CONVERSOR BUCK: CONVERSÃO DE MATLAB PARA SCILAB

BUCK CONVERTER: CONVERSION FROM MATLAB TO SCILAB

CONVERTIDOR BUCK: CONVERSIÓN DE MATLAB A SCILAB

Nirlan Neckir Zamprogno de Souza[1]
Eliane Silva Custódio[2]
Maria Aline Gonçalves[3]

Resumo

O estudo propõe a conversão de um modelo de conversor Buck que foi originalmente desenvolvido em MATLAB/Simulink, para o ambiente Scilab/Xcos. Essa conversão visa oferecer uma alternativa gratuita para modelagem e simulação em eletrônica de potência. Tanto o MATLAB quanto o Scilab possuem operadores e funções fundamentais em comum. A principal diferença entre os dois *softwares* está nas funções específicas para a criação e manipulação de funções de transferência. Esse estudo também explora a conversão de blocos específicos do Simulink para o Xcos. Nesse trabalho, são obtidas com sucesso as funções de transferência, os diagramas de Bode e a resposta a um trem de impulsos. Apesar do Scilab ter uma documentação menor, menos didática e com menos exemplos de aplicação do que o MATLAB, o presente estudo demonstrou que é perfeitamente possível converter uma modelagem realizada em MATLAB e Simulink para Scilab e Xcos e obter quase todas as informações do modelo.

Palavras-chave: conversor Buck; MATLAB; Scilab; função de transferência; discretização.

Abstract

The study proposes the conversion of a buck converter model, originally developed in MATLAB/Simulink, to the Scilab/Xcos environment. The objective of this conversion is to provide a cost-free alternative for modeling and simulation in the field of power electronics. Both MATLAB and Scilab are based on a similar set of fundamental operators and functions. The primary distinction between the two software platforms pertains to the specific functions for creating and manipulating transfer functions. Furthermore, this study examines the conversion of Simulink blocks to Xcos. In this process, transfer functions, bode diagrams, and impulse train responses have been successfully obtained. Despite Scilab having less comprehensive documentation, a less didactic approach, and a smaller selection of application examples in comparison to MATLAB, the present study has demonstrated that it is feasible to convert a model developed in MATLAB and Simulink to Scilab and Xcos, and to retrieve most of the information from the model.

Keywords: Buck converter; MATLAB; Scilab; transfer function; discretization.

Resumen

El estudio propone la conversión de un modelo de convertidor Buck, originalmente desarrollado en MATLAB/Simulink, para el Scilab/Xcos. Esa conversión tiene como objetivo ofrecer una alternativa gratuita para la modelización y simulación en electrónica de potencia. Tanto MATLAB como Scilab comparten operadores y funciones fundamentales. La principal diferencia entre los dos programas radica en las funciones específicas para la creación y manipulación de funciones de transferencia. Ese estudio también explora la conversión de bloques específicos de Simulink a Xcos. En ese trabajo se obtienen con éxito las funciones de transferencia, los diagramas de Bode y la respuesta a un tren de impulsos. A pesar de que Scilab tiene menos documentación, es menos didáctico y ofrece menos ejemplos de aplicación en comparación con MATLAB, el presente estudio demostró que es perfectamente posible convertir una modelización realizada en MATLAB y Simulink a Scilab y Xcos y obtener casi toda la información del modelo.

[1] Estudante do curso de Engenharia da Computação – UNINTER. E-mail: 3539631@alunouninter.com.
[2] Professora Mestre em Engenharia Elétrica – Escola Superior Politécnica – UNINTER. E-mail: eliane.s@uninter.com.
[3] Professora Mestre em Engenharia Elétrica – Escola Superior Politécnica – UNINTER. E-mail: 4584871@alunouninter.com.

Palabras clave: Convertidor Buck; MATLAB; Scilab; función de transferencia; discretización.

1 Introdução

O MATLAB é uma plataforma de programação e cálculo extensamente utilizada no mundo acadêmico das engenharias para modelagem matemática (Ibrahim, 2006; Janík; Žáková, 2011). Porém, por ser um software privado, e pago, muitas vezes o valor da sua licença é impeditivo para que instituições de ensino de orçamento reduzido e estudantes de baixa renda o utilizem (Ibrahim, 2006; Pendharkar, 2005). A melhor opção *open source*, gratuita disponível, é o Scilab, que fornece funcionalidades semelhantes (Ibrahim, 2006; Janík; Žáková, 2011), apesar de ainda apresentar deficiências significativas (Silva; Cunha, 2006).

O objetivo desse trabalho é contribuir para a difusão e ensino do uso do Scilab, em particular para modelagem e simulação de sistemas de controle em eletrônica de potência. Para isso, foi realizada a conversão do código do projeto de um conversor Buck com compensador e controle digital modelado em MATLAB (Custódio; Gonçalvez; Pedroso, 2023) para Scilab, bem como a conversão do circuito modelado graficamente em diagrama de blocos do Simulink (MATLAB) para o Xcos (Scilab).

Em um país em desenvolvimento, como o Brasil, com grandes dificuldades de investimento em educação e pesquisa, e uma população predominantemente de baixa renda, é importante democratizar o acesso a softwares gratuitos de modelagem computacional. Para isso, tentou-se mostrar que é possível e viável converter uma modelagem realizada em MATLAB e Simulink para Scilab e Xcos, permitindo que esses possam ser uma opção para projetos de pesquisa com orçamento curto.

Na seção dois, a seguir, é feita uma breve apresentação do Scilab e do MATLAB. Na seção três é apresentado o conversor Buck utilizado, a metodologia para conversão do código de MATLAB para Scilab e do diagrama em Simulink para Xcos. Na seção quatro são apresentados e discutidos os resultados da conversão. Por fim, na seção cinco, são realizadas as considerações finais do projeto.

2 Scilab e MATLAB

O Scilab nasceu a partir do software Blaise, criado nos anos 80 no IRIA (*Institut national de recherche en sciences et technologies du numérique*), pela contribuição de François Delebecque e Serge Steer. Foi inspirado no Matlab Fortran de Cleve Moler, quem mais tarde foi um dos fundadores da *The MathWorks* (Scilab, 2024).

A primeira versão *open source* do Scilab foi liberada em 1994 pela INRIA (*Institut National de Recherche en Sciences et Technologies du Numérique*). Hoje o software é mantido pelo *Scilab Team* dentro da *Dassault Systèmes* e é distribuído sob a licença GPL (*General Purpose License*), a qual permite usar, alterar e/ou distribuir o software gratuitamente (Scilab, 2024).

O Scilab é constituído de três programas: o Scilab propriamente dito, que permite realizar cálculo numérico, visualização de dados, desenvolvimento de algoritmos e de aplicações; o Xcos, que permite a modelagem visual e simulação de sistemas dinâmicos em domínio de tempo contínuo e discreto, incluindo processamento de sinais, sistemas de controle, sistemas mecânicos, termodinâmicos e eletrônicos; e o ATOMS (*Automatic Modules Management for Scilab*) que é o repositório de módulos de extensão "*Toolboxes"* para incorporar novas funcionalidades ao software (Scilab, 2024). O Scilab está disponível em versões para GNU/Linux, Windows e macOS, podendo ser baixado pelo site oficial do Scilab (Scilab, 2024).

O MATLAB é um software criado e comercializado pela The MathWorks desde a sua fundação, em 1984, por Jack Little e Cleve Moler. O MATLAB tem as mesmas funcionalidades que o Scilab, e possui um ambiente de modelagem visual e simulação, o Simulink, que é semelhante ao Xcos. Da mesma forma que o ATOMS, o MATLAB tem diversas extensões que podem ser compradas separadamente, e que inclui aplicações para diversas áreas como design de controle, processamento de sinais, comunicações, processamento de imagens, biologia computacional, modelagem e análise financeira, robótica, *machine learning*, inteligência artificial etc. (Matlab, 2024).

Tanto o Simulink como o Xcos utilizam o conceito de programação visual para a modelagem gráfica do circuito. Ambos fazem uso de um painel de ferramentas com blocos padronizados configuráveis que permitem a edição do circuito de modo visual, o que é de grande interesse para os usuários envolvidos em modelagem computacional (Shakin *et al*., 2020).

3 Metodologia

Esse é um trabalho de pesquisa de natureza aplicada, realizado a partir de uma abordagem qualitativa, com objetivo prescritivo de ajudar a propor uma solução para conversão de circuitos de potência com controle modelados em MATLAB para Scilab, por meio do estudo do caso apresentado.

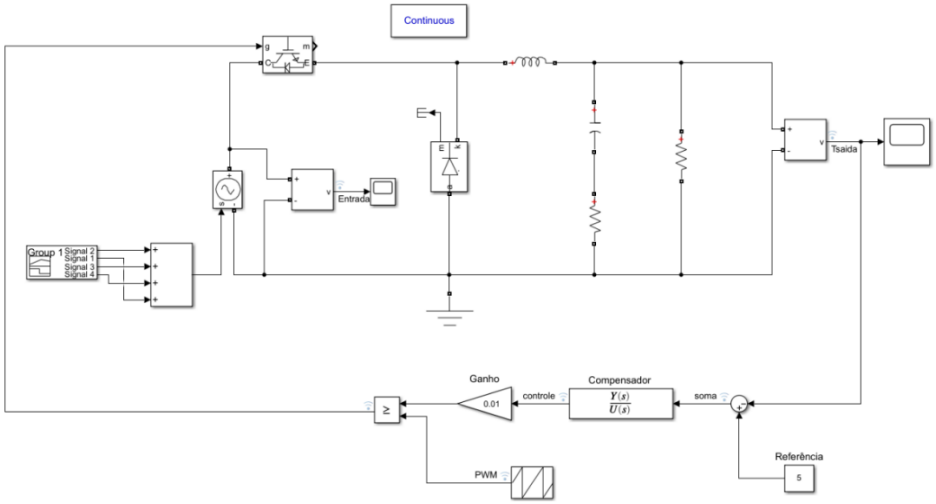
Esse projeto foi realizado utilizando Scilab versão 2024.0.0 e MATLAB versão 24.1.0.2603908, instalados em ambiente Windows 10, em um computador de processador Intel Core i7, de sétima geração, com 16 GB de RAM DDR4, placa de vídeo dedicada GeForce GTX 1050 Ti com 4 GB de GDDR5, e unidade de armazenamento SSD.

3.1 Conversor Buck no MATLAB/Simulink

Os circuitos eletrônicos de potência têm como objetivo alterar a tensão ou a corrente de um circuito, de modo que a fonte de alimentação atenda às especificações da carga. Eles são chaveados por semicondutores, que pela alteração da razão cíclica permitem controlar a saída do circuito. Eles têm ampla aplicação na indústria e estão presentes em todos os circuitos que processam potência. Por modificarem a tensão ou a corrente de saída, são chamados de conversores. Nesse sentido, o conversor Buck é um conversor do tipo CC-CC (CC – corrente contínua) abaixador de tensão.

O trabalho de Custódio; Gonçalvez; Pedroso (2023) modelou em MATLAB e Simulink um conversor Buck linear invariante no tempo (LIT), com operação em modo contínuo e não isolado. A tensão de saída foi regulada pelo modo de controle de tensão. Desse modo, a tensão de saída é realimentada e comparada a uma referência fixa. A saída do comparador é recebida por um compensador do tipo III fator K. A saída compensada é associada a uma chave PWM (*Pulse-width modulation)* dente-de-serra de 30 kHz com amplitude de -1 V a +1 V por meio de um operador relacional, cuja saída aciona a chave do Buck. O funcionamento desse conjunto determina o valor da razão cíclica do Buck, regulando a saída do conversor, conforme a Figura 1 do diagrama de blocos no Simulink.

Figura 1: Projeto do conversor Buck com compensador tipo III



Fonte: Custódio; Gonçalvez; Pedroso (2023).

3.2 Funções equivalentes entre MATLAB e Scilab

Os operadores aritméticos, lógicos e relacionais, e a forma de definir variáveis são as mesmas entre MATLAB e Scilab, que possuem a mesma sintaxe da linguagem C. A grande diferença está nas funções. A seguir são expostas as funções utilizadas para a conversão do código de MATLAB para Scilab.

No MATLAB, a criação de uma função de transferência é realizada por meio da função *tf*. A função *tf* recebe dois argumentos principais: os coeficientes do numerador e os coeficientes do denominador da função de transferência. Ambos são passados como matrizes, linha de coeficientes, em ordem decrescente de potência de *s*, com os elementos de cada matriz separados por espaço em branco, e cada matriz delimitada por colchetes, separadas entre si por uma vírgula (Matlab, 2024).

A criação da função de transferência no Scilab, por sua vez, é feita com a chamada da função *poly (0,'s')* que cria um polinômio com o símbolo *s*. A partir dela é possível fazer a manipulação simbólica de *s*. Em seguida, chama-se a função *syslin ('c', polinômio)* em que o argumento *polinômio* é a função de transferência desejada. A sintaxe para a definição de polinômio segue a sintaxe dos operadores aritméticos da linguagem C, com o uso do símbolo *s* definido na chamada da função *poly*.

A discretização da função de transferência no MATLAB é feita com o uso da função *c2d*. Assim, o usuário pode escolher o método de discretização (como ZOH ou Tustin) e especificar o tempo de amostragem. No Scilab, a discretização de uma função de transferência necessita da chamada de três funções. Para o método ZOH, a conversão é feita por meio da chamada das funções *tf2ss*, *dscr* e *ss2tf*. Já para a discretização pelo método Tustin, no Scilab, deve-se utilizar as funções *tf2ss*, *cls2dls* e *ss2tf*.

3.3 Blocos equivalentes entre Simulink e Xcos

A conversão de modelos e simulações do MATLAB/Simulink para o Scilab/Xcos envolve a substituição de diversos blocos funcionais pelos seus equivalentes no ambiente Scilab/Xcos. Embora os dois *softwares* compartilhem conceitos fundamentais semelhantes, existem diferenças importantes na implementação e configuração dos blocos, o que pode afetar o processo de migração. A seguir, discutem-se algumas das principais comparações entre blocos equivalentes.

3.3.1 Signal Builder

No Simulink, o bloco *Signal Builder* permite a geração de múltiplos sinais com diferentes formas de onda a partir de um único bloco. No Xcos, o bloco equivalente só permite a geração de um único sinal por bloco. Para replicar a funcionalidade completa do *Signal Builder* do Simulink, é necessário utilizar múltiplos blocos *Signal Builder* no Xcos, configurando cada um individualmente.

Na caixa de configuração dos parâmetros do bloco *Signal builder* do Xcos, o usuário deve definir o número de pontos a serem interpolados, o vetor de valores do eixo x, do eixo y, selecionar se o sinal é periódico ou não e se deseja ver o gráfico do sinal.

3.3.2 Add/Summation

O bloco *Add* no Simulink permite a soma ou subtração de múltiplos sinais, com configuração das entradas por meio de um menu gráfico em que o usuário escolhe quais sinais devem ser somados ou subtraídos. O bloco equivalente no Xcos é o *Summation*. A configuração das entradas no Xcos é feita utilizando uma notação matricial, em que cada par [1; -1] define se a entrada correspondente será somada ou subtraída.

Na caixa de configuração de parâmetros do bloco *Summation* do Xcos, no campo *Number of inputs or sign vector*, deve-se determinar o número de entradas em pares. Por exemplo, para criar 5 entradas que serão somadas e 5 que serão subtraídas, faz-se da seguinte forma: [1; -1] [1; -1] [1; -1] [1; -1] [1; -1]. Se algum valor do par é omitido, ele é lido como zero. Por exemplo [1;] [1;] [1;] [1;] [1;] são criadas 5 entradas que serão somadas.

3.3.3 Controlled Voltage Source

Tanto no Simulink como no Xcos, o bloco CVS (*Controlled Voltage Source*) permite a implementação direta de uma fonte de tensão controlada. Nas duas plataformas ele é utilizado da mesma forma, e já vem pré-configurado, o que facilitam a modelagem, convertendo um sinal de entrada em uma fonte de tensão equivalente. O CVS no Xcos tem 3 terminais. No terminal triangular preto deve ser conectado o sinal de entrada (vindo de um somador, por exemplo). O terminal quadrado preto é o polo positivo e o terminal quadrado branco é o polo negativo.

3.3.4 Osciloscópio

O osciloscópio no Xcos é o bloco CSCOPE. Para que ele funcione adequadamente, deve-se conectar a entrada de dados (terminal triangular preto) em um MUX, que deverá

receber todas as entradas que serão observadas no osciloscópio. É necessário, ainda, acrescentar um bloco CLOCK_c ligado à entrada de controle (terminal triangular vermelho) que definirá a frequência de amostragem do osciloscópio. Clicando em qualquer elemento com o botão direito do mouse é possível selecionar os parâmetros do bloco para configurá-lo.

A frequência de amostragem no Xcos é definida em termos do período (em segundos), que é o inverso da frequência. O tempo de inicialização é o atraso em segundos com relação ao primeiro disparo do clock. Diferentemente, no Simulink, a frequência de amostragem do osciloscópio é herdada diretamente do elemento que é medido.

O elemento equivalente no Simulink é o Scope, em que são conectados diretamente os sinais dos blocos a serem amostrados. A taxa de amostragem do Scope é herdada do sinal medido por padrão (*Sample time*: -1), porém, pode ser definida manualmente em termos do período em segundos, clicando sobre a conexão do sinal com o botão direito do *mouse* e inserindo o período de amostragem no campo *Sample time*.

3.3.5 Bloco CLR (Xcos) e TransferFunction (Simulink)

Para definir a função de transferência em malha aberta do compensador no Xcos, é utilizado o bloco CLR. Nele é possível editar as propriedades e especificar a equação do numerador e do denominador da função de transferência nos respectivos campos, com a mesma sintaxe da linguagem C.

Aqui é interessante frisar que o número de Euler no Scilab é definido como *% e*, e a constante pi é representada por *% pi*. No Matlab é somente *e* e *pi*. O Xcos permite a definição de variáveis no contexto da simulação, permitindo utilizar os símbolos no lugar dos valores em si, o que facilita a mudança de valores para teste, uma vez que as variáveis normalmente são usadas em mais de um local. Se não forem utilizados símbolos, os valores teriam que ser alterados manualmente, um a um, em cada local em que elas se repetem, toda vez que se desejasse mudar o valor de uma delas. Assim, para melhor visualização de uma equação, pode ser definido no contexto que *e = % e*. No Simulink, o bloco equivalente é o *Transfer Fcn*. Em suas propriedades são definidos os coeficientes do numerador e do denominador da função de transferência nos respectivos campos.

No Simulink, os parâmetros do bloco *Transfer Fcn* são obtidos a partir da variável definida no código que armazena a função de transferência. Essa variável é na verdade é uma estrutura que contém dentro dela duas matrizes: *Numerator*, que armazena os coeficientes do

numerador da FT, e *Denominator*, que armazena os coeficientes do denominador da FT. Ambos ficam em ordem decrescente de potência de *s*.

Considerando uma variável *cc* que contenha uma FT de um sistema *Single Input single Output* (SISO), o numerador da FT é definido como *cc. Numerator {1,1}* e o denominador é definido como *cc. Denominator {1,1}*. Como é um sistema *Single Input single Output* (SISO), utiliza-se somente a célula 1,1 de cada matriz. Se fosse um sistema *Multiple Input Multiple Output* (MIMO) os diferentes numeradores e denominadores poderiam ser definidos como {1,2}, {2,1} fazendo referência às FTs dos diferentes canais do sistema.

3.3.6 PWN

No Xcos, o PWM é modelado com o uso do elemento SAWTOOTH_f, que é controlado pelo período de amostragem do elemento SampleCLK. O elemento SAWTOOTH_f gera um sinal em dente de serra, cujo período é definido nos parâmetros do SampleCLK que é ligado na entrada de controle (vermelha) do elemento SAWTOOTH_f. Por padrão, o sinal do elemento SAWTOOTH_f se inicia em 0 V e vai até a amplitude máxima de 1 V, quando o SampleCLK está configurado para um período de amostragem de 1 s. Se o período for definido para 2 s, a amplitude máxima do dente de serra ficará em 2 V. Por outro lado, se o período for definido em 0,1 s, o sinal terá amplitude máxima de 0,1 V, e assim por diante.

O sinal em dente de serra é ligado a um bloco comparador, que compara o sinal em dente de serra com o sinal do compensador. Utilizando um comparador configurado em maior ou igual (≥) com o sinal em dente de serra ligado na entrada superior do comparador, sempre que o sinal dente de serra for maior ou igual que o sinal do compensador, a saída do comparador será 1. Portanto, uma variação positiva do sinal do compensador diminui a razão cíclica, e uma variação negativa do compensador aumenta a razão cíclica do conversor.

Já no MATLAB, o PWM é implementado usando o elemento *Sawtooth Generator* ligado diretamente a um comparador, o qual gera um sinal em dente de serra de 1 V a – 1 V, cuja frequência em Hz é definida diretamente em suas propriedades.

3.3.7 Diagrama de Bode

O diagrama de Bode é plotado tanto no Scilab como no MATLAB, usando a função *bode*, porém, elas não têm a mesma implementação. No MATLAB, pode plotar diretamente o diagrama passando como argumento a FT do sistema conforme exemplo abaixo.

Ela também pode ser usada para obter um vetor com as informações de fase e ganho em cada frequência. Nesse caso, a função não plota o gráfico diretamente, sendo necessário usar os valores do vetor *[mag, phase, wout]* como argumento de outras funções para plotar os gráficos.

No Scilab, o diagrama de Bode também é plotado pela chamada da função *bode* com a FT do sistema como argumento, de modo semelhante ao MATLAB, porém, com algumas diferenças importantes. Essa função no Scilab não tem a opção de retornar um vetor. Toda a personalização do gráfico deve ser feita pela passagem de argumentos na chamada da função. Além disso, cada chamada da função *bode* deve ser precedida da chamada da função *figure* para que cada diagrama de Bode seja plotado em uma janela de visualização individual.

4 Resultados e discussão

Considerando-se *GpBuck = N/D* como sendo a função de transferência em malha aberta (FTMA) do conversor Buck definida pelo projeto, conforme a Equação 1 abaixo:

$$GpBuck = \frac{Vin.Ro.C.Rse.s + Vin*Ro}{L*C*(Ro+Rse).s^2 + C*[(Ro*RL+Ro*Rse+Rse*RL)+L].s+(Ro+RL)} \qquad \textbf{(1)}$$

Criada no MATLAB pela chamada da função *tf([numerador], [denominador])*, que recebe dois argumentos. O primeiro são os coeficientes do numerador *N*, o segundo são os coeficientes do denominador *D*. O código abaixo mostra a implementação em MATLAB da Equação 1: GpBuck = tf ([Vin*Ro*C*Rse Vin*Ro], [(L*C*(Ro+Rse)) (C*(Ro*RL+Ro*Rse+Rse*RL) +L) (Ro+RL)]).

No Scilab, a criação da mesma função de transferência *GpBuck* (Eq. 1) é feita conforme o código a seguir, com uso das funções *poly* e *syslin*:

```
s = poly(0,'s');
H = (Vin*Ro*C*Rse*s + Vin*Ro) /(L*C*(Ro+Rse).
s^2+(C*(Ro*RL+Ro*Rse+Rse*RL) +L) *s+Ro+RL);
GpBuck = syslin ('c', H).
```

FTMA é a função de transferência em malha aberta não compensada do conversor, considerando as funções de transferência do PWM, do elemento sensor da tensão de saída, do conversor analógico-digital e do filtro. O código completo pode ser visto no anexo I.

No projeto do conversor Buck com compensador, além da função de transferência em malha aberta do conversor Buck (Eq. 1), foram definidas, ainda, a função de transferência em malha aberta do compensador *cc* (Eq. 2) e a função de transferência em malha aberta do conversor com o compensador *FTMAC* (Eq. 3):

$$cc = \frac{\frac{kc}{wz^2}s^2 + 2.\frac{kc}{wz}s + kc}{\frac{1}{wp^2}s^3 + \frac{2}{wp}s^2 + s} \quad \textbf{(2)}$$

$$FTMAC = FTMA.cc \quad \textbf{(3)}$$

No MATLAB, *cc* e *FTMAC* são definidas da seguinte forma:

```
cc = tf([(kc/wz^2) (2*kc/wz) (kc)], [(1/wp^2) (2/wp) 1 0])
FTMAC = FTMA*cc;
```

No Scilab as duas funções são implementadas do seguinte modo:

```
C = (kc/wz^2) *s^2+(2*kc/wz) *s+kc)/(1/wp^2) *s^3+(2/wp) *s^2+s).
cc = syslin ('c', C).
FTMAC = FTMA*cc.
```

Pelas Figuras 2 e 3, podem ser observadas, respectivamente, a criação com sucesso das funções de transferência no Scilab e no MATLAB, com resultados iguais:

Figura 2: Resultado de *GpBuck, cc e FTMAC* no console do Scilab

```
--> GpBuck
 GpBuck  =

          46.666667 +0.0001944s
   ---------------------------------
   1.6666667 +0.0005774s +2.615D-08s^2

--> cc
 cc  =

   979.13479 +0.2508113s +0.0000161s^2
   ---------------------------------
      s +0.0000439s^2 +4.829D-10s^3

--> FTMAC
 FTMAC  =

           42535.989 +11.073083s +0.0007432s^2 +2.907D-09s^3
   ------------------------------------------------------------------
   1.6666667s +0.0006506s^2 +5.233D-08s^3 +1.428D-12s^4 +1.263D-17s^5
```

Fonte: elaborado pelos autores (2024).

Figura 3: Resultado de *GpBuck*, *cc* e *FTMAC* no console do MATLAB

```
>> GpBuck

GpBuck =

          0.0001944 s + 46.67
  ---------------------------------
  2.615e-08 s^2 + 0.0005774 s + 1.667

Continuous-time transfer function.
Model Properties
>> cc

cc =

  1.606e-05 s^2 + 0.2508 s + 979.1
  --------------------------------
  4.829e-10 s^3 + 4.395e-05 s^2 + s

Continuous-time transfer function.
Model Properties
>> FTMAC

FTMAC =

            2.907e-09 s^3 + 0.0007432 s^2 + 11.07 s + 4.254e04
  ------------------------------------------------------------------------
  1.263e-17 s^5 + 1.428e-12 s^4 + 5.233e-08 s^3 + 0.0006506 s^2 + 1.667 s

Continuous-time transfer function.
Model Properties
```

Fonte: elaborado pelos autores (2024).

A função de transferência do compensador discretizada pelo método de Tustin, *cz* é implementada no MATLAB conforme o código abaixo:

```
cz = c2d (cc, ts,'tustin');
```

No Scilab, a discretização de *cc* pelo método de Tustin é implementada do seguinte modo:

```
cz = ss2tf(cls2dls(tf2ss(cc), ts));
```

As Figuras 4 e 5 mostram o mesmo resultado para *cz* nas duas plataformas:

Figura 4: Resultado de *cz* no console do Scilab

```
--> cz
 cz  =

   0.1356578 -0.2168332z -0.1235143z^2 +0.2289766z^3
   ------------------------------------------------
       -0.0188697 +0.2936034z -1.2747337z^2 +z^3
```

Fonte: elaborado pelos autores (2024).

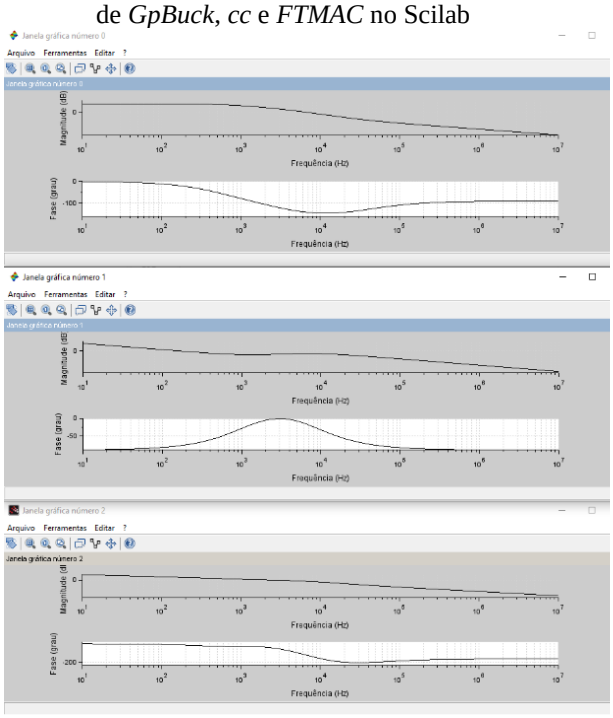
Figura 5: Resultado de *cz* no console do MATLAB

```
cz =

  0.229 z^3 - 0.1235 z^2 - 0.2168 z + 0.1357
  ------------------------------------------
     z^3 - 1.275 z^2 + 0.2936 z - 0.01887

Sample time: 3.3333e-05 seconds
Discrete-time transfer function.
Model Properties
```

Fonte: elaborado pelos autores (2024).

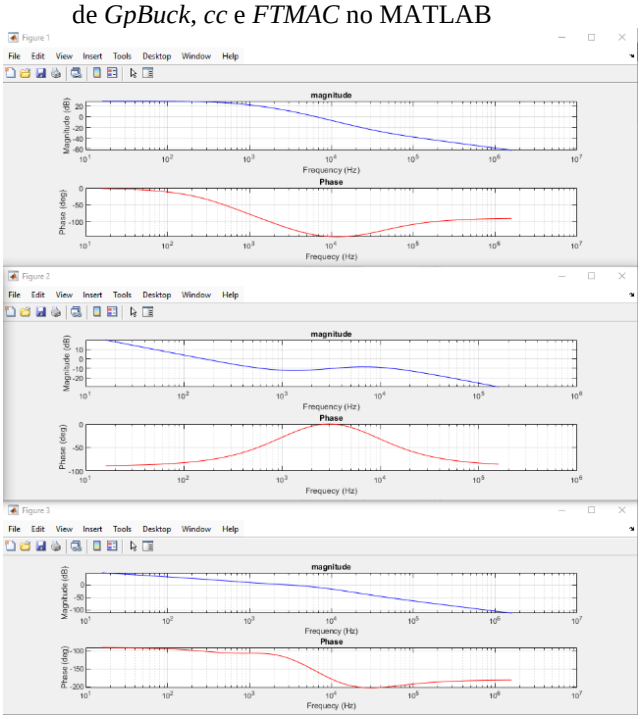
Nas Figuras 6 e 7, a seguir, são mostrados os resultados do diagrama de Bode de *GpBuck*, *cc* e *FTMAC* no Scilab e no MATLAB, respectivamente:

Figura 6: De cima para baixo, diagrama de Bode de *GpBuck*, *cc* e *FTMAC* no Scilab



Fonte: elaborado pelos autores (2024).

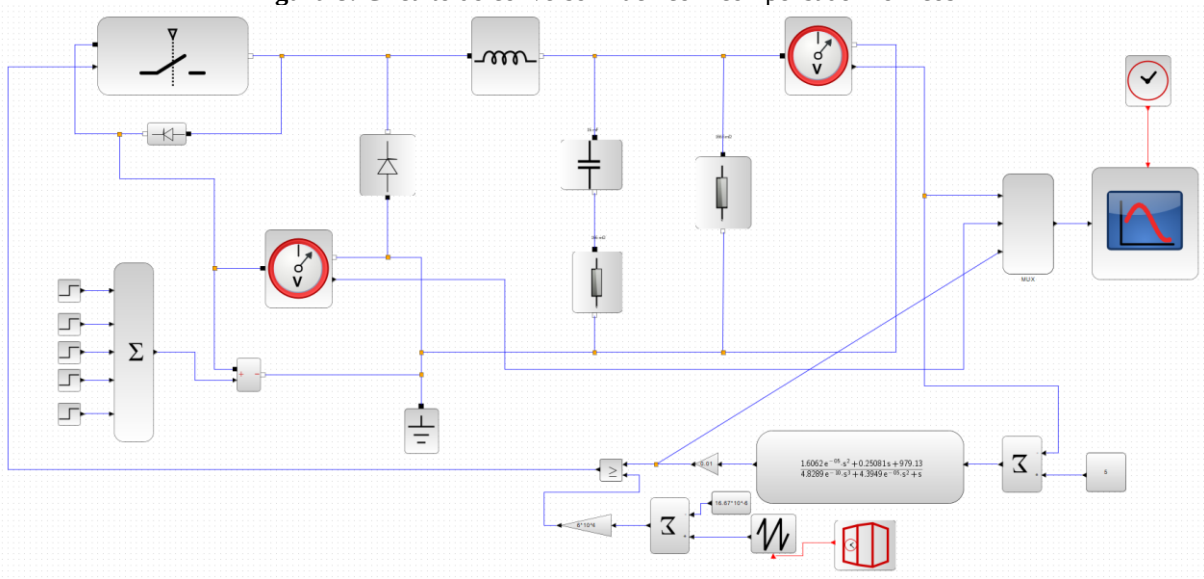
Figura 7: De cima para baixo, diagrama de Bode de *GpBuck*, *cc* e *FTMAC* no MATLAB



Fonte: elaborado pelos autores (2024).

O circuito completo em Xcos está na Figura 8, bem como a resposta a um trem de impulsos no Xcos (Fig. 9a) e no Simulink (Fig. 9b). O circuito no Simulink foi apresentado na Figura 1.

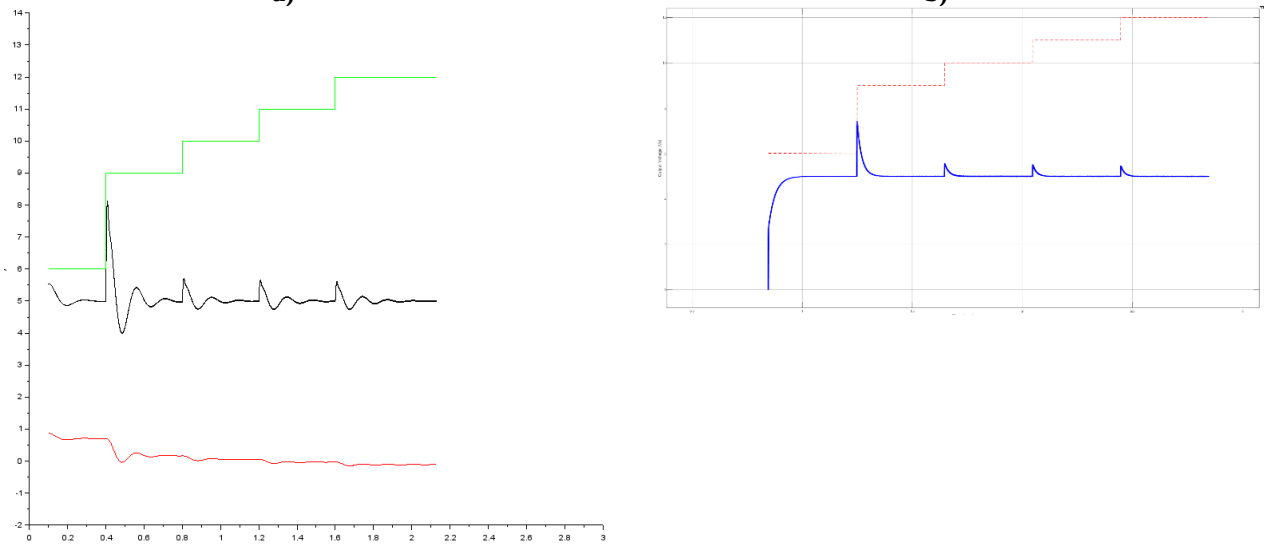
Figura 8: Circuito do conversor Buck com compensador no Xcos



Fonte: elaborado pelo autor (2024).

Figura 9: Aplicação de um trem de impulsos ao conversor Buck, a) implementação no Xcos em que verde indica o sinal de entrada, vermelho o sinal de saída do compensador e preto o sinal de saída do sistema e b) implementação no Simulink em que a linha pontilhada indica a entrada e a azul a saída do sistema

a) **b)**



Fonte: elaborado pelos autores (2024).

A diferença observada entre os sinais de saída do Xcos e do Simulink, muito provavelmente, decorrem do uso de um *switch* com diodo ideal no Xcos para substituir o MOSFET, e, portanto, sem elementos parasitas, o que no Xcos pode resultar em menor amortecimento e oscilações mais pronunciadas, levando a uma resposta subamortecida.

5 Considerações finais

Esse trabalho teve a proposta de fazer a conversão do modelo de um conversor Buck com compensador em MATLAB/Simulink para Scilab/Xcos. Apesar do Scilab ter uma documentação menor, menos didática e com menos exemplos de aplicação do que o MATLAB, o presente estudo demonstrou que é perfeitamente possível converter uma modelagem realizada em MATLAB e Simulink para Scilab e Xcos, obtendo quase todas as informações do modelo.

A única limitação encontrada, e não solucionada, foi a não obtenção da função de transferência do conversor com o compensador discretizada *cz* no Scilab (é possível obtê-la no MATLAB). Outro ponto é a falta do bloco MOSFET/IGPD, usado como chave no modelo em Simulink. No Xcos, foi substituído por um bloco Switch ideal e um diodo. A falta de elementos parasitas nessa chave ideal pode ter levado a um sinal subamortecido no Xcos, em comparação ao sinal com amortecimento crítico no Simulink.

Outra consideração importante é a configuração do Scilab logo após a instalação do programa. Para que um circuito compile no Xcos e possa ser simulado, o Scilab deve ter os pacotes MinGW gcc e MingGW, nessa ordem, instalados. Eles devem ser instalados a partir do gerenciador de módulos do Scilab, o ATOMS. O MinGW gcc está disponível por um link na descrição do módulo MinGW. Após a instalação do MinGW, é necessário reiniciar o Scilab e digitar, no console do Scilab, o comando *atomsInstall('mingw')*, reiniciando, mais uma vez, o programa.

Referências

CUSTÓDIO, E. S.; GONÇALVES, M. A.; PEDROSO, J. M. Conversor Buck chaveado com PWM com controle do tipo Fator K. *In*: ENCONTRO DE INICIAÇÃO CIENTÍFICA E FÓRUM CIENTÍFICO, 17., 2023, Curitiba. **Anais [...]**. Curitiba: Centro Universitário Internacional, 2023. DOI: 10.29327/1390731.17-8. Disponível em: https://www.even3.com.br/anais/enfocuninter2023/717602-conversor-buck-chaveado-com-pwm-com-controle-do-tipo-fator-k/. Acesso em: 09 out. 2024.

IBRAHIM, A. M. Free alternatives to MATLAB for undergraduate electronics engineering curricula. **IEEE**, Paris, p. 4598-4603, nov. 2006. DOI: 10.1109/IECON.2006.347594. Disponível em: https://ieeexplore.ieee.org/document/4153039. Acesso em: 09 Oct. 2024.

JANÍK, Z.; ŽÁKOVÁ, K. Online design of Matlab/Simulink and SciLab/Xcos block schemes. **IEEE**, Piešťany, 2011. DOI: 10.1109/ICL.2011.6059583. Disponível em: https://ieeexplore.ieee.org/document/6059583. Acesso em: 09 Oct. 2024.

MATLAB Documentation. **Mathworks**. 2024. Disponível em: https://www.mathworks.com/help/matlab/. Acesso em: 09 Oct. 2024.

PENDHARKAR, I. Rltool for Scilab: a public domain tool for SISO system design. **IEEE Control Systems Magazine**, Piscataway, v. 25, p. 23-25, 2005. DOI: 10.1109/MCS.2005.1388795. Disponível em: https://ieeexplore.ieee.org/abstract/document/1388795. Acesso em: 09 Oct. 2024.

SCILAB ENTERPRISES. **Scilab Documentation**. 2024. Disponível em: https://help.scilab.org/. Acesso em: 09 Oct. 2024.

SHAKIN, V. N. *et al.* Comparison of computer modeling of RC filter in Matlab and Scilab environments. **IEEE**, São Petersburgo, p. 1-5, 2020. DOI: 10.1109/WECONF48837.2020.9131473. Disponível em: https://ieeexplore.ieee.org/document/9131473. Acesso em: 09 Oct. 2024.

SILVA, E. M.; CUNHA, J. P. V. S. Scilab, Scicos e Rltool: softwares livres no ensino de engenharia elétrica. *In*: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 16., 2006, Salvador. **Anais [...]**. Salvador: Sociedade Brasileira de Automática, 2006. Disponível em: https://www.researchgate.net/publication/262006995_Scilab_Scicos_e_Rltool_em_softwares_livres_no_ensino_de_engenharia_eletrica. Acesso em: 09 Oct. 2024.