Simulación de Inversores PWM Monofásicos en Modo Diferencial con métodos basados en cuantificación

Mario Bortolotto[1] and Gustavo Migoni[2]

[1] FCEIA-UNR, CIFASIS-CONICET, Rosario, Argentina
mariob@fceia.unr.edu.ar
[2] CIFASIS, CONICET, Rosario, Argentina
migoni@cifasis-conicet.gov.ar

Resumen Este artículo presenta un estudio comparativo sobre la demanda de tiempo de CPU en la simulación de inversores monofásicos en modo diferencial con modulación por ancho de pulso (DMSI PWM, por sus siglas en inglés), utilizando métodos de integración numérica basados en cuantificación de los estados (Quantized State System, QSS) y métodos clásicos de discretización temporal.
Dado que estos sistemas presentan discontinuidades frecuentes, los métodos numéricos tradicionales implican altos costos computacionales, tanto por la detección y tratamiento de eventos como por la necesidad de pasos de integración muy pequeños. En este contexto, se analiza cómo los métodos QSS, diseñados para tratar eficientemente ecuaciones diferenciales ordinarias (ODEs) con discontinuidades, permiten reducir significativamente el tiempo de simulación.
En particular, se estudia el desempeño de los métodos QSS linealmente implícitos (Linearly Implicit Quantized State System, LIQSS), desarrollados para simular sistemas rígidos y discontinuos, frente a integradores clásicos como DASSL y CVODE-BDF. Este análisis es especialmente relevante en aplicaciones donde los modelos de inversores se integran en sistemas más complejos, con tiempos de simulación mucho mayores que los períodos de conmutación.
Como contribución principal, se presentan nuevos modelos de DMSIs PWM desarrollados en μ–Modelica, evaluados en los entornos QSS Solver y OpenModelica, comparando rigurosamente el rendimiento de ambas estrategias de simulación.

Keywords: Inversores PWM, QSS/LIQSS, simulación

Simulation of Single-Phase PWM Inverters in Differential Mode using quantization-based methods

Abstract. This paper presents a comparative study of CPU time requirements for the simulation of differential-mode single-phase PWM inverters (DMSIs PWM), using Quantized State System (QSS) integration methods and classical time-discretization-based methods.

These devices frequently exhibit discontinuities, making simulations with traditional methods computationally expensive due to the overhead involved in event detection and processing, as well as the requirement for very small time steps. In this context, the paper analyzes how QSS methods can significantly reduce simulation times, as they are well-suited to handling discontinuous ordinary differential equations (ODEs).
In particular, the study focuses on the performance of linearly implicit QSS methods (LIQSS), which were developed for the efficient simulation of stiff and discontinuous systems, and compares them with well-known classical solvers such as DASSL and CVODE-BDF. This analysis is especially relevant when inverter models are embedded within larger systems, where the total simulation time greatly exceeds the switching periods of the converters.
One of the main contributions of this work is the development of new DMSI PWM models in μ–Modelica, which were evaluated using QSS Solver and OpenModelica to enable a rigorous comparison between both simulation approaches.

Keywords: PWM inverters, QSS/LIQSS, simulation

1. Introducción

La simulación eficiente de sistemas de electrónica de potencia resulta fundamental para el diseño y mejora de aplicaciones en redes eléctricas (Mohan, 2012). Particularmente, en los sistemas de generación fotovoltaicos y/o aerogeneradores los inversores PWM son componentes esenciales (Erickson y Maksimovic, 2020). En el caso de aplicaciones para sistemas de energía renovable de baja potencia es muy común utilizar DMSIs PWM con el objetivo de convertir la energía eléctrica de manera eficiente (Albakri et al., 2024).

Para obtener resultados de simulación precisos de los DMSIs PWM se debe utilizar modelos matemáticos definidos por ODEs discontinuas. Esto se debe a la presencia de elementos que conmutan a alta frecuencia (diodos y transistores). Su simulación mediante métodos clásicos (basados en discretización temporal) resulta ineficiente, dado que el tiempo de ejecución requerido en la computadora (medido en tiempo de CPU o de reloj de pared) es considerablemente alto en comparación con el tiempo virtual de simulación (Simões y Farret, 2016). Esto se debe a que los algoritmos numéricos deben realizar múltiples cálculos en cada paso para detectar el instante de las discontinuidades y reinicializar los métodos luego de ese instante para evitar integrar a través de una discontinuidad (Cellier y Kofman, 2006). Además si al modelar los inversores se considera las características realistas de los elementos de conmutación, los modelos presentan dinámicas lentas y rápidas simultáneamente (rígidos). Como consecuencia, se debe usar métodos implícitos que requieren iteraciones costosas e inversión de matrices.

Por estos motivos la simulación de tan solo unos minutos de modelos realistas puede demandar varias horas de CPU, incluso en equipos de alto rendimiento

y usando métodos eficientes como DASSL (Petzold, 1983) que es uno de los métodos implementado en la mayoría de los software comerciales de simulación.

Como alternativa a los métodos clásicos, existen los métodos QSS basados en la discretización de los estados. Estos métodos emplean una representación en el espacio de estados de las ODEs y reemplazan el enfoque convencional de división del tiempo, por una cuantificación de los estados. Esta transformación conduce a un modelo de simulación de eventos discretos asincrónico en lugar de un modelo de ecuaciones en diferencias de tiempo discreto (Zeigler et al., 2018). Esta característica hace que tengan grandes ventajas para tratar con diversos sistemas con discontinuidades frecuentes. Existen versiones de los métodos QSS de hasta tercer orden (Cellier et al., 2008) y métodos $LIQSS$ para tratar sistemas rígidos(Migoni et al., 2013).

Observando las características de los métodos QSS y de los modelos realistas de DMSIs PWM, en este trabajo se lleva a cabo un *análisis comparativo* de la simulación de estos inversores utilizando $LIQSS$ y métodos clásicos avanzados, como $DASSL$ y $CVODE - BDF$.

El artículo está organizado de la siguiente manera: la **Sección 2** introduce los conceptos básicos que se utilizan para el desarrollo del artículo. La **Sección 3** presenta los modelos construidos para la simulación de distintas topologías de *inversores DMSIs PWM*. Luego, en la **Sección 4** se comparan indicadores de desempeño entre los distintos métodos y herramientas utilizados. Finalmente, en la **Sección 5** se presentan las conclusiones del trabajo.

2. Antecedentes

En esta sección se presenta el marco conceptual necesario para comprender el desarrollo posterior de este artículo. Se incluye una breve descripción de los métodos de integración por cuantificación (MIC) y se desarrollan las nociones fundamentales de QSS y LIQSS. Además se muestran las características esenciales del Simulador Autónomo de QSS (Stand–Alone QSS Solver) donde fueron implementado los modelos construidos para este trabajo. Por último se presenta un resumen sintético sobre los modelos de DMSI PWM.

2.1. Métodos QSS

Los MIC reemplazan la discretización temporal por la cuantificación de los estados, permitiendo modelar sistemas continuos como sistemas de eventos discretos (Zeigler et al., 2018) mediante el formalismo DEVS (*Discrete EVent System Specification*, Kofman et al., 2001).

Con esta ida se desarrolló la primera familia de MIC denominados QSS. En los métodos QSS, cada variable de estado $x_j(t)$ se representa por una versión cuantificada $q_j(t)$ que se actualiza solo cuando el error $|q_j(t) - x_j(t)|$ supera un umbral. Por ejemplo, en QSS1 (Kofman y Junco, 2001) $q_j(t)$ es una aproximación seccionalmente constante de $x_j(t)$ (Figura 1 a)), mientras que QSS2 (Figura 1

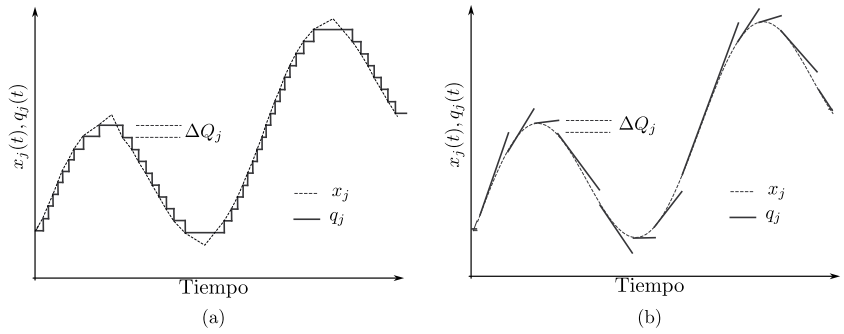

Figura 1. Funciones de cuantificación de orden cero (a) y primer orden (b).

b)) y QSS3 aplican aproximaciones de orden superior para mejorar la precisión sin aumentar significativamente el costo computacional.

Para comprender estos métodos, consideremos un sistema continuo definido por ODEs:

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

Los MIC lo aproximan reemplazando los estados $\mathbf{x}$ por versiones cuantificadas $\mathbf{q}$:

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

donde $|q_j(t) - x_j(t)| \leq \Delta Q_j$.

En el caso de QSS1, $q(t)$ es seccionalmente constante y puede escribirse como

$$q_j(t) = \begin{cases} x_j(t) & \text{si } |q_j(t^-) - x_j(t)| = \Delta Q_j \\ q_j(t^-) & \text{en caso contrario} \end{cases} \tag{3}$$

Una de las principales ventajas de los métodos QSS es su naturaleza asincrónica, que permite simular eficientemente sistemas con discontinuidades frecuentes (Cellier y Kofman, 2006). Sin embargo, los QSS presentan limitaciones para simular sistemas en los que coexisten dinámicas rápidas y lentas (sistemas rígidos) ya que aparecen oscilaciones de alta frecuencia que inducen altos costos computacionales. Para abordar esto, se desarrollaron los métodos $LIQSS$ (Migoni et al., 2013).

Los métodos de integración para sistemas rígidos requieren evaluar las ODEs en valores futuros de las variables de estado (SV). Para lograr esto, los métodos clásicos deben realizar iteraciones para saber el valor futuro de las SV. Los métodos $LIQSS$ aprovechan el hecho de que en los métodos QSS ya se conoce el valor futuro del estado cuantificado. Esto elimina la necesidad de iteraciones y permite implementaciones explicitas eficiente para sistemas rígidos, sin perder las ventajas de QSS para sistemas discontinuos.

Estos métodos están implementados en el Simulador Autónomo de QSS desarrollado por el grupo de investigación.

2.2. Stand–Alone QSS solver

La herramienta más eficiente y completa para simular con métodos QSS es QSS Solver (Fernández y Kofman, 2014). Los modelos se describen en μ-Modelica, un subconjunto de Modelica (Bergero et al., 2012), y se traducen automáticamente a código C posteriormente. Este código que incluye las ODEs, funciones de detección de eventos y controles de discontinuidades.

La herramienta extrae la información de la estructura (matrices de incidencia) y genera el código para la evaluación simbólica de la matriz Jacobiana. El código en C producido luego se vincula a los diferentes algoritmos QSS (QSS y $LIQSS$ de orden 1 a 3) o a métodos clásicos como DASSL, $CVODE$, etc.

Debido a que esta herramienta dispone de toda la información estructural y puede realizar evaluaciones simbólicas de las matrices Jacobianas, aun utilizando los métodos clásicos implementados se obtiene resultados de simulación significativamente más rápido que con otras interfaces. Además, el hecho de que se utilice el mismo fragmento de código para las ODEs en los diferentes algoritmos (QSS y métodos clásicos) permite realizar comparaciones de rendimiento justas entre ellos.

2.3. Inversores monofásicos en modo diferencial.

El principio de funcionamiento de un inversor monofásico en modo diferencial (DMSI) se basa en conectar dos convertidores conmutados bidireccionales de manera diferencial (Albakri et al., 2024) y utilizar señales sinusoidales desfasadas 180 grados como referencia de ciclo de trabajo d para cada una de ellas tal como se muestra en la Figura 2.

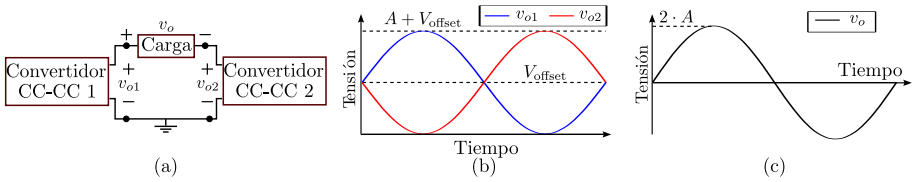

Figura 2. DMSI PWM: (a) esquema; (b) tensiones en los convertidores 1 y 2; (c) tensión de carga.

Al aplicar una señal sinusoidal en la referencia del ciclo de trabajo (d) de uno de los convertidores de la forma:

$$d_i = 0{,}5 + K_i \sin(\omega t + \phi_i) \quad (\text{con } K_i \leq 0{,}5) \tag{4}$$

se obtiene tensión v_0 de salida alterna con una componente de CC:

$$v_{o_i}(t) \simeq A \sin(\omega t + \phi_i) + V_{\text{offset}} = d_i V_{in} \tag{5}$$

Si se conectan dos de estas fuentes conmutadas como se muestra en la Figura 2 (a) y se utilizan ciclos de trabajo desfasados 180 grados ambas tensiones de salida (v_{o1} y v_{o2}) tendrán el mismo desfasaje ($\phi_1 = 0$ y $\phi_2 = \pi$) como puede verse en la Figura 2 (b)

Dado que en estas topologías la carga se conecta en modo diferencial entre la salida de ambos convertidores, se puede obtener una tensión de salida CA(Ecuación (6)) sin componente de CC como se puede observar en la Figura 2 (c).

$$v_o(t) \simeq 2A \sin(\omega t) \tag{6}$$

Si bien el análisis realizado permite comprender cómo funcionan cualitativamente los DMSIs PWM, para conocer con precisión el comportamiento de cada una de sus variables y poder controlarlas, es necesario recurrir a modelos.

3. Modelos

En esta sección se presentan los modelos de tres topologías de inversores DMSI PWM, utilizando una estrategia de modelado realista que contempla los elementos de conmutación para no perder detalle en los resultados.

En primer lugar se presentan los modelos de los elementos de conmutación; luego, se derivan las ecuaciones del circuito y, finalmente, se traducen a un modelo descrito en μ–Modelica.

3.1. Modelado de Elementos Discontinuos

El modelado de elementos discontinuos en inversores PWM requiere representar con precisión el comportamiento no lineal y su evolución temporal, generalmente mediante ODEs con discontinuidades.

Los diodos y transistores presentan características tensión-corriente (v–i) discontinuas. La Figura 3 (a) se muestra la característica real v–i de un diodo.

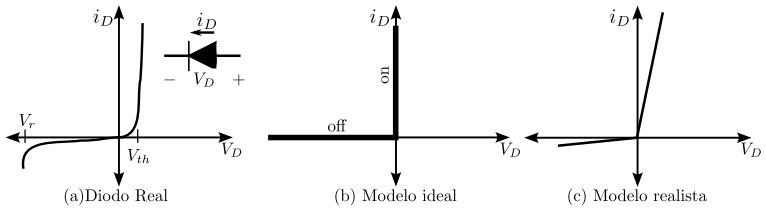

Figura 3. Curva v-i de: (a) diodo real; (b) modelo ideal; (c) modelo realista.

Existen dos enfoques para representar los componentes de conmutación. El enfoque *ideal* representa el componente como circuito abierto o cerrado, generando modelos con estructura variable que no son compatibles con simuladores

estándar. La Figura 3 (b) se ve la característica v–i del diodo obtenida con esta estrategia.

Otro enfoque más *realista* consiste en representar el componente de conmutación mediante una resistencia que cambia de valor: bajo (R_{On}) en conducción y alto (R_{Off}) en corte. En la Figura 3(c) se ve la característica v–i del diodo modelado con esta estrategia. Los modelos así obtenidos no presentan cambios estructurales sino paramétricos por lo que pueden ser simulados mediante herramientas estándar. Sin embargo, los modelos realistas así obtenidos suele presentar rigidez numérica (*stiffness*) por lo que se deben utilizar métodos numéricos implícitos con su consecuente alta carga computacional (Cellier y Kofman, 2006).

Para realizar el modelo, dado que el diodo comienza a conducir cuando $V_D > V_\gamma$ y deja de conducir cuando $i_D \leq 0$, la detección del estado de conducción se representa según:

$$diodeon = \begin{cases} 1 \text{ diodo conduce} \\ 0 \text{ diodo cortado} \end{cases} \tag{7}$$

A partir del estado de conducción *diodeon*, podemos definir la variable auxiliar $s = diodeon \cdot i_D + (1 - diodeon) \cdot V_D$, que representa tanto la tensión como la corriente, según el estado de conducción del diodo.

Finalmente, el comportamiento del diodo puede describirse en μ–Modelica como:

```
s = diodeon * iD + (1 - diodeon) * iD * Rd;
algorithm
  when s > 0 then Rd := ROn; diodeon := 1;
  elsewhen s < 0 then Rd := ROff; diodeon := 0;
  end when;
```

Este enfoque optimiza la simulación del diodo al alternar eficientemente entre sus dos estados.

De manera similar, se pueden modelar los transistores cuando operan en corte y saturación. En este caso, también pueden representarse mediante una resistencia de valor R_{On} cuando conducen, y R_{Off} cuando están en corte. A diferencia del diodo, este componente cuenta con una señal de control que comanda su estado de conducción.

A partir de estos modelos elementales de los componentes discontinuos, se pueden construir fácilmente modelos *realistas* de inversores.

3.2. Modelo del DMSI PWM buck

La Figura 4 muestra el esquema circuital básico del DMSI PWM *buck*. En esta configuración la tensión de salida es una señal de alterna $v_{out}(t)$ cuya amplitud esta acotada según $-U \leq v_{out}(t) \leq U$. Para esto, cuando la señal del PWM de la rama i esta en alto conduce el transistor T_{i1} y el T_{i_2} esta cortado y cuando la señal de PWM esta en bajo T_{i1} esta cortado y T_{i_2} conduce. Además, el control de los transistores debe garantizar que no conduzcan simultáneamente dos transistores de la misma rama (para evitar cortocircuitos). Esto se garantizar agregando un tiempo muerto T_{delay} entre el apagado y encendido de los transistores.

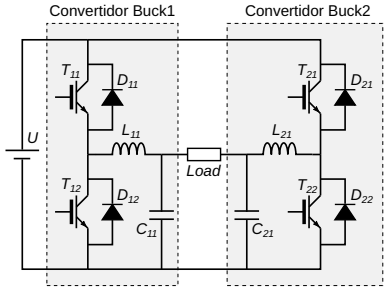

Figura 4. Esquema circuital del DMSI PWM buck.

A partir de este esquema, y teniendo en cuenta que el ij-ésimo transistor T_{ij} y el ij-ésimo diodo D_{ij} se modelan mediante las resistencias $R_{s_{ij}}$ y $R_{d_{ij}}$, respectivamente, se obtiene el siguiente modelo de orden 4 en ODEs para el DMSI PWM buck:

$$\frac{di_{L_{i1}}}{dt} = \frac{-i_{Di2}R_{di2} - u_{C_{i1}}}{L_{i1}} \quad \frac{du_{C_{i1}}}{dt} = \frac{i_{L_{i1}} - (u_{C_{i,1}} - u_{C_{3-i,1}})/R_{LOAD}}{C_{i1}} \tag{8}$$

En estas ecuaciones, el subíndice $i = 1, 2$ indica el convertidor (columna del inversor) correspondiente. Las variables $i_{L_{i1}}$ y $u_{C_{i1}}$ representan las corrientes en los inductores y las tensiones en los capacitores e $i_{D_{i2}}$ las corrientes en los diodos D_{i2}. Esta última se calcula según:

$$i_{D_{i2}} = \frac{(i_{L_{i1}}R_{s_{i1}} - U)R_{s_{i2}}}{R_{s_{i1}}R_{d_{i2}} + R_{s_{i1}}R_{s_{i2}} + R_{d_{i2}}R_{s_{i2}}} \tag{9}$$

La tensión de salida de este inversor se define como $dU = u_{C_{11}} - u_{C_{21}}$.

El siguiente código μ–Modelica es una implementación del modelo completo de la DMSI PWM buck. En el mismo ambos convertidores tienen referencias de ciclo de trabajo senoidales desfasadas 180 grados. Además, incorpora un tiempo muerto (*Delay*) antes de la transición del estado cortado a saturación de los transistores que se usa para evitar posibles cortocircuitos debido a que realmente el apagado de los transistores no ocurre en tiempo cero.

```
equation
  //buck1
    iD12=(iL11*Rs11-U)*Rs12/(Rs11*Rs12+Rs11*Rd12+Rs12*Rd12);
    s1=diodeon1*iD12+(1-diodeon1)*iD12*Rd12;
    der(iL11) = (-iD12*Rd12- uC11)/L11;
    der(uC11) = (iL11 - (uC11-1*uC21)/Rload)/C11;
  //buck2
    iD22=(iL21*Rs21-U)*Rs22/(Rs21*Rs22+Rs21*Rd22+Rs22*Rd22);
    s2=diodeon2*iD22+(1-diodeon2)*iD22*Rd22;
    der(iL21) = (-iD22*Rd22- uC21)/L21;
    der(uC21) = (iL21 - (uC21-1*uC11)/Rload)/C21;
    dU=uC11-uC21; //Salida de tension diferencial
algorithm
    when time > nextT then    //comienzo periodo de PWM
      lastT:=nextT;      nextT:=nextT+T;
```

```
    Rs12 :=ROff;        Rs22 := ROff;
    DC1:=(sin(314*time)+1)*0.3+0.2;        DC2:=1-DC1;
    nextT0:=time+Delay; //Retardo para el encendido de los
        transistores
  end when;
  when time -nextT0>0 then //Encendido de los transistores retardado
    Rs11 := ROn;        Rs21 := ROn;
  end when;

  when time-lastT-DC1*T>0 then //Cambio de estado del PWM1 (Buck 1)
    Rs11 := ROff;
    nextT1:= time+Delay;  //Retardo para el encendido de los
        transistores
  end when;
  when time-nextT1>0 then //Encendido de los transistores retardado
    Rs12 := ROn;
  end when;

  when time - lastT-DC2*T>0 then //Cambio de estado del PWM2 (Buck 2)
    Rs21 := ROff;
    nextT2:= time+Delay; //Retardo para el encendido de los
        transistores
  end when;
  when time-nextT2>0 then //Encendido de los transistores retardado
    Rs22 := ROn;
  end when;

  when s1>0 then //Control del estado de conduccion de los diodos
    Rd12:=ROn;        diodeon1:=1;
  elsewhen s1<0 then
    Rd12 := ROff;        diodeon1:=0;
  end when;
  when s2>0 then
    Rd22:=ROn;       diodeon2:=1;
  elsewhen s2<0 then
    Rd22 := ROff;       diodeon2:=0;
  end when;
```

La sección *equation* del código implementa las ecuaciones (8)–(9), mientras que la sección *algorithm* se encarga de implementar los PWM de cada convertidor, así como el control de los estados de conducción de cada transistor y diodo. Además, se asignan los valores de impedancia correspondientes a cada uno de ellos.

3.3. Modelo de los DMSIs PWM buck-boost y Cuk

Dentro de las topologías DMSI más difundidas se encuentran las del tipo reductores–elevadores. Los DMSIs PWM buck–boost y Cuk se encuentras como los más utilizados cuando se necesita mayor flexibilidad con los niveles de voltaje de salida. Los esquemas circuitales básicos de estos inversores se muestran en la Figura 5.

En particular el DMSI Cuk (Pérez et al., 2017) es una de fuentes más usadas ya que presenta la ventaja de permitir implementar una aislación galvánica entre la entrada/salida y presenta gran flexibilidad de voltajes de salida. Por este motivo suele ser uno de los inversores más utilizados para sistemas de energía renovable.

Como se hizo con el DMSI buck, se obtuvieron las ecuaciones de los modelos de los inversores buck–boost y Cuk, las cuales se expresan de manera compacta

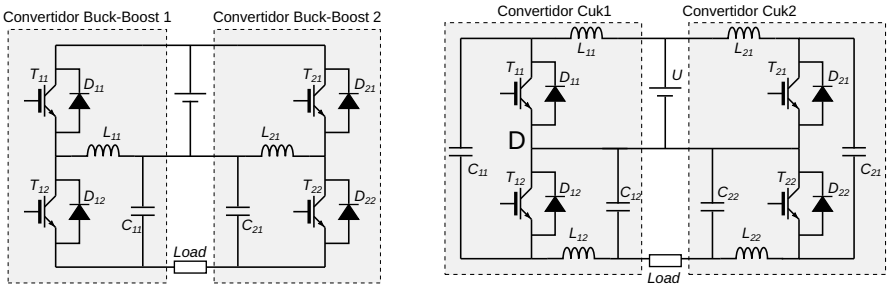

Figura 5. Inversores DMSI PWM: esquema circuital de buck–boost (izq.) y Cuk (der.).

en las Ecuaciones (10) y (11), respectivamente. A partir de estas ecuaciones, se implementaron los códigos en μ–Modelica que permiten simular estos modelos en QSS Solver y OpenModelica, de manera similar a lo mostrado para el caso del inversor buck.

$$\begin{cases} \frac{di_{L_{i1}}}{dt} = \frac{-u_{C_{i1}} - i_{D_{i2}} R_{d_{i2}}}{L_{i1}} \\ \frac{du_{C_{i1}}}{dt} = \frac{i_{D_{i2}} + i_{D_{i2}} R_{d_{i2}} / R_{s_{i2}} - (u_{C_{i,1}} - u_{C_{3-i,1}})/R_{LOAD}}{C_{i1}} \\ i_{D_{i2}} = \frac{(i_{L_{i1}} R_{s_{i1}} - u_{C_{i1}} - U) R_{s_{i2}}}{R_{s_{i1}} R_{d_{i2}} + R_{s_{i1}} R_{s_{i2}} + R_{d_{i2}} R_{s_{i2}}} \end{cases} \quad (i = 1, 2) \tag{10}$$

$$\begin{cases} \frac{di_{L_{i2}}}{dt} = \frac{-u_{C_{i2}} - i_{D_{i2}} R_{d_{i2}}}{L_{i2}} \\ \frac{du_{C_{i1}}}{dt} = \frac{i_{D_{i2}} + i_{D_{i2}} R_{d_{i2}} / R_{s_{i2}} - i_{L_{i2}}}{C_{i1}} \\ \frac{du_{C_{i2}}}{dt} = \frac{i_{L_{i2}} - (u_{C_{i2}} - u_{C_{3-i,2}})/R_{LOAD}}{C_{i2}} \\ \frac{dphi_i}{dt} = \frac{U + u_{C_{i2}} - u_{C_{i1}}}{L_{i2}} \\ i_{D_{i2}} = \frac{(R_{s_{i1}}(i_{L_{i2}} + i_{L_{i1}}) - u_{C_{i1}}) R_{s_{i2}}}{R_{s_{i1}} R_{d_{i2}} + R_{s_{i1}} R_{s_{i2}} + R_{d_{i2}} R_{s_{i2}}} \\ i_{L_{i1}} = \frac{L_{i2} phi_i + L_{i2} i_{L_{i2}}}{L_{i1}} \end{cases} \quad (i = 1, 2) \tag{11}$$

En este caso, la tensión de salida del buck–boost se define como $dU = u_{C_{11}} - u_{C_{21}}$, y la del Cuk como $dU = u_{C_{12}} - u_{C_{22}}$.

4. Resultados de simulación

Esta sección presenta los resultados de simulación de los tres modelos de inversores DMSI PWM descritos anteriormente, utilizando los parámetros mostrados en la Tabla 1. En los tres casos, se emplearon los siguientes ciclos de trabajo, que comandan cada convertidor:

$$d_{c1}(t) = 0{,}5 + 0{,}3 \sin(2\pi f_o t) \quad \text{y} \quad d_{c2}(t) = 1 - d_{c1}(t) \tag{12}$$

Con estas referencias de ciclo de trabajo, y tomando los valores $R_{\text{On}} = 1 \cdot 10^{-5}\ \Omega$ y $R_{\text{Off}} = 1 \cdot 10^{5}\ \Omega$ para las resistencias que modelan los elementos de conmutación (diodos y transistores), se obtienen las tensiones de CA de salida

Parámetro	DMSI PWM		
	buck	buck-boost	Cuk
U	$24\,V$	$24\,V$	$24\,V$
Frec.PWM	10kHz	$10\,kHz$	$10\,kHz$
R_{LOAD}	$10\,\Omega$	$10\,\Omega$	$10\,\Omega$
$L_{11} = L_{21}$	$10\,mH$	$10\,mH$	$10\,mH$

Parámetro	DMSI PWM		
	buck	buck-boost	Cuk
$L_{12} = L_{22}$	$10\,mH$	$10\,mH$	$10\,mH$
$C_{11} = C_{12}$	$10\,mF$	$10\,mF$	$10\,mF$
$C_{21} = C_{22}$	-	-	$10\,mF$

Cuadro 1. Parámetros de los DMSI PWM.

de los inversores, mostradas en la Figura 6, utilizando $LIQSS2$. Cabe mencionar que, para obtener una onda de salida efectivamente senoidal, es necesario implementar un lazo de control de la tensión de salida.

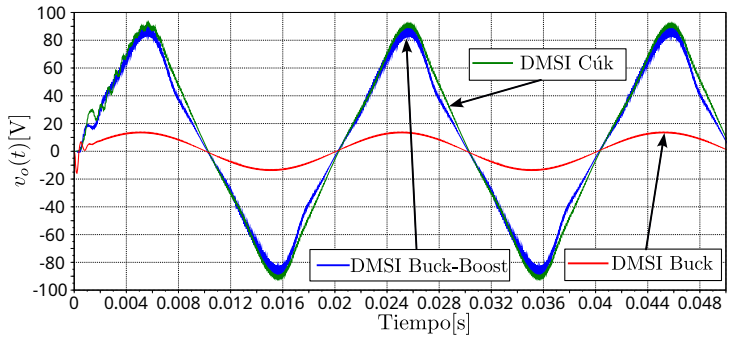

Figura 6. Tensión de salida de DMSIs tipo buck, buck–boost y Cuk.

Se midió el tiempo de CPU para simular 1 segundo cada modelo utilizando $LIQSS$ y los algoritmos clásicos $DASSL$ y $CVODE–BDF$. Además, se computó el error relativo de los resultados obtenidos en cada simulación a partir de la siguiente ecuación:

$$e_{rr} = \sqrt{\frac{\sum (x(k) - x_{ref}(k))^2}{\sum x_{ref}^2(k)}} \tag{13}$$

donde se tomaron como referencia los resultados de una simulación realizada en OpenModelica utilizando $DASSL$, con una resolución de $1 \cdot 10^{-12}$.

Tanto el tiempo de CPU como el error de simulación se utilizaron como indicadores de desempeño para comparar los algoritmos de simulación. Las simulaciones con $LIQSS$ y con los métodos clásicos $DASSL$ y $CVODE–BDF$ se ejecutaron en la misma herramienta (el QSS Solver) para que la comparación fuese independiente de la herramienta donde estén implementados los métodos. Además, para medir el desempeño del QSS Solver, los mismos modelos fueron simulados tanto en esta herramienta como en OpenModelica 1.24.5 utilizando $DASSL$. Estas herramientas se ejecutaron en un sistema operativo Ubuntu, utilizando un procesador (CPU) Intel(R)Core (TM) i3-2350MCPU @ 2.30GHz.

En la Tabla 2 puede verse los indicadores de desempeño obtenidos en las simulaciones del DMSI buck. Puede verse que en este caso $LIQSS2$ es 1.75 veces más rápido que $CVODE - BDF$ y 3.84 veces más rápido que $DASSL$. Además, se observa que, si bien todos los métodos cumplen con la tolerancia de error especificada en la simulación ($err = 1{\cdot}10^{-3}$), el valor obtenido con $LIQSS2$ es 2 veces menor que con $DASSL$ y 1.5 veces menor que con $CVODE$. Además, si se observan los tiempos de CPU requeridos por el QSS Solver y OpenModelica para simular este modelo utilizando en ambos casos $DASSL$, resulta evidente que la primera herramienta es mucho más eficiente (15.5 veces más rápida). Esto se debe principalmente a que la detección de eventos implementada en dicho solver es considerablemente más eficiente.

Método de Integración		**Error Relativo**	**Tiempo de CPU[ms]**
LIQSS2 (QSS-Solver)	$\Delta Q_i = 10^{-3}$	$0{,}7 \cdot 10^{-3}$	159,4
CVODE (QSS-Solver)	err.tol.=10^{-3}	$1{,}0 \cdot 10^{-3}$	278,9
DASSL (QSS-Solver)	err.tol.=10^{-3}	$1{,}85 \cdot 10^{-3}$	610,4
DASSL (OpenModelica)	err.tol.=10^{-3}	$1{,}51 \cdot 10^{-3}$	9449,0

Cuadro 2. Resultados de simulación del DMSI buck.

De manera equivalente, la Tabla 3 muestra los índices de desempeño obtenidos en el caso del inversor buck–boost. En este caso, la simulación realizada con $LIQSS2$ es 1.71 veces más rápida que con $CVODE$–BDF, y 4.47 veces más rápida que con $DASSL$, ambos implementados en el mismo solver. Respecto del *error relativo*, el valor obtenido con $LIQSS2$ es 4 veces menor que con $DASSL$ y $CVODE$ implementados en la misma herramienta, y 15 veces menor que el obtenido con $DASSL$ en OpenModelica. Además, se observa que el QSS Solver es 20 veces más rápido que OpenModelica al simular este modelo con $DASSL$. Notar que los resultados obtenido en el QSS Solver cumplen con la resolución especificada mientras los obtenidos en OpenModelica no lo hacen. Esto puede atribuirse a un mayor error en la detección de las discontinuidades en OpenModelica.

Método de Integración		**Error Relativo**	**Tiempo de CPU[ms]**
LIQSS2(QSS-Solver)	$\Delta Q_i = 10^{-3}$	$0{,}9 \cdot 10^{-3}$	143,8
CVODE(QSS-Solver)	err.tol.=10^{-3}	$3{,}6 \cdot 10^{-3}$	244,2
DASSL(QSS-Solver)	err.tol.=10^{-3}	$3{,}7 \cdot 10^{-3}$	640,9
DASSL (OpenModelica)	err.tol.=10^{-3}	$1{,}4 \cdot 10^{-2}$	13256,0

Cuadro 3. Resultados de simulación del DMSI buck–boost.

Por último, los índices de desempeño obtenidos en el caso del inversor Cuk pueden verse en la Tabla 4. En ella se observa que $LIQSS2$ es 1.3 veces más rápido que $CVODE$–BDF y 3 veces más rápido que $DASSL$, ambos ejecutados en el QSS Solver. Respecto al *error relativo*, se observa que todos los métodos

implementados en el QSS Solver cumplen con la resolución especificada, mientras que el resultado obtenido en OpenModelica no lo hace. Esto puede deberse a la diferencia en la implementación de la detección de discontinuidades. Además, OpenModelica resulta 17 veces más lento que el QSS Solver.

Método de Integración		**Error Relativo**	**Tiempo de CPU[ms]**
LIQSS2(QSS-Solver)	$\Delta Q_i = 10^{-3}$	$2{,}2 \cdot 10^{-3}$	350,7
CVODE(QSS-Solver)	err.tol.=10^{-3}	$5{,}0 \cdot 10^{-3}$	447,7
DASSL(QSS-Solver)	err.tol.=10^{-3}	$6{,}5 \cdot 10^{-3}$	1057,2
DASSL (OpenModelica)	err.tol.=10^{-3}	$2{,}2 \cdot 10^{-2}$	18173,0

Cuadro 4. Resultados de simulación para el DMSI Cuk.

A través de estos ejemplos se puede observar que los DMSIs PWM pueden ser simulados de manera más eficiente con $LIQSS$ que con los métodos tradicionales. Esto se evidencia no solo en términos de ganancia en el tiempo de CPU, sino también en el mejor cumplimiento de la tolerancia de error especificada para la simulación. Además, debe tenerse en cuenta que se está comparando un método de segundo orden ($LIQSS2$) con métodos de quinto orden.

5. Conclusiones y trabajos futuros

Se presentó la implementación de modelos realistas de tres topologías de DMSIs PWM. Estos modelos fueron programados utilizando el lenguaje μ–Modelica e implementados en el QSS Solver y en OpenModelica.

Se mostró que la simulación de estos modelos mediante métodos $LIQSS$ permite obtener resultados más rápidos y precisos que utilizando métodos basados en la discretización temporal. Esto permite reducir los tiempos de CPU sin pérdida de precisión en la simulación de este tipo de sistemas. Esta mejora en el rendimiento computacional se debe a que, a diferencia de los métodos QSS, los métodos clásicos deben ser reinicializados luego de la ocurrencia de cada discontinuidad, lo cual incrementa el tiempo de cómputo. Además, la detección de eventos temporales y de estado en los métodos QSS no implica un costo mayor al de un paso normal.

Si bien el uso de los métodos $LIQSS$ logró mejoras significativas en la simulación de DMSIs PWM, estas resultan insuficientes cuando se requiere simular sistemas que integran estos inversores y presentan elevados tiempos de simulación, como es el caso de grandes sistemas de generación de energía. Esto abre la puerta a trabajos futuros que deben orientarse en este sentido. El objetivo inicial es investigar el desempeño de los métodos de integración QSS en la simulación de DMSIs PWM, combinándolos con nuevas estrategias de modelado mixto (Bortolotto y Migoni, 2025). Se espera que el uso de estos algoritmos, basados en la cuantificación de los estados en reemplazo de métodos clásicos para la simulación de modelos mixtos, reduzca significativamente los requerimientos de CPU.

Referencias

Albakri, M., Darwish, A., & Twigg, P. (2024). A Comprehensive Review of DC/AC Single-Phase Differential-Mode Inverters for Low-Power Applications. *Electronics*, *13*(13), 2474.

Bergero, F., Floros, X., Fernández, J., Kofman, E., & Cellier, F. E. (2012). Simulating Modelica Models with a Stand-Alone Quantized State Systems Solver. En H. Elmqvist & M. Otter (Eds.), *Proceedings of the 9th International Modelica Conference* (pp. 237-246, Vol. 76). Linköping University Electronic Press.

Bortolotto, M., & Migoni, G. (2025). Mixed Modeling Approach for Efficient Simulation of Single-Phase PWM Inverters. *SIMULATION*, *101*(1), 73-85.

Cellier, F., & Kofman, E. (2006). *Continuous System Simulation*. Springer.

Cellier, F., Kofman, E., Migoni, G., & Bortolotto, M. (2008). Quantized State System Simulation. *Proceedings of SummerSim 08 (2008 Summer Simulation Multiconference)*.

Erickson, R. W., & Maksimovic, D. (2020). *Fundamentals of Power Electronics* (3rd edition). Springer Cham.

Fernández, J., & Kofman, e. (2014). A stand-alone quantized state system solver for continuous system simulation. *SIMULATION*, *90*(7), 782-799.

Kofman, E., & Junco, S. (2001). Quantized State Systems. A DEVS Approach for Continuous System Simulation. *Transactions of SCS*, *18*(3), 123-132.

Kofman, E., Lee, J., & Zeigler, B. (2001). DEVS Representation of Differential Equation Systems. Review of Recent Advances. *Proceedings of ESS'01*, 591-595.

Migoni, G., Bortolotto, M., Kofman, E., & Cellier, F. E. (2013). Linearly Implicit Quantization-Based Integration Methods for Stiff Ordinary Differential Equations. *Simulation Modelling Practice and Theory*, *35*, 118-136.

Mohan, N. (2012). *Power Electronics: A First Course*. Wiley.

Pérez, S., Vivert, M., Díez, R., & Patino, D. (2017). Modeling and control of a grid tied differential Cuk Inverter. *2017 IEEE Workshop on Power Electronics and Power Quality Applications (PEPQA)*, 1-9.

Petzold, L. R. (1983). A description of DASSL: a differential/algebraic system solver. En *Scientific computing (Montreal, Quebec, 1982)* (pp. 65-68). IMACS.

Simões, M. G., & Farret, F. A. (2016). *Modeling Power Electronics and Interfacing Energy Conversion Systems*. Wiley-IEEE Press.

Zeigler, B., Muzy, A., & Kofman, E. (2018). *Theory of Modeling and Simulation. 3rd Edition.* Academic Press.