The converter models include realistic features that take into account the possibility of entering in Discontinuous Conduction Mode, an undesirable situation that usually occurs at start–up or during transient evolutions. Typical time–averaged models cannot represent this situation.

3.3. Photovoltaic Arrays

PV cells have voltage-current and power-current nonlinear characteristics strongly dependent on insolation and temperature. According to [27], a possible expression for the output current $I_{pv}(t)$ of a PV cell is given by:

$$I_{pv}(t) = I_{ph}(t) - I_{rs}(t)\left(\exp\left(\frac{q\left(V_{pv}(t) + I_{pv}(t)R_s\right)}{A_c K T(t)}\right) - 1\right) \tag{3}$$

where $I_{ph}(t)$ is the generated current under a given insolation, I_{rs} is the cell reverse saturation current, V_{pv} is the voltage level on the PV cell terminals, q is the charge of an electron, R_s is the intrinsic cell resistance, A_c is the cell deviation from the ideal p-n junction characteristic, K is the Boltzman constant, and T is the cell temperature. $I_{ph}(t)$ depends on the insolation and temperature according to the following expression:

$$I_{ph}(t) = (I_{sc} + K_l(T(t) - T_r))\lambda(t)/100, \tag{4}$$

where I_{sc} is the short-circuit cell current at the reference temperature and insolation, K_l is the short-circuit current temperature coefficient and λ is the insolation measured in $mW\,cm^{-2}$.

The reverse saturation current depends on temperature according to the following expression:

$$I_{rs}(t) = I_{or}\left(\frac{T(t)}{T_{ref}}\right)^3 \exp\left(\frac{qE_{go}(1/T_r - 1/T(t))}{KA_c}\right), \tag{5}$$

where I_{or} is the reverse saturation current at the reference temperature T_{ref} and E_{go} is the band-gap energy of the semiconductor used in the cell.

PV cells are connected in serial-parallel configurations forming modules, which are the typical commercial units. In order to reach appropriate voltage and power levels, modules can be arranged with a similar architecture on arrays [28]. Solar power systems are composed of a PV array connected to the DC bus through a DC/DC power converter. Thus, the available current for a PV module can be expressed as follows:

$$I_{PV}^{av}(t) = n_p I_{ph}(t) - n_p I_{rs}(t)\left(\exp\left(\frac{q\left(V_{PV}(t) + I_{pv}(t)R_s\right)}{n_s A_c K T(t)}\right) - 1\right), \tag{6}$$

where V_{PV} is the voltage level in the PV module terminals, n_p is the number of parallel strings and n_s is the number of serial connected cells per string.

With Equations (3)–(6), a PV module can be modeled in Modelica as a one–port circuit component, with an additional input port that receives the insolation signal:

```
model PV_module
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  parameter Real q=1.6e-19; //[C]
  parameter Real Ac=1.6;
  parameter Real K=1.3805e-23;//[Nm/K]
  parameter Real K1=5.532e-3;//[ A/oC]
  parameter Real Ior=1.0647e-6;// [A]
  parameter Real Tref=303;// [K]
  parameter Real Eg=1.1;// [V]
  parameter Real Isc=8.51;// [A]
  parameter Real Rspv=0.01;
  parameter Real Tpv=273+25;
  parameter Real Irs=Ior*(Tpv/Tref)^3
  *exp(q*Eg*(1/Tref-1/Tpv)/K/Ac);
  parameter Integer Np=1;
  parameter Integer Ns=60;
  Real Iph; //insolation current
  Real exponent;
  Real Ipv;
  Real Vpv;
  Real lambdaph;
  Modelica.Blocks.Interfaces.RealInput u

equation
```

```
  lambdaph=u;     //insolation
  Iph=(Isc+K1*(Tpv-Tref))*lambdaph/100;
  Ipv+i=0;
  Vpv=v;
  Ipv-Np*Iph+Np*Irs*(exp(exponent)-1)=0;
  exponent=q*(Vpv/Ns+Ipv*Rspv/Np)/(K*Ac*Tpv);
end PV_Module;
```

3.4. MPPT algorithm

In a PV module, the generated power depends on the solar radiation, the temperature, and the module voltage. Given the values of insolation and temperature, there is an optimal voltage at which the maximum power is obtained. There are algorithms, called maximum power point tracking (MPPT), that are capable of computing this optimal voltage. In our HRES Library, the MPPT *IncCond* algorithm presented in [29] was implemented as a Modelica model:

```
model mppt
  Modelica.Blocks.Interfaces.RealInput i;
  Modelica.Blocks.Interfaces.RealInput u;
  Modelica.Blocks.Interfaces.RealOutput y;
  Real pot;
  Real potActFiltrada;
  discrete Real potact;
  discrete Real potprev;
  discrete Real prevu;
  discrete Real actu;
  discrete Real deltau;
  discrete Real deltap;
  discrete Real vref(start=30);
  parameter Real Ts=0.1;
  parameter Real deltaVpvRefPanel=0.5;

equation
  pot=u*i;
  der(potActFiltrada)=(pot-potActFiltrada)*100;
  y=pre(vref);

algorithm
  when sample(0,Ts) then
    potprev:=potact;
    potact:=potActFiltrada;
    prevu:=actu;
    actu:=u;
    deltau:=actu-prevu;
    deltap:=potact-potprev;
    if abs(deltau)>0.1*deltaVpvRefPanel then
      if abs(deltap)>0.2 then
        if deltap/deltau>0 then
          vref:=vref+deltaVpvRefPanel;
        end if;
        if deltap/deltau<0 then
```

```
          vref:=vref-deltaVpvRefPanel;
        end if;
      end if;
    end if;
  end when;
end mppt;
```

3.5. PEM fuel cell

In this paper, a static isothermal model of the PEM fuel cell is used. The internal potential is given by the Nernst Equation [30]:

$$E_{cell}(t) = E_{0,cell}(t) + \frac{RT}{2F}\log(p_{H_2}(t)\sqrt{p_{O_2}(t)}) \tag{7}$$

where $E_{0,cell}$ is the reference potential, R is the gas constant $(8.3143 J/molK)$, T is the fuel cell temperature, F is the Faraday constant $(96,487 C/mol)$, p_{H_2} is the hydrogen pressure and $p_{O_2}(t)$ is the oxygen pressure. The reference potential $E_{0,cell}(t)$ depends on the temperature according to

$$E_{0,cell} = E_{0,cell^o} - k_E(T - 298) \tag{8}$$

where $E_{0,cell^o}$ is the reference potential at standard conditions (1.229 V at 25°C and 1 atm) and k_E is a constant $(8.5e^{-4} V/K)$.

The output voltage of the fuel cell is less than the internal voltage E_{cell} due to the presence of the activation voltage drop, the ohmic voltage drop, and the concentration voltage drop.

The activation voltage losses can be approximated by the following expression

$$V_{act}(t) = \eta_0 + (T - 298)\cdot a + V_{act2}(t) + R_{act}(t)I_{fc}(t) \tag{9}$$

with η_0, a, and b being empirical constants whereas $R_{act}(t)$ depends on the current and temperature according to a polynomial approximation.

The ohmic voltage drop is expressed by:

$$V_{ohm}(t) = R_{ohm}(t)I_{fc}(t) = (R_{ohm0} + R_{ohm1}(t) + R_{ohm2})I_{fc}(t) \tag{10}$$

where R_{ohm0} is an empirical constant, whereas $R_{ohm1}(t)$ depends on the current and R_{ohm2} depends on the temperature.

The concentration voltage drop is given by:

$$V_{conc}(t) = -\frac{RT}{\alpha\eta F}\log\left(1 - \frac{I_{fc}(t)}{I_{limit}}\right) \tag{11}$$

where I_{limit} is the fuel cell current limit.

The stack potential is the result of the sum of the n_{fc} cell potentials ($E_{stack} = n_{fc}E_{cell}$). And the voltage at the fuel cell terminals is:

$$V_{fc,out}(t) = E_{stack} - V_{act}(t) - V_{ohm}(t) - V_{conc}(t) \tag{12}$$

In this work a Nexa 1.2 kW PEMFC system is adopted and the parameter identification procedure developed in [31] is used, where a polynomial expression approximates $V_{act} + V_{ohm}$.

Finally the hydrogen consumption rate ($\dot{\eta}_{H_2,fc}$) is a function of the fuel cell current:

$$\dot{\eta}_{H_2,fc}(t) = \frac{n_{fc}I_{fc}(t)}{2F} \tag{13}$$

Based on Eqs.(7)–(13), we built the following Modelica representation of the PEM fuel cell:

```
model FuelCell
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  Modelica.Blocks.Interfaces.RealInput Pref_FC;
  Modelica.Blocks.Interfaces.RealOutput fH2_FC;
  Real Estack;
  Real Vohm;
  Real Vconc;
  Real E0_cell_std; //Standard reference potential
  Real E0_cell;      //Reference potential
  Real E_cell;     //Nernst Equation (Ideal)
  Real Vact;
  Real React0;
  Real React1;
  Real React2;
  Real IFC;
  Real UFC;
  parameter Real p_H2 = 0.3; //pressure of hydrogen [atm]
  parameter Real p_O2 = 1;   //pressure of oxygen [atm]
  parameter Real Nc = 47; //number of series cell
  parameter Real R = 8.3143; //Gas constant [J/mol*K]
  parameter Real F = 96487; //Faraday constant [C/mol]
  parameter Real ke = 8.5e-4; //Constant [V/K]
  parameter Real deltaG = - 237153.66;  //Gibbs free energy per mole of reaction [J/mol]
  parameter Real ne = 2; //number of electrons in reaction
  parameter Real nu_0 = 26.5230; //[V]
  parameter Real a = 8.9224e-2;
  parameter Real I_limit = 75; // Fuel cell current limit [A]
  parameter Real T=298;
  parameter Real MH2=2.016e-3; //[Kg/mol] Molar mass of oxygen

equation
  Pref_FC=i*u;
  E0_cell_std = -deltaG/(ne*F);
  E0_cell=E0_cell_std - ke*(T-298);
  E_cell = E0_cell + (R*T/(2*F))*Modelica.Math.log(p_H2*p_O2^(0.5));
  Estack = E_cell*Nc;
  React0 = -1.0526;
  React1 = (6.945e-11)*IFC^6 - (1.7272e-8)*IFC^5 + (1.7772e-6)*IFC^4 -
           (9.8133e-5)*IFC^3 + (3.1430e-3)*IFC^2 - (3.5320e-2)*IFC;
  React2 = (1.3899e-3)* (T - 298);
  Vact=nu_0 - (T - 298)*a+(React0 + React1 + React2)*IFC;
  Vohm = IFC*(1.7941-(2.3081e-2)*IFC-(2.0060e-3)*(T - 298));
  Vconc = - (R*T/(ne*F))*Modelica.Math.log(1-(IFC/I_limit));
  IFC=Pref_FC/UFC;
  UFC = Estack - Vact - Vohm - Vconc;
  fH2_FC=MH2*Nc*IFC/2/F;
end FuelCell;
```

3.6. Electrolyzer

The voltage–current relation of a PEM electrolyzer can be modeled by the following temperature dependent equation proposed by Ulleberg [32]:

$$V_{cell,ez}(t) = V_{rev,ez} + \frac{r_1 + r_2 T}{A_{ez}} I_{ez}(t) + (s_1 + s_2 T + s_3 T^2) \log\left(\frac{t_1 + \frac{t_2}{T} + \frac{t_3}{T^2}}{A_{ez}} I_{ez}(t) + 1 \right) \tag{14}$$

where $V_{cell,ez}$ is the cell voltage, $V_{rev,ez}$ is the reversible voltage, I_{ez} is the electrolyzer current, A_{ez} is the area of electrode, and T is the temperature. r_i, s_i, and t_i are empirical parameters.

The electrolyzer power P_{ez} can be then computed as

$$P_{ez}(t) = n_c V_{cell,ez}(t) I_{ez}(t) \tag{15}$$

where n_c is the number of cells in serial connection. This power allows to compute the hydrogen production rate $\dot{\eta}_{H_2,ez}$ according to Faraday's Law:

$$\dot{\eta}_{H_2,ez}(t) = \eta_f \frac{P_{ez}(t)}{V_{cell,ez}(t) \cdot 2 \cdot F} \tag{16}$$

where η_f is the Faraday efficiency (usually between 80–90%) and F the Faraday constant.

Using Eqs.(14)–(16) the following Modelica model can be built:

```
model electrolyzer
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  parameter Real F = 96485; //[C mol^-1]
  parameter Real Urev = 1.229; //[V]
  parameter Real r1 = 7.331e-5;//[ohm m^2]
  parameter Real r2 = -1.107e-7;//[ohm m^2 C-1]
  parameter Real r3 = 0;
  parameter Real s1 = 1.586e-1;//[V]
  parameter Real s2 = 1.378e-3;//[V C-1]
  parameter Real s3 = -1.606e-5;//[V C-2]
  parameter Real t1 = 1.599e-2;//[m^2 A^-1]
  parameter Real t2 = -1.302;//[m^2 A^-1 C-1]
  parameter Real t3 = 4.213e2; //[m^2 A^-1 C-2]
  parameter Real A = 0.25; //[m^2]
  parameter Real nf=1  //Faraday efficiency
  parameter Real nc = 1; //number of serial cells
  parameter Real T =  40;
  Real PotRef;
  Real fH2;
  Real Ucell;
  Modelica.Blocks.Interfaces.RealOutput y;
  Modelica.Blocks.Interfaces.RealInput u1;

equation
  Ucell=u;
  PotRef=u1;
  Ucell = Urev + ((r1+r2*T)*(PotRef/(Ucell*nc))/A) +
  (s1+s2*T+s3*T*T)*Modelica.Math.log10(((t1+t2/T+t3/T/T)*(PotRef/(Ucell*nc))/A)+1);
  PotRef=i*u;
  fH2 = nf*PotRef / (Ucell*2*F);
  y=fH2;
end electrolyzer;
```

3.7. Hydrogen Storage

Hydrogen is usually stored in pressurized tanks. The hydrogen pressure inside the tank can be modeled using the ideal gas equation:

$$p_{H_2}(t) = \frac{RTn_{H_2}(t)}{V_t} = \frac{RT\int(\dot{\eta}_{H_2,ez(t)} - \dot{\eta}_{H_2,fc(t)})\,dt}{V_t} \tag{17}$$

where n_{H_2} is the number of moles of hydrogen in the tank, R is the specific gas constant, T is the temperature and V_t is the volume of the tank. Based on this equation, the following Modelica model represents a pressurized tank:

```
model Tank
  Modelica.Blocks.Interfaces.RealOutput p_tank_bar;
  Modelica.Blocks.Interfaces.RealInput FH2_elect;
  Modelica.Blocks.Interfaces.RealInput FH2_fuelCell;
  parameter Real R=8.314;
  parameter Real T=273+40;
  parameter Real V=0.1;
  Real p_tank(start=8e5); // pressure in [Pa]
equation
  der(p_tank)=R*T/V*(FH2_elect-FH2_fuelCell); //tank presure [Pa]
  p_tank_bar=1e-5*p_tank;  //tank presure [Bar]
end Tank;
```

3.8. Compressor

The work needed to store the hydrogen in the pressurized tank is done by a compressor. The expression of the isoentropic compression power required to elevate the electrolyzer exit pressure to the tank pressure is given by [33]:

$$P_{co}(t) = \frac{1}{\eta_{isoen}\eta_m}\dot{\eta}_{H_2,ez}(t)RT\frac{k}{k-1}\left(\frac{p_{H_2}(t)^{k/_{k-1}}}{p_{ez}} - 1\right) \tag{18}$$

where $P_{co}(t)$ is the electric power consumed by the compressor), $\eta_{isoen} = 0.8$ is the isoentropic efficiency of the compressor, $\eta_m = 0.9$ is the lumped efficiency of the electric motor that drives the compressor, $k \approx 1.4$ is the adiabatic index and p_{ez} is the output pressure of the electrolyzer (in this case, the atmospheric pressure). The compressor can be then modeled by the following Modelica class:

```
model Compressor
  Modelica.Blocks.Interfaces.RealInput FH2_electrolizer;
  Modelica.Blocks.Interfaces.RealInput p_tank;
  Modelica.Blocks.Interfaces.RealOutput Consumed_Pow;
  parameter Real R=8.314;
  parameter Real k=1.4;
  parameter Real nu_iso=0.9;
  parameter Real nu_m=0.8;
  parameter Real T=273+40;
  parameter Real p_elect=1;
equation
  Consumed_Pow=FH2_electrolizer*((p_tank/p_elect)^(1-1/k)-1)*T*R*k/(k-1)/(nu_iso*nu_m);
end Compressor;
```

3.9. Energy storage system

The Energy Storage System is composed of a bank of Lead-Acid Batteries. Each battery is modelled as a controlled voltage source with a serial resistance [34]. The open circuit source voltage is calculated by a non linear equation dependent on the actual charge of the battery ($\int i_b dt$):

$$E_b = E_0 - K\frac{Q}{Q - \int i_b dt} + Ae^{(-B\int i_b dt)}, \tag{19}$$

where E is the open source voltage, E_0 is a constant voltage, K is the polarization voltage, Q is the nominal capacity of the battery, A is the amplitude of the exponential zone, and B is the inverse of the time constant of the exponential zone. This model assumes a constant resistance value during the charge and discharge process. The State of charge of the battery (SoC_B) is defined as:

$$SoC_B = 100\left(1 - \frac{\int i_b dt}{Q}\right) \tag{20}$$

Based on Eqs.(19)–(20), the Battery was represented by the following Modelica model:

```
model Battery
  extends Modelica.Electrical.Analog.Interfaces.OnePort;
  parameter Real E0=12.6463;
  parameter Real Q=0.65;
  parameter Real A=0.66;
  parameter Real B=2884.61;
  parameter Real K=0.33;
  parameter Real R=0.25;
  parameter Real SoC0=90;
  Real Qt;
  Real SoC;
  Real E;
  Modelica.Blocks.Interfaces.RealOutput y;

initial equation
  Qt=(1-SoC0/100)*Q;

equation
  der(Qt)=-i/3600;
  E=E0-K*Q/(Q-Qt)+A*exp(-B*Qt);
  u=E+R*i;
  SoC=(1-Qt/Q)*100;
  y=SoC;
end Battery;
```

3.10. Energy management strategy and bus voltage control

The net power ($P_{net}(t)$) is defined as the difference between the power generated by the PV array and the power consumed in the load. When $P_{net}(t)$ is positive, this power can be used to charge the battery or to produce hydrogen via electrolysis. Otherwise, when $P_{net}(t)$ is negative, the missing power must be supplied by the batteries or the PEM fuel cell. The decision among the different choices (producing hydrogen or charging the battery in the first case, using the battery or consuming hydrogen in the second situation) is taken by an energy management strategy (EMS).

The EMS, based on the knowledge about the battery state and the power balance, is in charge of computing the electrolyzer and fuel cell power reference signals. Then, a closed loop control layer manipulates the duty cycle of the DC-DC converters in order to follow these reference signals.

In a similar way, the bus voltage is regulated by the battery through a closed loop control.

EMS uses hysteresis loops to regulate the energy flows [35] in order to prevent frequent on–off switch commutations. The electrolyzer as well as the fuel cell work in variable power mode to improve system efficiency [36].

The EMS strategy was also implemented as a Modelica model that receives the battery state of charge and the power consumption and generation signals and computes the power reference signals for the fuel cell and the electrolyzer.

3.11. A Complete HRES Model

Making use of the new Modelica library, and following the configuration of Fig.2, we built the HRES model depicted in Figure 4. There, in order to improve the full diagram visualization, some sub–systems of the library were grouped together, forming more complex sub–systems.

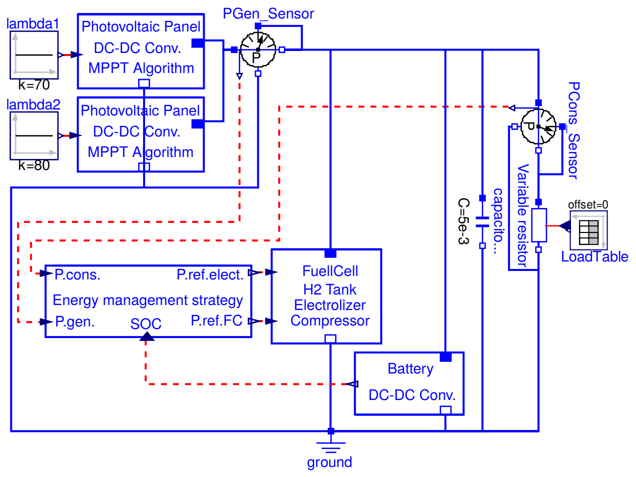

Figure 4: Modelica model of a HRES

For instance, the model labeled as *Fuel Cell / H2 Tank / Electrolyzer Compressor* comprises the models of the PEM Fuel cell, the hydrogen tank and the Electrolyzer as shown in Figure 5.

The set of parameters used in the different sub–systems are listed in Table 1.

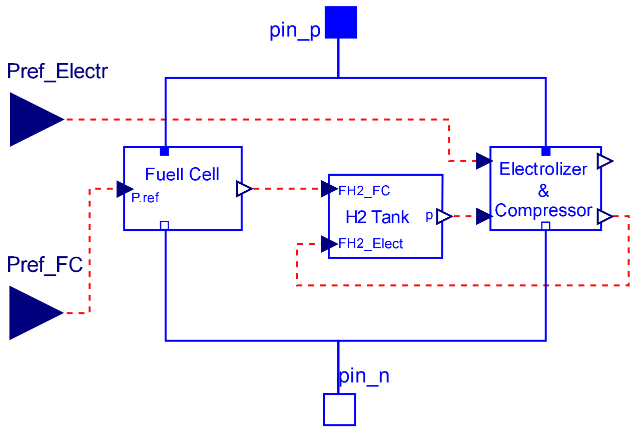

Figure 5: Fuel Cell and Electrolyzer subsystem

4. Simulation Results

In this section, we simulate the previously described HRES model under various conditions comparing the performance of different ODE solvers.

4.1. Experimental Setup

We simulated the complete HRES model of Figure 4 using the set of parameters listed in Figure 1. We also introduced some modifications to the model to evaluate the performance of the ODE solvers under the appearance of different phenomenons: first, we added a small inductance to the originally purely resistive load, what provokes that the system becomes stiff. Then, we added more PV modules to the model what increases the size of the problem.

In all cases, the simulations were performed under the following conditions

- The models were first processed by the ModelicaCC compiler, converting them into μ–Modelica language.
- The resulting μ–Modelica models were then simulated using DASSL, DOPRI and LIQSS2 algorithms implemented in the Stand Alone QSS Solver.
- Dymola and OpenModelica implementations of DASSL were also tested, but the simulation times obtained were greater than those of the QSS-Solver, so only the results obtained by the later tool are reported.

Parameter	Value
DC-DC converters	
Inductor (L)	$15e^{-3}$ Hy
Switching frequency (f_s)	10 kHz
PV Panel	
Electron charge (q)	1.6e-19 C
Constant of p-n junction characteristc (A_c)	1.6
Boltzmann constant (K)	1.3805e-23 Nm/K
Short circuit coefficient (K_l)	5.532e-3 A/K
Reverse saturation current (I_{or})	1.0647e-6 A
Reference temperature (T_{ref})	303 K
Band-gap energy (E_g)	1.1 V
Short circuit current (I_{sc})	8.51 A
Intrinsic cell resistance (R_s)	0.01 Ω
Reference temperature (T_{ref})	303 K
Number of parallel strings (n_p)	1
Number of serial cells per string (n_s)	60
PEM Fuel Cell - Nexa 1.2 kW	
Pressure of hydrogen (p_{H_2})	0.3 atm
Pressure of oxygen (p_{O_2})	1 atm
Number of series cell (n_{fc})	47
Gas constant (R)	8.3143 J/mol.K
Faraday constant (F)	96487 C/mol
Fuel Cell constant (F)	$8.5e^{-4}$ V/K
Reference potential ($E_{0,cell0}$)	1.229 V
Number of electrons in reaction (η)	2
Constant η_0	26.5230 V
Constant a	8.9224e-2
Activation equivalent resistance polynomial (R_{act})	$-1.0526 + 6.945e^{-11} \cdot I^6 - 1.7272e^{-8} \cdot I^5$ $+1.7772e^{-6} \cdot I^4 - 9.8133e^{-5} \cdot I^3$ $+3.1430e^{-3} \cdot I^2 - 3.5320e^{-2} \cdot I$ $+1.3899e^{-3} \cdot (T - 298)$
Ohmic resistance constant (R_{ohm0})	1.7941
Ohmic current dependent resistance (R_{ohm1})	$-2.3081e^{-2} \cdot I$
Ohmic temperature dependent resistance (R_{ohm2})	$-2.0060e^{-3}(T - 298)$
Fuel cell current limit (I_{limit})	75 A
Number of cells in the stack (n_{fc})	47
Electrolyzer	
Reversible voltage ($V_{rev,ez}$)	1.229 V
Parameter r_1	$7.331e^{-5}$ Ω m^2
Parameter r_2	$-1.107e^{-7}$ Ω m^2 C^{-1}
Parameter r_3	0
Parameter s_1	$1.586e^{-1}$ V
Parameter s_2	$1.378e^{-3}$ V C^{-1}
Parameter s_3	$-1.606e^{-5}$ V C^{-2}
Parameter t_1	$1.599e^{-2}$ m^2 A^{-1}
Parameter t_2	-1.302 m^2 A^{-1} C^{-1}
Parameter t_3	$4.213e^2$ m^2 A^{-1} C^{-2}
Area of electrode (A_{ez})	0.25 m^2
Number of serial cells (n_c)	21
Pressurized tank	
Volume (V_t)	10 m^3
Energy Storage System - Battery	
Constant voltage (E_0)	12.6463 V
Polarization voltage (K)	0.33 V
Nominal capacity (Q)	0.65 Ah
Amplitude of exponential zone (A)	0.66 V
Inverse of exponential zone time constant (B)	2884.61 Ah^{-1}

Table 1: Models parameters

- We made simulations under three different tolerance settings: the typical relative tolerance of 10^{-3} and more stringent tolerances of 10^{-4} and 10^{-5}.

- The final simulation time was 3000 seconds.

- The simulations were performed on a PC with Ubuntu OS and Intel (R) Core (TM) i7-3770 CPU @ 3.40GHz processor.

- Errors were measured comparing the trajectories of the different simulations against reference trajectories using the formula

$$e_{rr} = \sqrt{\frac{\sum \left(i_{L_{bat}}[k] - \hat{i}_{L_{bat}}[k] \right)^2}{\sum \hat{i}^2_{L_{bat}}[k]}} \tag{21}$$

  where $i_{L_{bat}[k]}$, the current in the inductance of the DC-DC converter that connect the battery to the DC bus, was the variable chosen to evaluate the relative error. The reference $\hat{i}_{L_{bat}}[k]$ used to calculate the error was obtained by simulating the models using DASSL with a relative tolerance of 10^{-8}.

4.2. Case 1: A Purely Resistive Load

We first simulated the system of Fig.4 with a purely resistive load. The radiation was fixed in 800 $W \cdot m^{-2}$ and 1000 $W \cdot m^{-2}$ for each panel, resulting through the MPPT algorithm in 400 W of generated power. The load is a resistor $R = 100\ \Omega$, in parallel with a variable resistance representing a variable power consumption. This variable resistance takes the values $R_v(t = 0) = 1005\ \Omega$, $R_v(t = 1150) = 10\ \Omega$, $R_v(t = 1450) = 5\ \Omega$, and $R_v(t = 2500) = 1005\ \Omega$.

Figure 6 shows the evolution of the power generated and consumed at the different subsystems. It can be seen in the main figure that the power balance is guaranteed by the EMS. The long simulation time hides fast transient changes. The upper left sub-figure shows a detail of the power transient when the fuel cell switches on.

The SoC battery evolution and the working mode of the electrolyzer and fuel cell is depicted in Figure 7. The double hysteresis EMS behavior can be understood from this figure. When the SoC level reaches 80%, the electrolyzer switches on and it remains in this state until the SoC level falls to

76%. The onoff state of the fuel cell is managed in a similar way when the SoC level reaches 40% and 44% respectively.

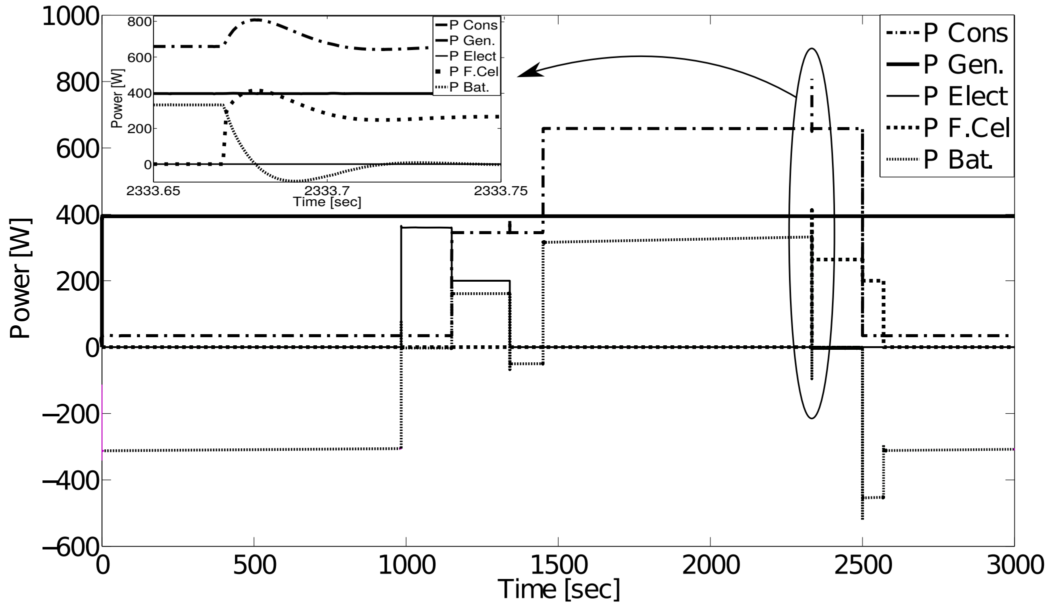

Figure 6: Power balance

Figure 8 shows the DC bus voltage evolution. It can be seen that the mean value of this voltage is controlled around 56V, but it exhibits transitory changes whenever the electrolyzer or the fuel cell state changes or when the load is modified. A detailed view of the bus voltage evolution during a transient state can be seen in the upper left side of the Figure. Also, the upper right corner of the figure shows an even more detailed view of the bus voltage exhibiting the ripple introduced by the different DC-DC converters.

Fig. 9 shows the trajectory of the inductor current $i_L(t)$ at of one of the DC-DC converters that interconnects a photovoltaic panel with the DC bus. The Figure shows that the converter operates in discontinuous conduction during transient evolutions.

While similar trajectories to those of Fig.6–7 can be obtained from simpler time–average models, the detailed ripple signals of Figs.8–9 require the usage of realistic models like the ones used in the new library.

Regarding simulation performance, Table 2 compares the CPU time and the number of scalar function evaluations (i.e. the number of times that each component of the right hand side of Eq.(1) is invoked by the solver) taken by

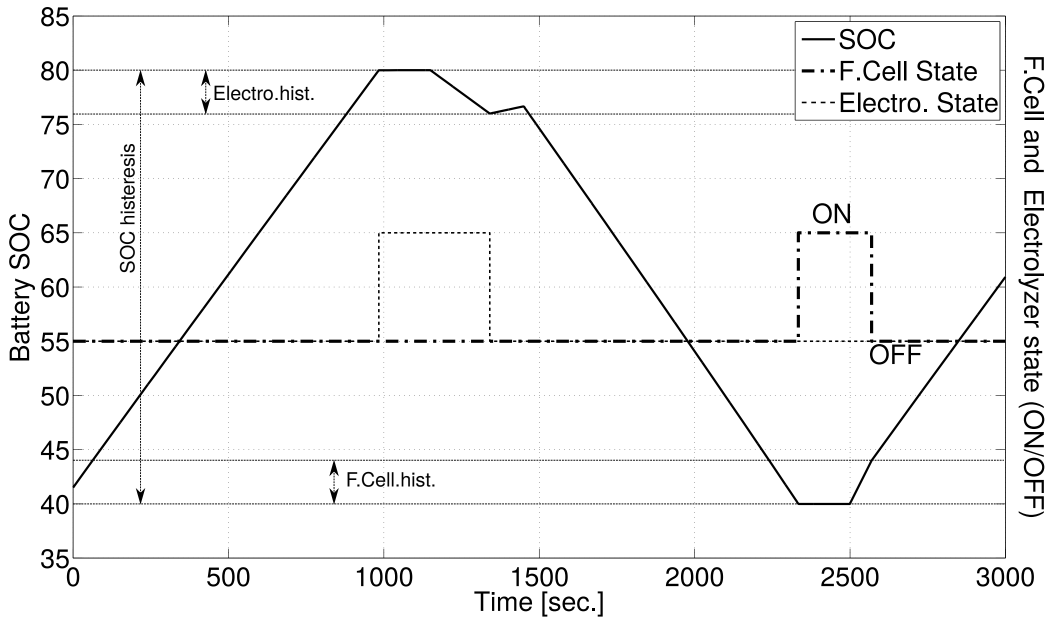

Figure 7: Battery SOC

DOPRI and LIQSS2 for different tolerance settings. DASSL results are not reported because they are more than 10 times slower than LIQSS2 (in this simulation, the system is not very stiff and the usage of an implicit solver like DASSL is not actually necessary).

Integration method		Relative error	Function (f_i) evaluations	CPU [min.]
DOPRI	err.tol=10^{-3}	$1.16 \cdot 10^{-4}$	$2.37 \cdot 10^{10}$	235.0
	err.tol.=10^{-4}	$1.07 \cdot 10^{-5}$	$2.38 \cdot 10^{10}$	233.3
	err.tol.=10^{-5}	$8.15 \cdot 10^{-7}$	$2.40 \cdot 10^{10}$	238.3
LIQSS2	err.tol=10^{-3}	$5.01 \cdot 10^{-3}$	$1.25 \cdot 10^{10}$	79.0
	err.tol=10^{-4}	$6.42 \cdot 10^{-5}$	$1.42 \cdot 10^{10}$	106.5
	err.tol=10^{-5}	$1.41 \cdot 10^{-5}$	$1.97 \cdot 10^{10}$	166.3

Table 2: Simulation Performance of LIQSS and DOPRI (non–stiff case)

It can be seen that both methods fulfill the accuracy requirements although DOPRI errors are much lower than requested. This can be explained by the fact that the step size is shortened by the presence of discontinuities, producing a very small error.

For a typical relative error tolerance of 10^{-3}, LIQSS2 is about 3 times

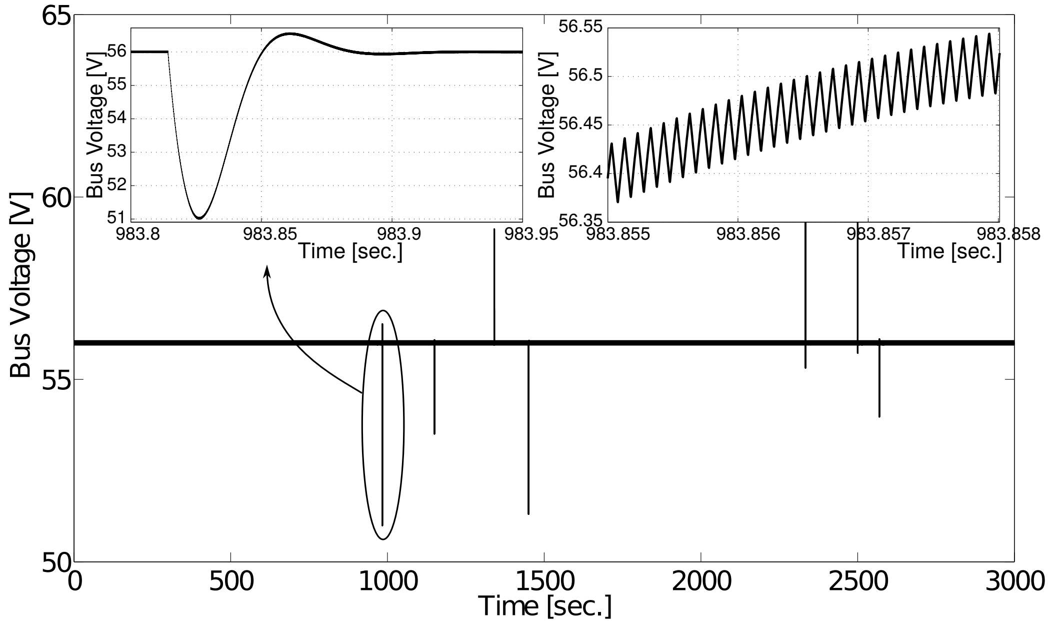

Figure 8: DC bus Voltage

faster than DOPRI. Then, for more stringent tolerance settings, this advantage tends to disappear. This can be explained by the fact that LIQSS is only second order accurate while DOPRI is 5th order accurate. Thus, obtaining higher accuracy requires much more steps in LIQSS2.

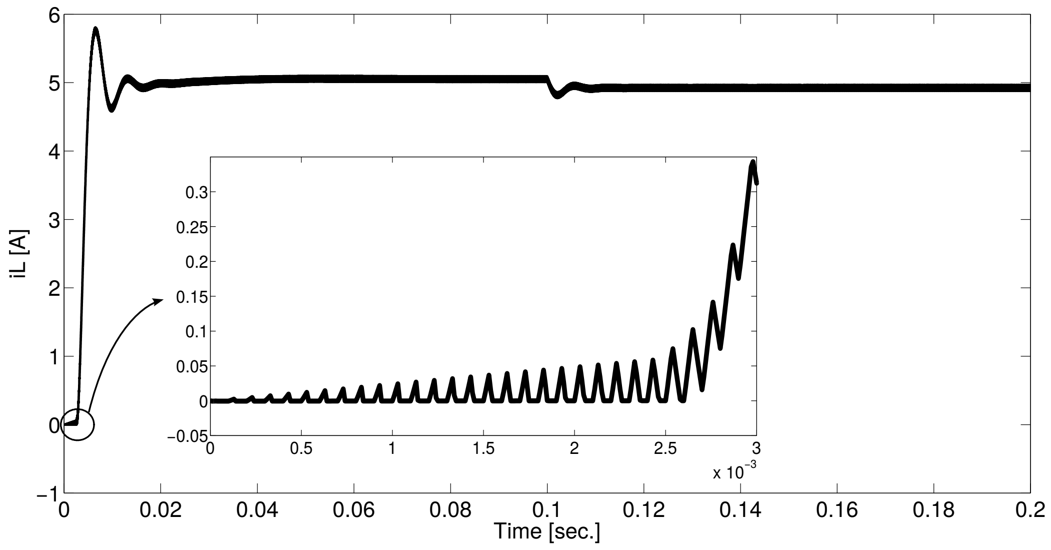

Figure 9: Inductor Current at a Boost Converter

4.3. Case 2: Stiff RL Load

In the previous example, the model was not actually stiff (except during some short time intervals when the power electronic converters entered in discontinuous conduction model). More realistic models usually consider the presence of parasitic inductances and/or capacitors that invariantly lead to stiffness.

In order to evaluate the performance of the algorithms under these conditions, we modified the load including a small inductance of $L = 10^{-5}Hy$ connected in series with the fixed resistor of $R = 100\Omega$, and repeated the experiments of the previous case.

This time, due to stiffness reasons, DOPRI failed to provide results in a reasonable CPU time. Thus, we only compared the results of DASSL and LIQSS2 summarized in Table 3.

Now, LIQSS2 is from 15 to 20 times faster than DASSL. The implicit nature of DASSL implies that it must solve a set of algebraic equations during each step, paying for that a very high computational cost. Then, the fact that LIQSS2 is explicit explains the huge difference between both solvers.

Integration method		Relative error	Function (f_i) evaluations	CPU [min.]
DASSL	err.tol=10^{-3}	$9.6 \cdot 10^{-4}$	$1.16 \cdot 10^{10}$	1395.0
	err.tol.=10^{-4}	$9.5 \cdot 10^{-4}$	$3.34 \cdot 10^{11}$	3050.0
	err.tol.=10^{-5}	$7.1 \cdot 10^{-4}$	$5.31 \cdot 10^{11}$	3783.3
LIQSS2	err.tol=10^{-3}	$8.9 \cdot 10^{-3}$	$1.30 \cdot 10^{10}$	96.0
	err.tol=10^{-4}	$7.2 \cdot 10^{-5}$	$1.68 \cdot 10^{10}$	141.5
	err.tol=10^{-5}	$3.4 \cdot 10^{-5}$	$2.23 \cdot 10^{10}$	205.0

Table 3: Simulation Performance of LIQSS and DASSL (stiff case)

4.4. Case 3: Several PV modules

Based on the previous model (with the stiff RL load), we studied the effects of increasing the number of PV modules on the computational load of each solver.

This time, the final simulation time was reduced to $t_f = 100$ seconds in order to obtain faster results, and we used an error tolerance of 10^{-4}.

Figure 10 plots the relation between the CPU time and the number of panels, showing that the computational cost of LIQSS2 grows linearly with the number of panels while the cost of DASSL grows approximately in a cubic way. Consequently, LIQSS2 with 8 panels results 65 times faster than DASSL.

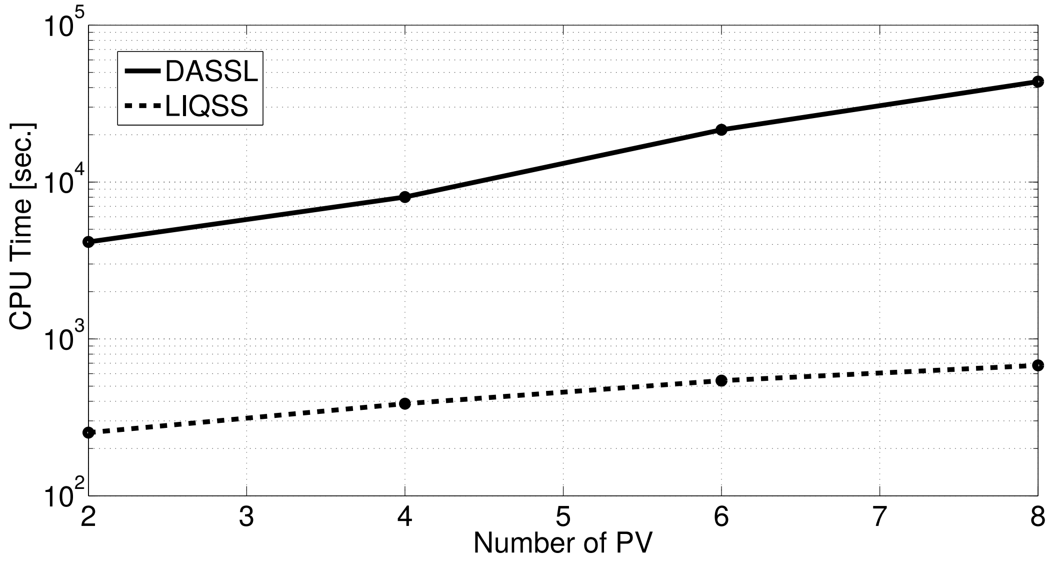

Figure 10: Number of PV Modules vs. CPU Time

5. Conclusions

A new Modelica library of realistic HRES components was developed, modeling energy generation and storage elements as well as power converters and different controllers. Using this library, a complete HRES model was built and simulated under different configurations.

Due to the realistic models of DC-DC converters that include switching circuit components, the resulting HRES models cannot be simulated in reasonable time by standard ODE solvers. The simulations with DASSL (using Dymola or OpenModelica), for instance, are more than 30 times slower than the real–time. Thus, simulating an hour of the system evolution takes more than an entire day.

To overcome this problem, we made use of a novel compiler that automatically translates these Modelica models into μ–Modelica language so that they can be simulated using the QSS Stand–Alone Solver using LIQSS methods. Following this approach, the LIQSS2 algorithm simulates the systems reaching almost real–time performance, improving the classic solvers speed in more than one order of magnitude on realistic stiff cases.

In order to study the behavior of the solvers when the system becomes larger and more complex, we modified the models adding more photovoltaic modules. The study showed that DASSL computational load grows almost cubically with the size of the problem while LIQSS2 exhibits a linear growth. That way, with 8 PV modules LIQSS2 is about 65 times faster than DASSL.

The efficient treatment of discontinuities and the explicit resolution of stiffness are the main reasons that explain the advantages of LIQSS2 over classic ODE solvers.

Taking into account these remarks, the conclusions of this work can be summarized as follows:

- The Modelica language allows to easily model complex and realistic HRES systems.

- Quantized State System solver exhibit noticeable advantages over classic solvers to simulate these systems.

- The combined usage of the ModelicaCC compiler and the QSS Stand Alone Solver allows to simulate these realistic HRES models in reasonable times

Regarding future work, one of the goals is to extend the library including more components (wind generators, three-phase inverters, more sophisticated controllers etc.) with even more realistic models (including parasitic elements, current paths, more realistic models of transistors and diodes, etc.) comparing again the performance of the different solvers in their presence.

We are also working on the development of a hybrid power converters modeling approach, that uses realistic switching models during transients and time–averaged model during steady state. With those models, we expect to combine the accuracy of realistic models with the simulation speed of time–averaged models.

Software Tools

- The models used in this article can be downloaded from `http://www.fceia.unr.edu.ar/~kofman/files/hres.mo`.
- The QSS Solver is an open source project available at `https://sourceforge.net/projects/qssengine/`.
- The ModelicaCC compiler is another open source project available at `https://sourceforge.net/projects/modelicacc/`

[1] P. García, J. P. Torreglosa, L. M. Fernández, F. Jurado, Optimal energy management system for stand-alone wind turbine/photovoltaic/hydrogen/battery hybrid system with supervisory control based on fuzzy logic, International Journal of Hydrogen Energy 38 (33) (2013) 14146–14158.

[2] L. Valverde, F. Rosa, A. del Real, A. Arce, C. Bordons, Modeling, simulation and experimental set-up of a renewable hydrogen-based domestic microgrid, International Journal of Hydrogen Energy 38 (27) (2013) 11672–11684.

[3] G. Gahleitner, Hydrogen from renewable electricity: An international review of power-to-gas pilot plants for stationary applications, International Journal of Hydrogen Energy 38 (5) (2013) 2039–2061.

[4] F. E. Cellier, E. Kofman, Continuous System Simulation, Springer-Verlag, New York, 2006.

[5] S. Cuk, R. Middlebrook, A general unified approach to modelling switching DC-to-DC converters in discontinuous conduction mode, in: Power Electronics Specialists Conference, IEEE, 1977, pp. 36–57.

[6] E. Van Dijk, H. J. Spruijt, D. M. O'Sullivan, J. B. Klaassens, PWM-switch modeling of DC-DC converters, IEEE Transactions on Power Electronics 10 (6) (1995) 659–665.

[7] M. Castañeda, A. Cano, F. Jurado, H. Sánchez, L. M. Fernández, Sizing optimization, dynamic modeling and energy management strategies of a stand-alone pv/hydrogen/battery-based hybrid system, International Journal of Hydrogen Energy 38 (10) (2013) 3830–3845.

[8] J.-K. Kuo, C.-F. Wang, An integrated simulation model for pem fuel cell power systems with a buck dc–dc converter, International Journal of Hydrogen Energy 36 (18) (2011) 11846–11855.

[9] G. Migoni, M. Bortolotto, E. Kofman, F. E. Cellier, Linearly implicit quantization-based integration methods for stiff ordinary differential equations , Simulation Modelling Practice and Theory 35 (2013) 118 – 136.

[10] G. Migoni, F. Bergero, E. Kofman, J. Fernndez, Quantization-Based Simulation of Switched Mode Power Supplies., Simulation: Transactions of the Society for Modeling and Simulation International 91 (4) (2015) 320–336.

[11] X. Floros, F. Bergero, N. Ceriani, F. Casella, E. Kofman, F. E. Cellier, Simulation of Smart-Grid Models using Quantization-Based Integration Methods, in: 10th International Modelica Conference, 2014.

[12] J. Fernández, E. Kofman, A stand-alone quantized state system solver for continuous system simulation, Simulation: Transactions of the Society for Modeling and Simulation International 90 (7) (2014) 782–799.

[13] E. C. Federico Bergero, Mariano Botta, E. Kofman, Efficient Compilation of Large Scale Modelica Models, in: 11th International Modelica Conference, 2015.

[14] P. Fritzson, V. Engelson, Modelica - A unified object-oriented language for system modeling and simulation, in: ECOOP'98 - Object-Oriented Programming, Springer, 1998, pp. 67–90.

[15] D. Brück, H. Elmqvist, S. E. Mattsson, H. Olsson, Dymola for multi-engineering modeling and simulation, in: 2nd International Modelica Conference, 2002.

[16] S. Baggi, D. Rivola, D. Strepparava, R. Rudel, A Modelica Library for Simulation of Electrical Energy Storage Coupled with Photovoltaic Systems, 12. Nationale Photovoltaik Tagung (2014) 10–11.

[17] B. Verbruggen, J. Van Roy, R. De Coninck, R. Baetens, L. Helsen, J. Driesen, Object-oriented electrical grid and photovoltaic system modelling in Modelica, in: 8th International Modelica Conference, 2011, pp. 730–738.

[18] M. A. Rubio, A. Urquia, L. González, D. Guinea, S. Dormido, FuelCell Lib - A Modelica library for modeling of fuel cells, in: 4th International Modelica Conference, 2005, pp. 75–82.

[19] E. Rothuizen, W. Mérida, M. Rokni, M. Wistoft-Ibsen, Optimization of hydrogen vehicle refueling via dynamic simulation, International Journal of Hydrogen Energy 38 (11) (2013) 4221–4231.

[20] N. Noguer, D. Candusso, R. Kouta, F. Harel, W. Charon, G. Coquery, A PEMFC multi-physical model to evaluate the consequences of parameter uncertainty on the fuel cell performance, International Journal of Hydrogen Energy 40 (10) (2015) 3968–3980.

[21] P. Fritzson, P. Aronsson, H. Lundvall, K. Nystrom, A. Pop, L. Saldamli, D. Broman, The OpenModelica Modeling, Simulation, and Development Environment., in: 46th Conference on Simulation and Modeling (SIMS'05), 2005, pp. 83–90.

[22] J. Åkesson, M. Gäfvert, H. Tummescheit, JModelica – an Open Source Platform for Optimization of Modelica Models, in: 6th Vienna International Conference on Mathematical Modelling, 2009.

[23] J. Dormand, P. Prince, A family of embedded Runge-Kutta formula, Journal of Computational and Applied Mathematics 6 (1) (1980) 19 – 26.

[24] L. R. Petzold, A description of DASSL - A differential/algebraic system solver, Scientific computing 1 (1983) 65–68.

[25] E. Kofman, S. Junco, Quantized State Systems. A DEVS Approach for Continuous System Simulation, Transactions of SCS 18 (3) (2001) 123–132.

[26] F. Bergero, X. Floros, J. Fernández, E. Kofman, F. E. Cellier, Simulating Modelica models with a Stand–Alone Quantized State Systems Solver, in: 9th International Modelica Conference, 2012.

[27] F. Valenciaga, P. Puleston, P. Battaiotto, Power control of a photovoltaic array in a hybrid electric generation system using sliding mode techniques, in: IEEE Proceedings on Control Theory and Applications, Vol. 148, 2001, pp. 448–455.

[28] H. L. Tsai, Insolation-oriented model of photovoltaic module using matlab/simulink, Solar energy 84 (7) (2010) 1318–1326.

[29] K. Hussein, I. Muta, T. Hoshino, M. Osakada, Maximum photovoltaic power tracking: an algorithm for rapidly changing atmospheric conditions, Generation, Transmission and Distribution, IEE Proceedings-142 (1) (1995) 59–64.

[30] J. Larminie, A. Dicks, M. S. McDonald, Fuel cell systems explained, Vol. 2, Wiley New York, 2003.

[31] R. Salim, M. Nabag, H. Noura, A. Fardoun, The parameter identification of the Nexa 1.2 kW PEMFC's model using particle swarm optimization, Renewable Energy 82 (2015) 26–34.

[32] O. Ulleberg, Stand-alone power systems for the future: optimal design, operation and control of solar-hydrogen energy systems, Ph.D. thesis, Norwegian Univ. Sci. Technol., Trondheim, Norway, ph.D. dissertation (1998).

[33] D. Ipsakis, S. Voutetakis, P. Seferlis, F. Stergiopoulos, C. Elmasides, Power management strategies for a stand-alone power system using renewable energy sources and hydrogen storage, International Journal of Hydrogen Energy 34 (16) (2009) 7081–7095.

[34] O. Tremblay, L.-A. Dessaint, Experimental validation of a battery dynamic model for EV applications, World Electric Vehicle Journal 3 (1) (2009) 1–10.

[35] K. Zhou, J. Ferreira, S. De Haan, Optimal energy management strategy and system sizing method for stand-alone photovoltaic-hydrogen systems, International journal of hydrogen energy 33 (2) (2008) 477–489.

[36] Ø. Ulleberg, The importance of control strategies in PV–hydrogen systems, Solar Energy 76 (1) (2004) 323–329.

List of Figures

List of Tables