Simulación eficiente de sistemas híbridos de generación de energía renovable

Gustavo Migoni[†‡],Pablo Rullo[†], Federico Bergero[†‡], Joaquín Fernández[†‡] y Ernesto Kofman[†‡]

[†] *CIFASIS-CONICET*
[‡] *FCEIA - UNR . Riobamba 245 bis - (2000) Rosario*

Resumen— **En este artículo se estudia la performance de los métodos de integración numérica por cuantificación de estados (QSS) aplicados a la simulación de sistemas híbridos de generación de energía basados en fuentes renovables.**

Los modelos realistas de este tipo de sistemas presentan discontinuidades muy frecuentes, son rígidos y poseen características no lineales, por lo que su simulación mediante algoritmos clásicos de integración puede resultar muy ineficiente. En contrapartida, existen métodos de QSS linealmente implícitos (LIQSS) que permiten simular eficientemente modelos con estas características.

En este trabajo se realiza un estudio comparativo entre los métodos de LIQSS y DASSL (el método clásico más robusto y eficiente para este tipo de modelos), mostrando que los métodos de cuantificación mejoran los tiempos de simulación en más de un orden de magnitud trabajando sobre un modelo de generación híbrida de relativa complejidad. Además, se muestran las herramientas desarrolladas para la utilización de estos métodos numéricos en modelos desarrollados en el lenguaje estándar Modelica, que permite el modelado gráfico muy simple de esta clase de sistemas.

Palabras Clave— **Sistemas de Generación Híbrida, Simulación por Cuantificación de Estados**

1. Introducción

El desarrollo de fuentes de energías renovables es una área en constante crecimiento impulsada por la actual dependencia de los combustibles fósiles con sus problemas socio-ambientales asociados.

Un sistema híbrido de generación de energía (*Hybrid Renewable Energy Systems* - HRES) convencional está constituido por arreglos de paneles fotovoltaicos y/o generadores eólicos que alimentan cargas DC, AC o mixtas. La energía entregada por las fuentes renovables dependen fundamentalmente de condiciones ambientales por lo que se hace necesario un sistema almacenador de energía que funcione como atenuador de estas variaciones, para lo cual se utilizan generalmente baterías.

El funcionamiento eficiente de estos sistemas depende fuertemente del correcto diseño de diversos sistemas de control, que actuan principalmente sobre las fuentes conmutadas que conectan los distintos elementos.

Debido a la complejidad de los modelos matemáticos resultantes, es imprescindible recurrir a la simulación numérica tanto para dimensionar los distintos componentes como para posteriormente diseñar y ajustar los sistemas de control mencionados.

La simulación de este tipo de sistemas suele ser muy problemática para los métodos de integración numérica debido principalmente a la presencia de elementos de conmutación operando en alta frecuencia. En estos casos, los algoritmos deben realizar una gran cantidad de cálculos en cada paso para determinar el instante exacto de ocurrencia de la discontinuidad [1], luego de la cual deben reinicializar la simulación.

Además, al representar de manera realista los elementos de conmutación (diodos y transistores) resultan modelos rígidos que requieren el uso de algoritmos implícitos que realizan iteraciones e inversiones matriciales costosas. En consecuencia, simular un segundo de evolución de un sistema de este tipo puede demandar varios minutos (e incluso horas) aún en una computadora muy poderosa.

Por este motivo se suele recurrir al uso de modelos promediados para los convertidores DC-DC en la gran mayoría de los modelos de HRES. Esta simplificación si bien es adecuada para muchos casos de estudio, es acotada en cuanto a condiciones de operación de los mismos y enmascara muchos fenómenos de importancia presentes en los convertidores reales. Entre estos fenómenos podemos citar la presencia de fallas en los elementos de conmutación, el contenido armónico introducido por los convertidores, la conducción discontinua durante transitorios, entre otros.

Actualmente hay disponible una familia de algoritmos para simular ecuaciones diferenciales ordinarias (ODE) denominados QSS (Quantized State System) que reemplazan la discretización temporal por la cuantificación de las variables de estado, cuyas características los tornan potencialmente muy convenientes para la simulación de HRES. Entre otras cosas, estos métodos son muy eficientes para tratar discontinuidades y permiten integrar ciertos sistemas rígidos sin realizar iteraciones.

De hecho, se ha mostrado que estos métodos tienen grandes ventajas en la simulación de distintas topologías de convertidores DC-DC bajo hipótesis realistas [7].

Una limitación de los métodos de QSS era que su implementación requería del uso de herramientas específicas de software que no eran amigables para describir modelos complejos. Sin embargo, recientemente se desa-

rrolló un simulador autónomo de QSS [3] que puede simular modelos descriptos en el lenguaje Modelica previamente procesados por un compilador desarrollado a tal fin [2]. Dichos modelos pueden desarrollarse en cualquier interfaz gráfica de Modelica simplemente arrastrando y conectando componentes circuitales, mecánicos, etc.

El objetivo de este trabajo es analizar la performance de los métodos LIQSS para simular HRES, para lo cual se realizaron modelos realista de un HRES mediante una interfaz gráfica del lenguaje Modelica, programando algunos sub-modelos que no se encontraban en la libreria estándar (panel solar, batería y algoritmos de control). Luego se utilizó el compilador antes mencionado para convertir los modelos completos en código $\mu-$Modelica y poder utilizar el simulador autónomo QSS. Finalmente se realizó un análisis del costo computacional y del error de simulación obtenido utilizando LIQSS y DASSL, analizando también el incremento del costo computacional en función de la cantidad de paneles.

El artículo está organizado de la siguiente manera: La Sección 2 introduce los conceptos básicos sobre las herramientas de modelado y simulación utilizadas en el resto del artículo. La Sección 3 presenta un esquema típico de HRES, sus componentes principales e interconección y su representación en Modelica. Finalmente, las secciones 4 y 5 presentan los resultados de simulación y conclusiones.

2. Herramientas de Modelado y Simulación

Esta sección presenta la metodología y el lenguaje de modelado utilizado junto con los métodos de cuantificación de estados QSS. Finalmente se describe la herramienta utilizada para la simulación.

2.1. Modelica

Modelica [5] es un lenguaje estándar orientado a objetos para el modelado de sistemas físicos. En él se pueden describir modelos multi-dominio (eléctricos, hidráulicos, mecánicos, etc) a través de ecuaciones acausales.

Aunque el lenguaje es textual, existen diversas herramientas que permiten desarrollar el modelo gráficamente (como OpenModelica, Dyomla, etc.) liberando así al modelador de tener que escribir código Modelica.

Existe además un extenso repositorio de modelos desarrollados por la comunidad, la Librería Estándar Modelica (MSL), que incluye componentes de diversas áreas, a partir de los cuales se pueden desarrollar nuevos modelos.

Mediante el uso de la MSL (y de otras liberías abiertas y comerciales de modelos existentes) y los editores gráficos mencionados, la tarea de modelado se puede reducir a arrastrar, parametrizar y conectar componentes de circuitos, sistemas mecánicos, etc.

Por ejemplo, la Figura 1 muestra el modelo de un convertidor Boost bidireccional en un editor gráfico de Modelica, armado totalmente con elementos de la MSL.

Los modelos Modelica describen sistemas de ecuaciones diferenciales algebraicas (DAE) que son convertidas en ODEs para su simulación. Dicha conversión es realizada por los compiladores de Modelica, entre los cuales encontramos también OpenModelica, Dymola, Wolfram SystemModeler, Jmodelica, entre otras.

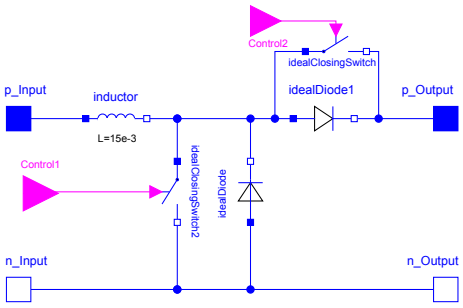

Figura 1: Boost Bidireccional en Modelica

Luego, este sistema ODE puede ser integrado con métodos de integración numérica clásicos de discretización temporal (como Euler, Runge-Kutta, DASSL) o como veremos a continuación, pueden utilizarse métodos de cuantificación de estados.

2.2. Métodos de QSS

Un sistema de tiempo continuo se puede ser descripto mediante un conjunto de ODEs:

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

donde $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ representa el vector de estados.

El modelo matemático de la Ec.(1) se puede simular usando métodos de integración numérica clásicos (Euler, Runge–Kutta, etc.) o alternativamente los algoritmos de QSS [1], que resuelven analíticamente una aproximación de esta ODE que resulta de reemplazar el vector de estados $\mathbf{x}(t)$ por una versión cuantificada $\mathbf{q}(t)$ del mismo:

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

Aquí, cada componente $q_j(t)$ del vector $\mathbf{q}(t)$ está relacionada con la componente $x_j(t)$ del vector de estado mediante una de cuantificación con histéresis, de manera que sólo hay cambios en $q_j(t)$ cuando esta difiere de $x_j(t)$ en una cantidad ΔQ_j denominada quantum.

Actualmente hay disponibles métodos QSS de primer, segundo y tercer orden denominados QSS1, QSS2 y QSS3. Estos métodos comparten la misma definición de la Ec.(2) y sólo difieren en que las variables cuantificadas $q_j(t)$ siguen trayectorias seccionalmente constantes, lineales o parabólicas en cada caso.

Los métodos de QSS son muy eficientes para detectar discontinuidades gracias a la forma de las trayectorias de las variables cuantificadas. Además, al ocurrir una discontinuidad no se requiere reinicializar la simulación sino sólo realizar un paso adicional cuyo costo es idéntico al un paso normal del algoritmo. Por este motivo, los métodos QSS son muy eficientes simulando sistemas con muchas discontinuidades.

En cuanto a los sistemas rígidos (con presencia de dinámicas rápidas y lentas simultáneas) hay una subfami-

lia de métodos linealmente implícitos de QSS denominados LIQSS, que combinan la idea de los algoritmos QSS con la de los métodos clásicos linealmente implícitos.

La idea básica de los LIQSS y de los métodos implícitos es evaluar la derivada de los estados en instantes futuros del tiempo. En los métodos clásico esto requiere iteraciones y/o inversiones de matrices para resolver ecuaciones implícitas. En el caso de los LIQSS, esto no es necesario ya que el valor futuro de las variables cuantificadas es conocido ($q_j(t) = x_j(t) \pm \Delta q_j$). En consecuencia, la implementación de los LIQSS es explícita.

Tal como en QSS, existen algoritmos LIQSS que realizan aproximaciones de primer, segundo y tercer orden. Estos algoritmos, si bien son explícitos, permiten simular eficientemente muchos sistemas rígidos.

2.3. Simulador de QSS autónomo

La implementación de los métodos de QSS es algo más compleja que la de los métodos clásicos, ya que en QSS se utiliza información estructural de las ecuaciones. Esto se debe a que cada variable de estado tiene su propio tiempo de actualización (son métodos asincrónicos) y por razones de eficiencia cada vez que cambia una variable cuantificada $q_j(t)$ no se evalúa todo el lado derecho de la Ec.(2) sino sólo las componentes de $\mathbf{f}$ que dependen explícitamente de la variable $q_j(t)$.

Hace un tiempo nuestro grupo desarrolló un simulador de QSS autónomo (Stand Alone QSS Solver) [3] que contiene una implementación muy eficiente de toda la familia de métodos de QSS. Dicha herramienta además extrae automáticamente la información estructural necesaria, de manera que el usuario sólo debe brindar el modelo a simular.

El QSS solver utiliza como lenguaje de entrada para sus modelos un subconjunto de Modelica denominado μ-Modelica. Este sub lenguaje restringe la sintaxis de Modelica al mínimo necesario para describir una ODE híbrida (parte continua y discontinua).

La herramienta convierte esta descripción en código C que es luego compilada junto con el método de integración numérica elegido.

El QSS-Solver implementa, además de los métodos de QSS y LIQSS, dos métodos clásicos de discretización temporal: DASSL [8] y DOPRI. Ambos métodos clásicos están implementados sobre los códigos más eficientes disponibles, e incluyen manejo eficiente de discontinuidades.

2.4. Compilador ModelicaCC

La transformación de un modelo Modelica a uno descripto en μ-Modelica (tal cual lo requiere el QSS Solver) requiere varios pasos: quitar la estrucutra jerárquica de clases, convertir la DAE en un sistema ODE, reemplazar las estrucutras no soportadas en μ-Modelica por estructuras equivalentes, etc.

Para realizar esto de manera automática se desarrolló recientemente un compilador de Modelica denominado ModelicaCC [2]. Mediante el uso de este compilador entonces, un modelo desarrollado usando cualquier editor gráfico de Modelica puede convertirse automáticamente en uno que puede simularse con el QSS-Solver.

3. Modelo del Sistema Híbrido de Generación

En esta sección se presenta el sistema híbrido de generación eléctrica a estudiar, describiendo su esquema general, la topología eléctrica y los modelos dinámicos de sus componentes.

En cada caso, se describe también su implementación en el lenguaje Modelica.

3.1. Esquema General

En la Fig.2 puede verse un esquema del HRES completo, compuesto por arreglos de PF como fuentes principals de generación y un banco de baterías como almacenador de energía. Estos subsistemas se conectan a través de convertidores DC-DC a un bus de continua, lo que permite la correcta adaptación de las tensiones. La carga es alimentada directamente desde el bus. La tensión en el bus es controlada por la batería mediante un lazo PI que intercambia energía con el bus mediante un convertidor DC-DC tipo boost bidireccional.

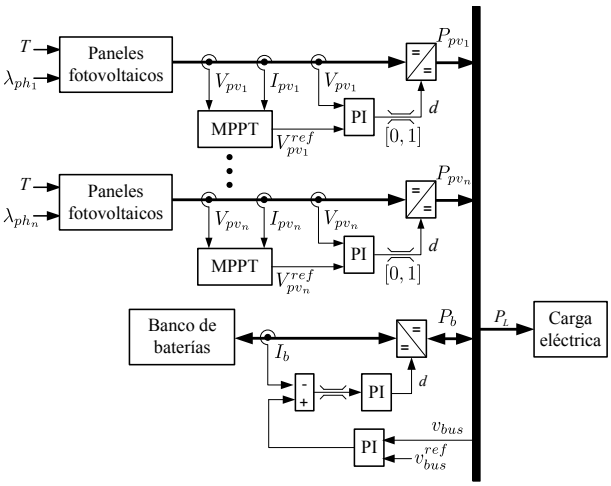

Figura 2: Sistema híbrido de generación de energía

3.2. Convertidores DC-DC

Los convertidores de potencia permiten adecuar los niveles de tensión de entrada y salida de los distintos componentes del sistema mediante elementos que conmutan en alta frecuencia variando su ciclo de trabajo.

Existen múltiples topologías de convertidores, entre las que encontramos el *buck* o reductor, *boost* o elevador y *buck-boost* o reductor-elevador. Aquí se utilizarán convertidores *boost* unidireccionales para los arreglos de paneles fotovoltaicos, dado que el flujo de energía se da sólo desde la fuente hacia el bus. Para el banco de baterías en cambio, es necesario un convertidor bidireccional, ya que el flujo de energía puede cambiar de dirección.

La implementación en Modelica con elementos de la libraría estándar de este último modelo puede verse en la Fig.1.

3.3. Paneles fotovoltaicos

Un panel fotovoltaico (PF) está formado por un conjunto de celdas fotovoltaicas que se conectan en serie para alcanzar tensiones de trabajo adecuadas. A su vez los paneles pueden conectarse en arreglos serie-paralelo de acuerdo a la tensión y potencia necesaria.

Aquí consideramos un modelo descipto en [4], donde la ecuación característica corriente-voltaje resulta

$$I_{pv} = I_{ph} - I_{rs}\left(e^{\frac{q(V_{pv}+I_{pv}R_s)}{kA_cT}} - 1\right), \qquad (3)$$

donde I_{ph} es la corriente generada

$$I_{ph} = \left[I_{sc} + K_1\left(T - T_{ref}\right)\right]\lambda_{ph} \setminus 100, \qquad (4)$$

e I_{rs} la corriente de saturación inversa

$$I_{rs} = I_{or}\left(\frac{T}{T_{ref}}\right)^3 e^{\frac{qEg(1/T_{ref}-1/T)}{KA_c}}, \qquad (5)$$

En estas ecuaciones, V_{pv} es la tensión en bornes de la celda, T es la temperatura de la celda,λ_{ph} es la radiación medida en $mW\ cm^{-2}$ y los restantes son parámetros físicos.

En el caso de un PF o un arreglo de PF, el modelo se extiende de la Ec. (3) como

$$I_{pv} = N_pI_{ph} - N_pI_{rs}\left(e^{\frac{q(V_{pv}/N_s+I_{pv}R_s/N_p)}{kA_cT}} - 1\right), \qquad (6)$$

donde N_s es el numero de celdas en serie por rama, y N_p el numero de ramas dispuestas en paralelo.

Para este modelo, no se utilizó la librería estándar de Modelica sino que se construyó directamente el componente mediante el siguiente código Modelica.

```
model Panel
  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; //Corriente por radiacion
  Real exponente;
  Real Ipv;
  Real Vpv;
  Real lambdaph;
  Modelica.Blocks.Interfaces.RealInput u

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

Allí vemos que el modelo del panel presenta una ecuación no lineal (debido el exponente) implícita. Esto generará un lazo alebraico en la ODE.

3.4. Control MPPT

En los paneles fotovoltaicos la potencia generada depende tanto de la radiación solar y la temperatura, como de la tensión en bornes que se impone. De esta manera existe un punto para cada valor de radiación y temperatura donde la potencia resulta máxima. Existen algoritmos capaces de hacer un seguimiento de estos puntos de máxima potencia (*Maximum Power Point Tracking*, MPPT). En particular para este trabajo se utiliza el método *Inc-Cond* [6]. Cada arreglo de PF se conecta al bus a través de un convertidor boost DC-DC que permite implementar el algoritmo de MPPT. Este esquema (Arreglo de PF - Convertidor DC-DC con MPPT) puede repetirse en el caso de requerir mayores niveles de generación.

Este algoritmo se implementó en un modelo de Modelica.

3.5. Batería

La batería es modelada por una fuente de tensión controlada y una resistencia en serie [9]. La tensión a circuito abierto de esta fuente es calculada a través de una ecuación no lineal dependiente del estado de carga actual ($\int i_b dt$):

$$E_b = E_0 - K\frac{Q}{Q - \int i_b dt} + Ae^{(-B\int i_b dt)}, \qquad (7)$$

donde E es la tensión a circuito abierto y los restantes son parámetros físicos del modelo.

3.6. Modelo Completo HRES

En la figura 3 puede verse el modelo completo del sistema de generación utilizado en este trabajo. El mismo esta compuesto por cuatro módulos de generación fotovoltáica (Boost_Panel) conectados en paralelo, un banco de baterías (BatteryAndBuckBoost) y una carga.

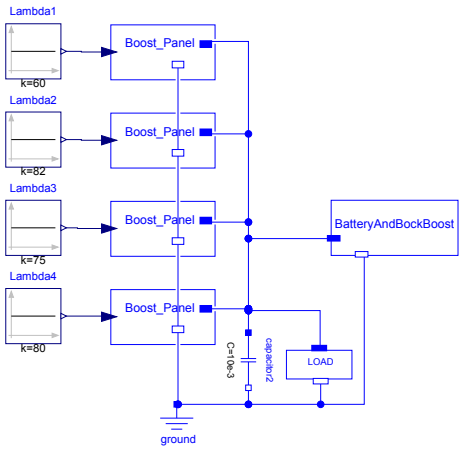

Figura 3: Diagrama en Modelica del sistema completo

Cada subsistema Boost-Panel, BatteryAndBuckBoost y Load de la figura 3 agrupan los distintos componentes eléctricos que intervienen en el sistema. Un detalle del subsistema BoostPanel puede verse en la figura 4,

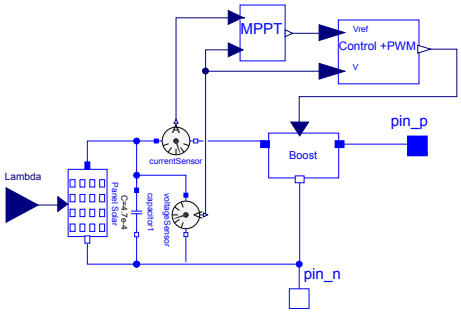

Figura 4: Subsistema Boost-Panel en Modelica

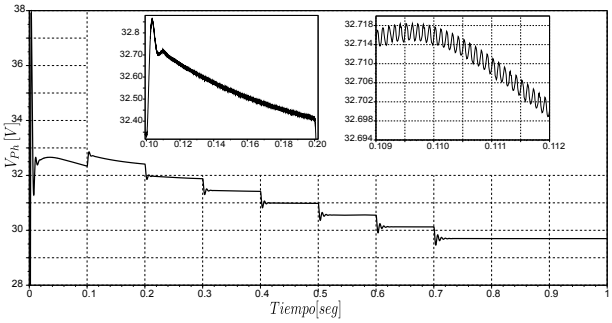

Figura 5: Tensión en el panel fotovoltaico

4. Resultados

En esta sección se muestran los resultados de simulación, comparando la performance obtenida al simular el modelo de generación con los métodos LIQSS y con el método clásico DASSL. Las comparaciones se realizaron tomando siempre un tiempo final de simulación de 1 segundo (en el que se alcanza el estado de régimen permanente) y con diferentes requerimientos de precisión.

Para las simulaciones se utilizó una PC con un SO Ubuntu y un procesador Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz. Para ambos métodos, se utilizó la herramienta de simulación *QSS-Solver* de modo que ambos métodos simularán exactamente el mismo modelo. Esta herramienta implementa el código DASSRT para DASSL (que trata eficientemente las discontinuidades).

Se probaron también las implementaciones de DASSL de Dymola y OpenModelica, pero los tiempos obtenidos eran mayores que los del *QSS-Solver* por lo que sólo se informan los resultados obtenidos por esta última herramienta.

En cuanto a los parámetros de comparación, se utilizaron el tiempo de CPU requerido por la simulación, el número de evaluación de funciones escalares y el error relativo cometido, siendo este último calculado según:

$$e_{rr} = \sqrt{\frac{\sum (i_{L_{bat}}(k) - \hat{i}_{L_{bat}}(k)}{\hat{i}^2_{L_{bat}}(k)}} \qquad (8)$$

donde $i_{L_{bat}}(k)$ es una referencia de la corriente en la inductancia del convertidor DC-DC de la batería generada utilizando LIQSS con una precisión de 10^{-9}. Para la simulación del sistema, se utilizó una frecuencia de conmutación de 10KHz.

La figura 5 muestra la evolución de tensión a la salida del panel solar. Esta tensión cambia su valor medio cada 0.1 segundos, período al cual se actualiza la tensión de referencia del algoritmo MPPT y, como era de esperar, presenta un pequeño ripple debido a la frecuencia de conmutación del convertidor DC-DC.

En la figura 6 puede verse la trayectoria de la corriente $i_L(t)$ en la bobina de uno de los convertidores DC-DC que interconecta un panel fotovoltaico con el bus de continua. En la misma se ve claramente que el convertidor opera durante un tiempo en conducción discontinua durante el transitorio. Este fenómeno, junto con el mayor grado de detalle observado en las señales, justifica la utilización de modelos realistas ya que obtener resultados de simulación con modelos que operan tanto en conducción continua como discontinua con modelos promediados es bastante complicado y provee mucha menos información en su salida.

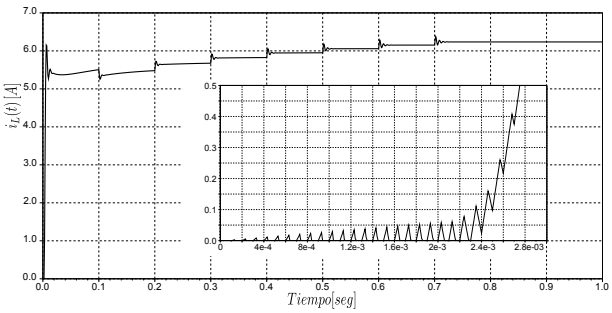

Figura 6: Corriente en la bobina de un convertidor Boost

En el cuadro 1 se puede ver la comparación de los tiempos de CPU, la cantidad de evaluaciones de funciones escalares y los errores obtenidos con los diferentes métodos.

Método de Integración		Error Relativo	Eval. Jacobiano	Eval. de Fun. f_i	CPU [mseg]
DASSL	err.tol=10^{-3}	$3{,}0 \cdot 10^{-3}$	155811	63744050	90523
	err.tol.=10^{-5}	$9{,}8 \cdot 10^{-4}$	578724	232353242	318576
LIQSS2	err.tol=10^{-3}	$3{,}6 \cdot 10^{-3}$	–	4802696	4154
	err.tol=10^{-5}	$7{,}4 \cdot 10^{-5}$	–	8419172	8706

Cuadro 1: Comparación de resultados de simulación con los distintos métodos.

En la misma puede verse que todos los métodos cumplen con el requisito de precisión excepto DASSL con una precisión de 10^{-5}. Se ve que aun siendo los métodos LIQSS de menor orden que DASSL, LIQSS2 resulta 22 veces mas rápido que DASSL al seleccionar una precisión de 10^{-3} y 36 veces mas rápido con una precisión de 10^{-5}. Observando la relación de cantidad de evaluación de funciones, vemos que resulta ser 13 y 28 veces

mayor en DASSL que en LIQSS con precisiones de 10^{-3} y 10^{-3} respectivamente, lo que se corresponde con los tiempos de CPU medidos, teniendo en cuenta además que DASSL evalúa e invierte frecuentemente el Jacobiano del sistema.

Además de analizar el costo computacional y error para este modelo particular con 4 paneles, se analizo la influencia de incrementar el orden del sistema y el número de discontinuidades sobre el tiempo de CPU requerido por la simulación. Para realizar este análisis, se simulo el mismo sistema incrementando el número de paneles N_p entre 1 y 8, siendo el orden del modelo $n = 3 \cdot N_p + 7$. La figura 7 muestra el tiempo de CPU en función del numero de paneles N_p.

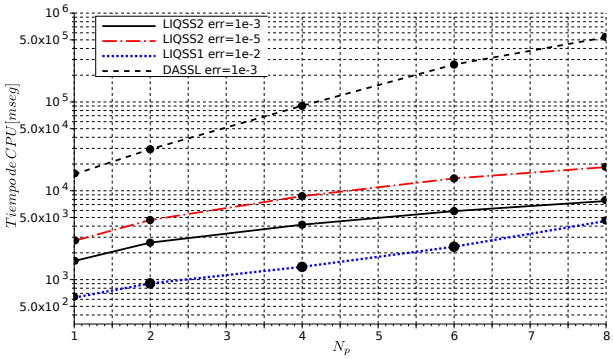

Figura 7: Tiempo de simulación en función de la cantidad de paneles

Como puede verse en la misma, el costo computacional en todos los métodos QSS crece linealmente con el número de paneles (proporcional al orden del sistema) mientras que en DASSL este crecimiento es aproximadamente cúbico, ya que debe invertirse una matriz Jacobiana que crece con el número de paneles. En consecuencia, con 8 paneles LIQSS2 es 70 veces más rápido que DASSL para la precisión de 10^{-3}.

5. Conclusiones

En este artículo se analizo el desempeño de los métodos LIQSS en la simulación de una HRES realizando comparaciones con los resultados obtenidos con DASSL.

A partir de este análisis se mostró que el uso de LIQSS2 mejora en más de un orden de magnitud los tiempos de simulación respecto del uso de DASSL. Más aún, al aumentar el tamaño del modelo, se observó que esta mejora es aún más notoria. Por otro lado, se pudo ver que en todos los casos los LIQSS respetan los requisitos de precisión.

Estas ventajas se pueden explicar a partir de la manera eficiente en que los métodos de QSS tratan las discontinuidades y a partir de las características explícitas de los métodos de LIQSS para simular sistemas rígidos y para tratar sistemas ralos.

En cuanto al proceso de modelado, utilizar conjuntamente Modelica y el Solver QSS permite realizar este tipo de modelos de manera sencilla, simplemente arrastrando e interconectando elementos básicos de librería. De este modo el usuario no familiarizado con programación ni métodos numéricos puede realizar este tipo modelos en un entorno amigable y obtener resultados de simulación eficientemente.

En cuanto a trabajo futuro, el objetivo es extender este estudio comparativo analizando lo que ocurre al agregar mayor realismo a la parte circuital (incluyendo elementos parásitos, circuitos de disparo, modelos más realistas de transistores y diodos, etc.) y al agregar más componentes al sistema de generación, tanto a nivel de algoritmos de control, como de elementos de generación, acondicionamiento y almacenamiento (generadores eólicos, inversores trifásicos, electrolizadores, pilas de combustible, etc.).

Referencias

[1] François E. Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer-Verlag, New York, 2006.

[2] Esteban Campostrini Federico Bergero, Mariano Botta and Ernesto Kofman. Efficient compilation of large scale modelica models. In *11th International Modelica Conference - Versaille, France*, 2015.

[3] Joaquín Fernández and Ernesto Kofman. A Stand-Alone Quantized State System Solver for Continuous System Simulation. *Simulation: Transactions of the Society for Modeling and Simulation International*, 90(7):782–799, 2014.

[4] Diego Feroldi, Pablo Rullo, and David Zumoffen. Energy management strategy based on receding horizon for a power hybrid system. *Renewable Energy*, 75:550–559, 2015.

[5] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-Interscience, New York, 2004.

[6] KH Hussein, I Muta, T Hoshino, and M1 Osakada. Maximum photovoltaic power tracking: an algorithm for rapidly changing atmospheric conditions. *IEE Proceedings-Generation, Transmission and Distribution*, 142(1):59–64, 1995.

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

[8] Linda R Petzold et al. A description of dassl: A differential/algebraic system solver. In *Proc. IMACS World Congress*, pages 430–432, 1982.

[9] Olivier Tremblay and Louis-A Dessaint. Experimental validation of a battery dynamic model for ev applications. *World Electric Vehicle Journal*, 3(1):1–10, 2009.