signals

MDPI

Article

Using Python for the Simulation of a Closed-Loop PI Controller for a Buck Converter

Acacio M. R. Amaral [1,*] and Antonio J. Marques Cardoso [2]

[1] CISE—Department of Informatics and Systems, Polytechnic Institute of Coimbra, ISEC, Quinta da Nora, P-3030-199 Coimbra, Portugal

[2] CISE—Electromechatronic Systems Research Center, University of Beira Interior, Calçada Fonte do Lameiro, P-6201-001 Covilhã, Portugal; ajmcardoso@ieee.org

* Correspondence: amramaral@ieee.org or acacio@isec.pt

Abstract: This paper presents a Python-based simulation technique that can be used to predict the behavior of switch-mode non-isolated (SMNI) DC-DC converters operating in closed loop. The proposed technique can be implemented in an open-source numerical computation software, such as Scilab, Octave or Python, which makes it versatile and portable. The software that will be used to implement the proposed technique is Python, since it is an open-source programming language, unlike MATLAB, which is one of most-used programming and numeric computing platforms to simulate this type of system. The proposed technique requires the discretization of the equations that govern the open-loop operation of the converter, as well as the discretization of the transfer function of the controller. To simplify the implementation of the simulation technique, the code must be subdivided into different modules, which together form a package. The converter under analysis will be a buck converter operating in CCM. The proposed technique can be extended to any other SMNI DC-DC converter. The validation of the proposed technique will be carried out by comparing it with the results obtained in LTspice.

Keywords: simulation; python; buck converter; CCM

check for updates

Citation: Amaral, A.M.R.; Cardoso, A.J.M. Using Python for the Simulation of a Closed-Loop PI Controller for a Buck Converter. *Signals* **2022**, *3*, 313–325. https://doi.org/10.3390/signals3020020

Academic Editors: Ioannis Dassios and Clemente Cesarano

Received: 3 April 2022
Accepted: 27 April 2022
Published: 20 May 2022

1. Introduction

Power electronics is a branch of electronics whose purpose is to develop systems that convert power from one form (source) to another form (load), as efficiently as possible. The aforementioned systems can be designed to operate in the power range from few W to MW, and can be found in home appliances, electric drives, renewable energy applications, electric trains, electric and hybrid cars and buses, industrial drives, and wireless power charging (smartphones and cars) [1], among others.

The aforementioned applications require different types of systems, which can be subdivided into four categories: AC/AC, AC/DC, DC/DC and DC/AC. In turn, each of these systems may have multiple circuits which can be used for a particular application. Usually, an application is composed of several stages which may contain different categories of power converters.

This paper focuses on the DC/DC converters, which are widely used in computer power supplies (ATX power supply) and servers, as well as in different types of charges and adapters (smartphones, tablet, laptops, etc.) that are used in our daily life.

The design of power converters requires the understanding of the working principle of the circuits and their analysis, but also the use of simulation tools.

One of the most commonly used tools to simulate power converters [2–11], as well as its components [12], is MATLAB/Simulink. The simplicity of the design of power converters, the high level of precision and user-friendly graphical interface makes this software very appealing [13,14], especially with regard to Simulink as it allows for Model-Based Design (MBD). MATLAB/Simulink plays a key role in carrying out research tasks [2–14].

One of the major disadvantages of MATLAB is the fact that it is a proprietary programming language unlike Python. As the algorithms implemented in MATLAB are proprietary, it is not possible to access their code. This particularity can significantly limit the implementation of control methods, as well as the development of fault diagnosis techniques [15].

Although Python does not allow MBD as MATLAB does, it has computational power and tools that make it possible to simulate DC/DC converters, as will be demonstrated in this paper.

On the other hand, the use of Python is clearly advantageous when applying machine-learning algorithms. Python has many libraries and packages for old and new machine-learning models, being widely used in this context in both industry and academia [16,17]. MATLAB has also some machine-learning algorithms that can provide immediate visual feedback, however these algorithms are proprietary.

The use of machine-learning algorithms can be very useful in converter design, in particular with regard to the control methods [7–9], and in the development of fault diagnosis techniques [18–21], making the proposed technique particularly suitable in these contexts.

The rest of this article is organized as follows. Section 2 describes the application of the proposed technique to a converter operating in open loop. Section 3 presents the converter design process and Section 4 shows how to apply the proposed technique to a converter operating in closed loop. Section 5 shows the feasibility of the proposed technique through its comparison with LTspice. Finally, Section 6 presents the conclusions.

2. Simulation of the Open-Loop Operation

The buck converter is suitable for IC (Integrated Circuit) power supplies, such as those used in Internet of Things (IoT) [22], rural DC micro-grids [23], fast charging of electric vehicles (EV) [24], solar photovoltaic (PV) systems [25].

Figure 1 shows the electric circuit of a buck converter.

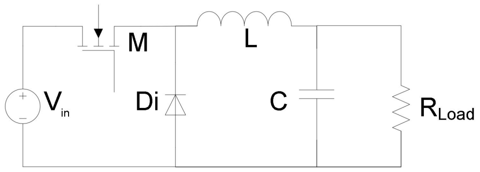


Figure 1. Buck converter circuit.

The buck converter (Figure 1) is composed of two semiconductors—a diode (Di) and a transistor (M), and two reactive elements—an inductor (L), and a capacitor (C). It can operate in two conduction modes: continuous conduction (CCM) and discontinuous conduction mode (DCM). The DCM will rise if the inductor current reaches a zero value.

In this paper it will be considered that the buck converter is operating in CCM, thus, only two stages will be analyzed: the conduction state and the non-conduction state.

2.1. Stage I—Conduction State

During the conduction stage, the transistor M turns on and the diode Di is reverse-biased, so the current will pass through the inductor L which stores energy in the form of a magnetic field; simultaneously, the capacitor C is charged (Figure 2).

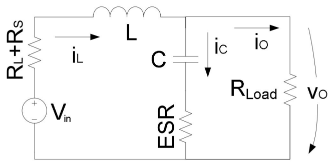


Figure 2. Buck converter circuit during the conduction state.

Through the analysis of the previous circuit, it is possible to write the following equations:

$$\begin{cases} \frac{dv_C}{dt} = \frac{i_L}{K_1} - \frac{v_C}{R_{Load} \times K_1}, \\ K_1 = \left(C + \frac{C \times ESR}{R_{Load}}\right) \\ \frac{di_L}{dt} = \frac{Vin}{L} + i_L \times KA + v_C \times KB, \\ K_A = \left(-\frac{R_S + R_L}{L} - \frac{ESR \times C}{K_1 \times L}\right), K_B = \left(-\frac{1}{L} + \frac{ESR \times C}{R_{Load} \times K_1 \times L}\right) \end{cases} \tag{1}$$

2.2. Stage II—Non-Conduction State

During the non-conduction state, the transistor is turned off, which forces the current to free wheel around the path consisting of L, C and Di (Figure 3).

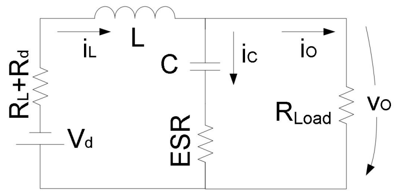


Figure 3. Buck converter circuit during the non-conduction state.

Through the analysis of the previous circuit, it is possible to write the following equations:

$$\begin{cases} \frac{dv_C}{dt} = \frac{i_L}{K_1} - \frac{v_C}{R_{Load} \times K_1}, \\ K_1 = \left(C + \frac{C \times ESR}{R_{Load}}\right) \\ \frac{di_L}{dt} = -\frac{Vd}{L} + i_L \times KA + v_C \times KB, \\ K_A = \left(-\frac{R_d + R_L}{L} - \frac{ESR \times C}{K_1 \times L}\right), K_B = \left(-\frac{1}{L} + \frac{ESR \times C}{R_{Load} \times K_1 \times L}\right) \end{cases} \tag{2}$$

where V_{in}, R_L, L, R_S, V_d, R_d, C, ESR, R_{Load}, i_L, i_C, i_O, v_O and v_C, represent the input voltage, inductor resistance, inductor inductance, drain-source resistance, diode forward voltage drop, diode internal resistance, capacitor capacitance, capacitor internal resistance, load resistance, inductor current, capacitor current, load current, output voltage and output voltage component due exclusively to C, respectively.

Finally, the following set of equations can be presented, that apply to both states.

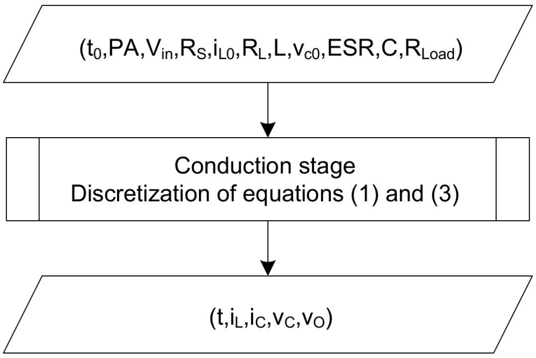
$$\begin{cases} i_C = C \times \frac{dv_C}{dt} \\ v_O = ESR \times i_C + v_C \end{cases} \tag{3}$$

2.3. Open-Loop Simulation

As previously mentioned, the simulation technique will be implemented in Python. Thus, it is important to briefly introduce the main tools that will be used in this context.

Python is a high-level, general-purpose programming language that can be used to develop numerical computation applications, such as simulation of physical systems. For this purpose, it is essential to use some Python libraries. The most important ones, in this context, are NumPy and Matplotlib. The Numerical Python package (Numpy) is used to manipulate arrays and matrices, and the Matplotlib package is used for plotting.

To simplify the implementation of the simulation technique, the code must be subdivided into different modules, which together form a package. The first module must contain two functions that compute i_L, i_C, v_C and v_O values during the conduction state (Figure 4) and during the non-conducting state (Figure 5).

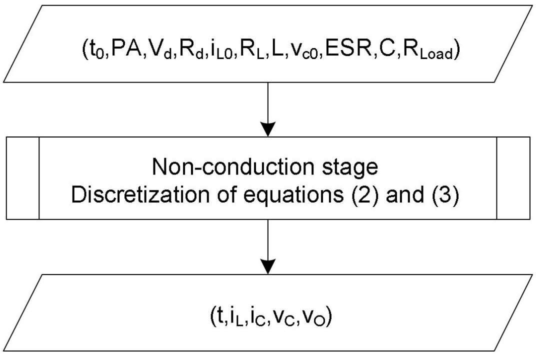

(t_0,PA,V_{in},R_S,i_{L0},R_L,L,v_{c0},ESR,C,R_{Load})
Conduction stage
Discretization of equations (1) and (3)
(t,i_L,i_C,v_C,v_O)

(a)

```
def conduction_state(t0,PA,Vin,Rs,il0,Rl,L,vc0,ESR,C,R):
    K1 = C + ESR*C/R
    KA = - ESR*C/(K1*L) - (Rl+Rs)/L
    KB = ESR*C/(R*K1*L) - 1/L
    vc = vc0*(1 - PA/(R*K1)) + il0*(PA/K1)
    il = Vin*PA/L + il0*(1 + KA*PA) + vc0*(KB*PA)
    ic = C*(vc - vc0)/PA
    vo = ESR*ic + vc
    t = t0 + PA
    return [t,il,ic,vc,vo]
```

(b)

Figure 4. Function that models the behavior of the converter during the on state: (**a**) Algorithm and (**b**) Python code.


(t_0,PA,V_d,R_d,i_{L0},R_L,L,v_{c0},ESR,C,R_{Load})
Non-conduction stage
Discretization of equations (2) and (3)
(t,i_L,i_C,v_C,v_O)

(a)

```
def non_conducting_state(t0,PA,Vd,Rd,il0,Rl,L,vc0,ESR,C,R):
    K1 = C + ESR*C/R
    KA = - ESR*C/(K1*L) - (Rl + Rd)/L
    KB = ESR*C/(R*K1*L) - 1/L
    vc = vc0*(1 - PA/(R*K1)) + il0*(PA/K1)
    il = - Vd*PA/L + il0*(1 + KA*PA) + vc0*(KB*PA)
    ic = C*(vc - vc0)/PA
    vo = ESR*ic + vc
    t = t0 + PA
    return [t,il,ic,vc,vo]
```

(b)

Figure 5. Function that models the behavior of the converter during the off state: (**a**) Algorithm and (**b**) Python code.

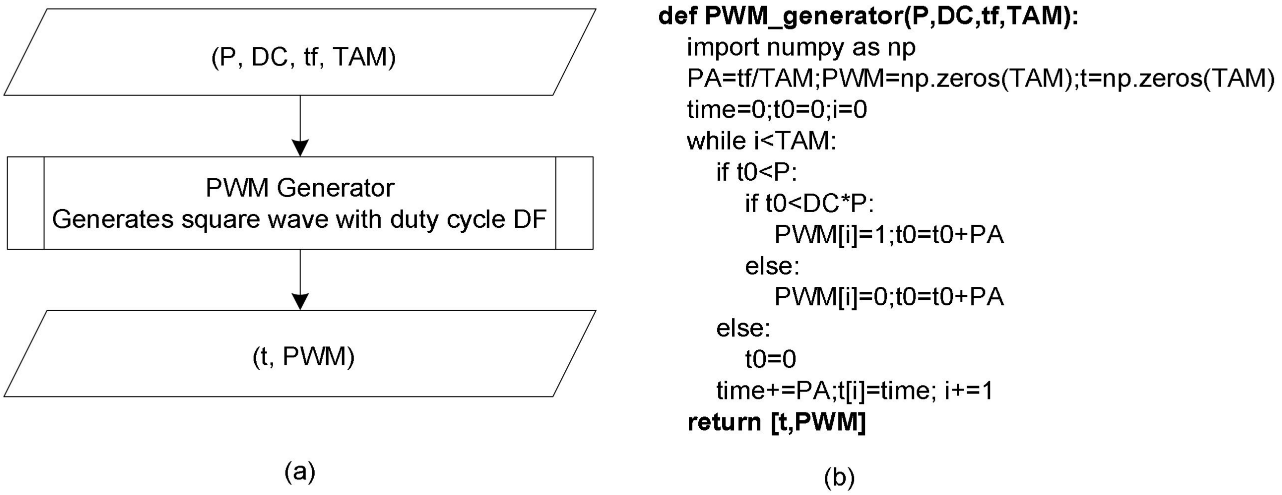
Thus, after discretizing Equation (1), it is possible to construct a function that describes the operation of the converter during the conduction state (Figure 4). This function must receive as input arguments the initial time, t_0, the sampling period, PA, the characteristics of the converter (V_{in}, R_S, L, ESR, C and R_{Load}) and the initial conditions (the inductor current, i_{L0}, and capacitor voltage, v_{C0}, at the initial time).

Following this, the same procedure must be applied for the non-conduction state, in order to obtain the non-conducting state function (Figure 5).

The previous function (Figure 5) takes practically the same input arguments as the function in Figure 4. The difference between the two functions is related to the input arguments, as can be seen by comparing Equations (1) and (2).

After creating the module which reproduces the conduction and non-conducting states, it is necessary to create a new one: the module that will be responsible for generating the output signal of the Pulse Width Modulator (PWM). This module will be composed of a single function that receives as input arguments the converter switching period (P), the duty cycle (DC), the final simulation time (tf) and the total number of iterations (TAM). In turn, the function returns the time vector, as well as a square waveform with period P and duty cycle DC.

Finally, it is necessary to create a new module that contains the main function. The main function needs to import the previous modules (Figures 4–6), as well as the Numpy and Matplotlib packages, so that it can simulate the buck converter.



```
def PWM_generator(P,DC,tf,TAM):
  import numpy as np
  PA=tf/TAM;PWM=np.zeros(TAM);t=np.zeros(TAM)
  time=0;t0=0;i=0
  while i<TAM:
    if t0<P:
      if t0<DC*P:
        PWM[i]=1;t0=t0+PA
      else:
        PWM[i]=0;t0=t0+PA
    else:
      t0=0
    time+=PA;t[i]=time; i+=1
  return [t,PWM]
```

Figure 6. Function that generates the output PWM signal: (**a**) Algorithm and (**b**) Python code.

Figure 7 shows the main function flowchart.

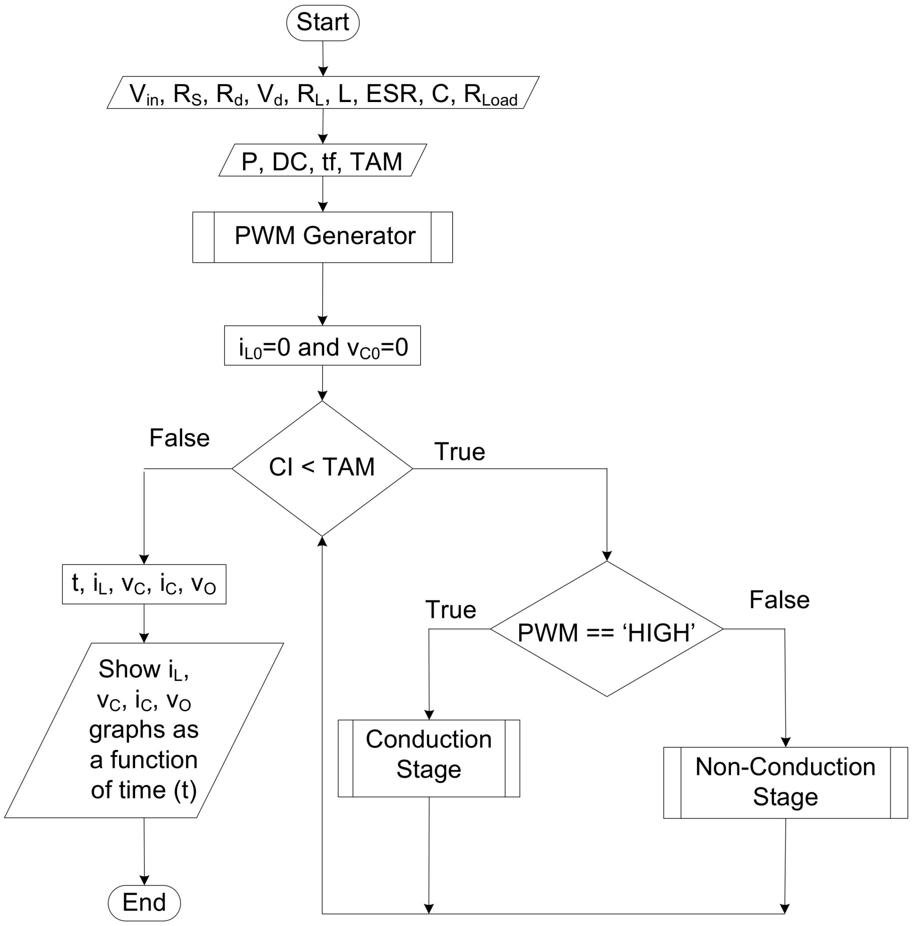

Start
V_{in}, R_S, R_d, V_d, R_L, L, ESR, C, R_{Load}
P, DC, tf, TAM
PWM Generator
i_{L0}=0 and v_{C0}=0
False
CI < TAM
True
t, i_L, v_C, i_C, v_O
True
PWM == 'HIGH'
False
Show i_L, v_C, i_C, v_O graphs as a function of time (t)
Conduction Stage
Non-Conduction Stage
End

Figure 7. Main function flowchart. The conduction stage, non-conduction stage and PWM generator functions are represented in Figures 4–6, respectively.

CI represents the current iteration and t, i_L, v_C, i_C and v_O represent the vectors referring to the following quantities: time, inductor current, output voltage component due exclusively to C, capacitor current and output voltage, respectively.

3. Power-Stage Design Process

To obtain design formulas for the buck converter power section, it is necessary to consider the following rules:

- Converter is operating in steady-state regime (V_{in} > V_O);
- Average voltage across inductor is zero ($\langle v_L \rangle = 0$);
- Average current through capacitor is zero ($\langle i_C \rangle = 0$);
- All components are considered ideal;
- Capacitor capacitance is large enough.

Considering the converter conduction state (Figure 2), it is possible to write the following equations:

$$\begin{cases} v_{LON} = L \times \frac{di_L}{dt} \Rightarrow L \times \frac{\Delta i_L}{D \times T} = V_{in} - V_O \\ v_M = 0; v_D = V_{in}; i_M = i_L; i_d = 0 \end{cases}$$
$$\Rightarrow \quad \Delta i_{LON} = \frac{V_{in} - V_O}{L \times f} \times D;\ v_M = 0;\ v_D = V_{in};\ i_M = i_L;\ i_d = 0 \tag{4}$$

where V_{in}, V_O, v_{LON}, Δi_L, D, T, f, v_M, i_M, i_D and v_D represent the input voltage, mean value of output voltage, voltage across the inductor during the conduction state, inductor current ripple, duty cycle, switching period, switching frequency, MOSFET voltage, MOSFET current, diode current and diode voltage, respectively.

Considering the non-conduction state (Figure 3), it is possible to write the following equations:

$$\begin{cases} v_{LOFF} = L \times \frac{\Delta i_L}{(1-D) \times T} = -V_O \\ v_M = Vin; v_D = 0; i_M = 0; i_d = i_L \end{cases}$$
$$\Rightarrow \quad \Delta i_{LOFF} = \frac{V_O}{L \times f} \times (1 - D);\ v_M = V_{in};\ v_D = 0;\ i_M = 0;\ i_d = i_L \tag{5}$$

From the previous equations, it is possible to compute the Di and M ratings:

- Maximum voltage value across both Di and M is equal to V_{in} (minimum voltage rating).
- Peak current through Di and M is equal to maximum inductor peak current (minimum current rating).

The inductor peak current can be calculated as follows:

$$i_L = \langle i_L \rangle + \frac{\Delta iL}{2} \tag{6}$$

Assuming that the average voltage across the inductor is zero, it is possible to write:

$$\langle V_L \rangle = \frac{(V_{in} - V_O) \times D \times T + (-V_O) \times (1-D) \times T}{T} = 0$$
$$\Rightarrow \langle V_L \rangle = V_{in} \times D - V_O = 0 \Rightarrow \ V_O = V_{in} \times D \tag{7}$$

Assuming that the average value of the capacitor current is zero and neglecting the ESR effect (ESR << R_{Load}), it is possible to conclude that the AC component of the inductor current is equal to the capacitor current. In this way, it is possible to write:

$$i_C = C \times \frac{dv_C}{dt} \Rightarrow \frac{dq}{dt} = C \times \frac{dv_C}{dt} \Rightarrow \Delta q = C \times \Delta v_C$$
$$\Rightarrow \begin{cases} \Delta v_C = \frac{\Delta q}{C} \Rightarrow \Delta v_C = \frac{1}{2} \times \frac{\Delta i_L}{2} \times \frac{T}{2} \times \frac{1}{C} \\ \Delta q = \frac{\Delta i_L}{2} \times \frac{T}{2} \times \frac{1}{2} \end{cases} \tag{8}$$

Using Equation (5), it is possible to rewrite Equation (8) as follows

$$\Delta v_C = \frac{V_O \times (1 - D)}{8 \times L \times f^2 \times C} \tag{9}$$

Equation (9) is commonly used to choose the capacitor capacitance. Following this, formulas that allow for the computation of the inductor inductance will be presented.

The critical inductance value takes into account the bound condition between the CCM and the DCM. The above condition occurs when the average inductor current is equal to half the ripple of the inductor current. Thus, using Equation (4), it is possible to calculate the critical value of L ($L_{critical}$) as:

$$\begin{aligned} &\langle i_L \rangle = \frac{V_O}{R} = \frac{\Delta i_{LON}}{2} = \frac{V_{in} - V_O}{2 \times L_{critical} \times f} \times D \\ &\Rightarrow L_{critical} = \frac{1 - D}{2 \times f} \times R \end{aligned} \tag{10}$$

In DCM, the relationship obtained in Equation (7) no longer holds, which makes the control system more complex.

If the converter operates in CCM, as the circuit under analysis, it is usual to specify a limit to the ripple value of i_L (ΔiL_{Max}). In this case, Equation (5) can be used.

Following this, the passive components of the converter power section will be chosen considering Equations (5), (7) and (9). For this purpose, the following specifications (Table 1) will be used.

Table 1. Power section specifications.

Input Voltage	**Output Voltage**	**Load Range**	**Output Voltage Ripple (Δv_O)**	**Inductor Ripple Current (Δi_L)**	**Operating Frequency**
19 V	5 V	5 W to 50 W	5 mV	40% of I_O	100 kHz

At first, it is important to calculate the duty cycle; so using Equation (7), it is possible to compute D:

$$D = \frac{V_O}{V_{in}} = \frac{5}{19} \cong 0.26$$

The inductor ripple will be greater when the load current is minimum. Thus, the minimum L value can be computed as follows, using Equation (5):

$$L = \frac{V_O}{\Delta i_L \times f} \times (1 - D) \Rightarrow \begin{cases} L > \frac{5}{\Delta i_L \times 100k} \times (1 - 0.26) \\ \Delta i_L = 0.4 \times I_O = 0.4 \times \frac{\min(P_O)}{V_O} = 0.4 \end{cases} \Rightarrow L > 92.5\ \mu H$$

Finally, it is possible to calculate C through Equation (9):

$$C > \frac{5 \times (1 - 0.26)}{8 \times (92.5\mu) \times (100k)^2 \times (5m)} = 100\ \mu F$$

Considering the above conditions, the component tolerance and a safety margin, it was decided to choose the following passive elements:

$$L = 200\ \mu H \text{ and } C = 220\ \mu F$$

Since the capacitance value is relatively high, an Alcap capacitor was chosen. As the ESR value of a 220 µF Alcap is in the order of 0.2 Ω at 1 kHz [26], it is possible to conclude that the output voltage ripple will be essentially the result of the ESR:

$$\Delta v_O \cong ESR \times \Delta i_L \cong 0.2 \times \Delta i_L \tag{11}$$

4. Simulation of the Closed-Loop Operation

In this section, the implementation of the proposed technique for a converter operating in closed loop will be shown.

At first, it is important to analyze the control circuit (Figure 8).

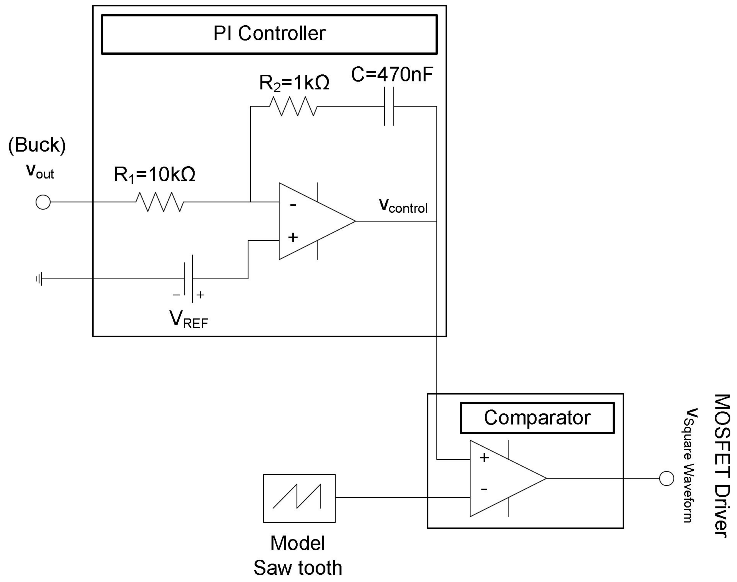


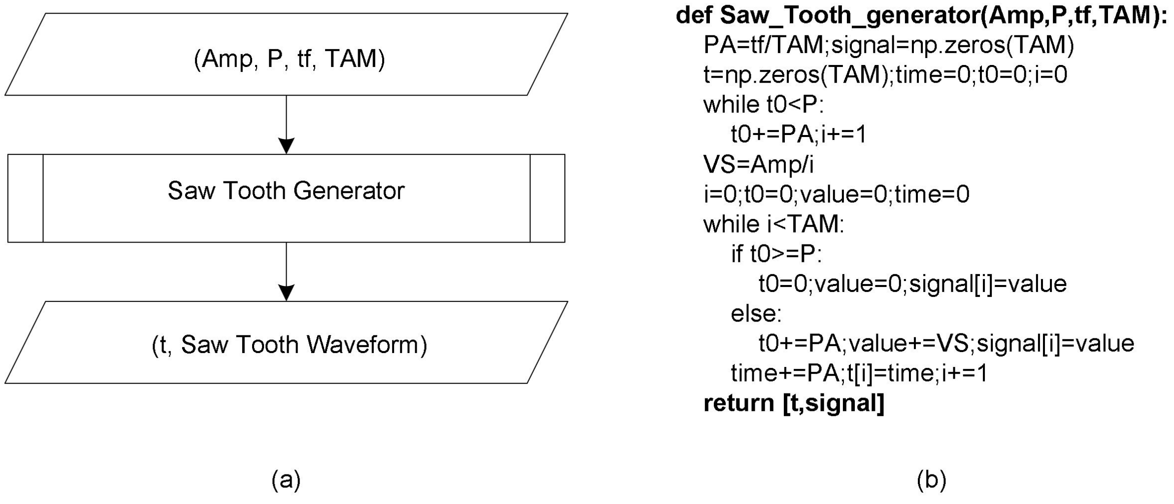
Figure 8. Schematics of the control circuit.

As can be seen in Figure 8, the control circuit is composed of three fundamental elements: the PI controller, the Comparator and the Saw Tooth Generator.

To implement the PI controller in Python it is necessary to discretize its transfer function Equation (12):

$$i_C = \frac{V_{ref} - v_{out}}{R_1} \wedge v_{control} = v_c + R_2 \times \left(\frac{V_{ref} - v_{out}}{R_1} \right) + V_{ref} \tag{12}$$

The saw tooth generator must be implemented in a specific function (Figure 9). The function should receive as input arguments the amplitude of the Saw Tooth waveform (Amp), the period (P), the final simulation time (tf) and the total number of iterations, TAM. In turn, the function returns the time vector, as well as a saw tooth waveform with period P.



```
def Saw_Tooth_generator(Amp,P,tf,TAM):
    PA=tf/TAM;signal=np.zeros(TAM)
    t=np.zeros(TAM);time=0;t0=0;i=0
    while t0<P:
        t0+=PA;i+=1
    VS=Amp/i
    i=0;t0=0;value=0;time=0
    while i<TAM:
        if t0>=P:
            t0=0;value=0;signal[i]=value
        else:
            t0+=PA;value+=VS;signal[i]=value
        time+=PA;t[i]=time;i+=1
    return [t,signal]
```

Figure 9. Function that generates the Saw Tooth Waveform: (**a**) Algorithm and (**b**) Python code.

The comparator can be implemented through a simple control-of-flow statement (if . . . elif . . . else). The previous instruction must take into account the possibility of op-amp saturation.

Finally, the previous functions must be properly included into the algorithm presented in Figure 7.

5. Comparison—Proposed Technique versus LTspice

To show the applicability of the proposed simulation technique, a comparison of the proposed technique with LTspice will be presented in this section.

Figure 10 shows the implementation of the circuit under analysis in LTspice.

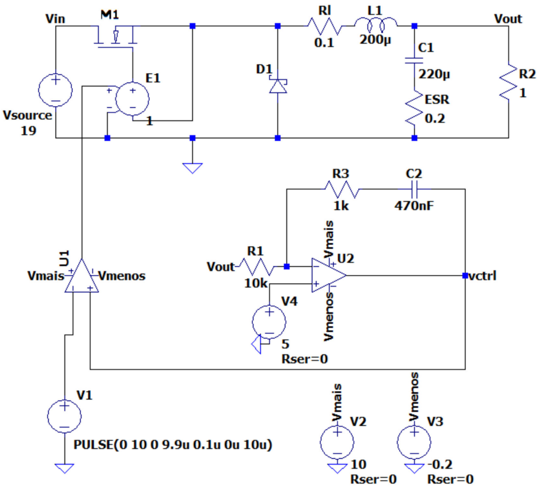


Figure 10. Circuit implementation in LTspice.

To assess the feasibility of the proposed simulation technique, four different operating conditions are considered: steady state (Figure 11), start up (Figure 12), load variation (Figure 13) and input voltage variation (Figure 14).

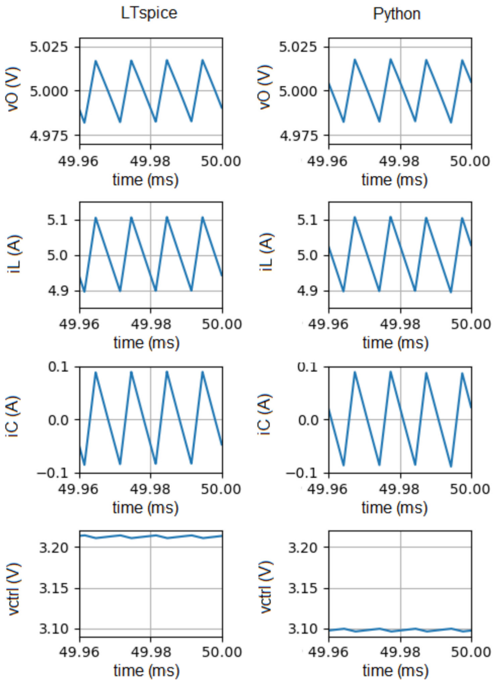


Figure 11. Waveforms of v_O, i_L, i_C and v_{ctrl} obtained in steady-state regime, with LTspice (**left plots**) and with the proposed simulation technique (**right plots**).

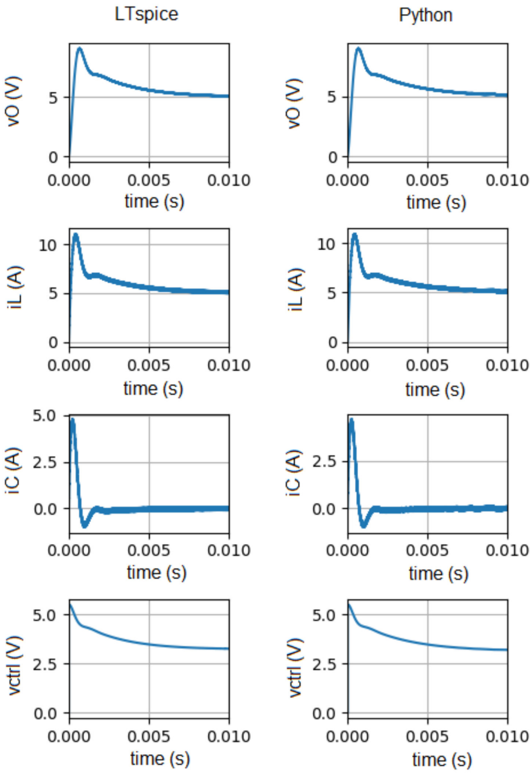


Figure 12. Waveforms of v_O, i_L, i_C and v_{ctrl} obtained during the converter start up, with LTspice (**left plots**) and with the proposed simulation technique (**right plots**).

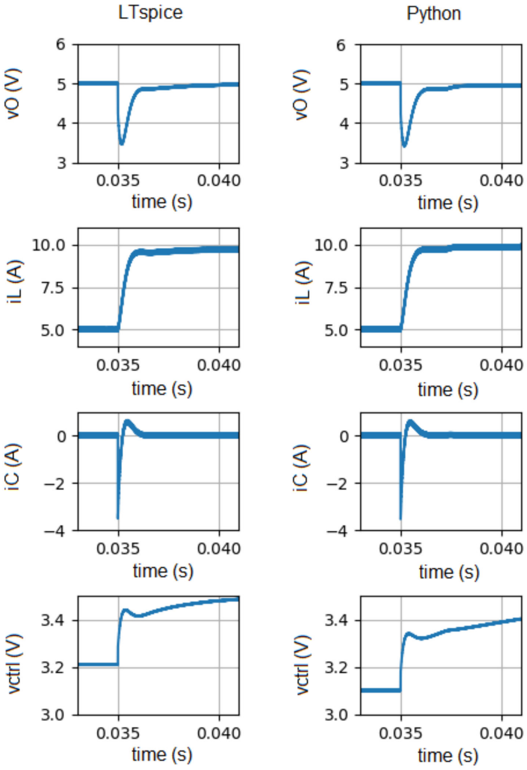


Figure 13. Waveforms of v_O, i_L, i_C and v_{ctrl} obtained during a load variation (from 5 A to 10 A) at 35 ms, with LTspice (**left plots**) and with the proposed simulation technique (**right plots**).

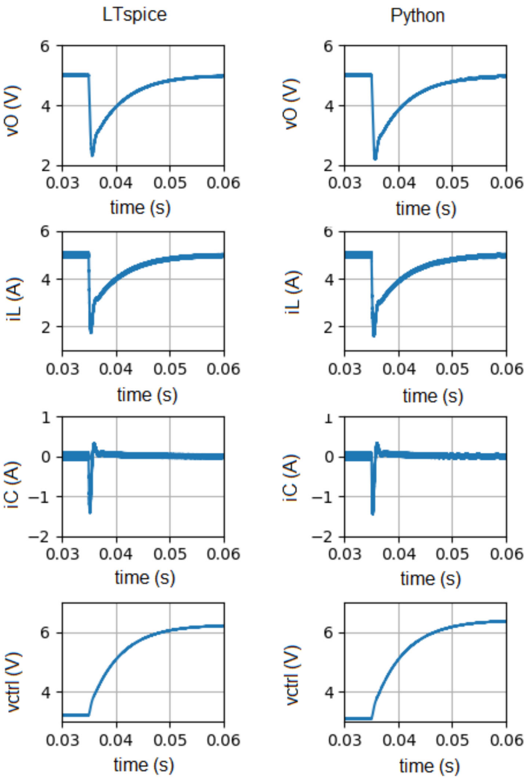


Figure 14. Waveforms of v_O, i_L, i_C and v_{ctrl} obtained during an input voltage variation (from 19 V to 9 V) at 35 ms, with LTspice (**left plots**) and with the proposed simulation technique (**right plots**).

Figures 11–14 show the time evolution of v_O, i_L, i_C, and v_{ctrl} obtained in LTspice (left plots) and also with the proposed simulation technique (right plots).

Figure 11 shows the waveforms related to the steady state regime.

Next, the results related to converter start up (Figure 12) will be presented.

Figure 13 shows the behavior of the converter when subjected to a load variation (from 5 A to 10 A at 35 ms).

Figure 14 shows the behavior of the converter when subjected to an input voltage variation (from 19 V to 9 V at 35 ms).

Figures 11–14 show the waveforms of the output voltage (v_O), inductor current (i_L), capacitor current (i_C) and control voltage (v_{ctrl}) obtained with the proposed technique and in LTspice. v_{ctrl} shows a slight difference between both simulations, which is essentially due to the time taken by the MOSFET to switch. This time period is not taken into account in Python simulation.

In summary, it is possible to conclude that the results of both simulators are quite similar. Thus, it can be concluded that the proposed solution can be used profitably to predict the behavior of DC-DC converters.

6. Conclusions

In this paper, a Python-based simulation technique for SMNI DC-DC converters, operating in closed loop, was presented. It was demonstrated that the proposed simulation technique can be used to predict the behavior of DC-DC converters, and for that purpose, LTspice was used to validate the results. This conclusion can be corroborated when a finer comparison is made between the waveforms obtained in LTspice and the proposed technique. Thus, regarding the average value, the maximum error is 0.01%, and as far as ripple is concerned, the maximum error is 1.8%. The errors are essentially due to the sampling period, which in the case of LTspice, is variable (variable-step). In the previous analysis, the control signal was not considered because in the simulation performed in Python, the time taken by the MOSFET to switch was not considered.

MATLAB/Simulink is one of the most commonly used pieces of software in the simulation of power converters, However, since it is a commercial software, the algorithms implemented in MATLAB are proprietary. This characteristic makes the implementation of control methods and the development of fault diagnosis techniques difficult.

The major advantage of the proposed solution is that it can be implemented in any open-source numerical computational software, making it versatile and portable. Therefore, in the authors' opinion, the proposed simulation technique makes the implementation of control methods, as well as the development of fault diagnosis techniques in DC-DC converters, much simpler.

On the other hand, an algorithm was presented that can be used in the selection of the components that compose the power section of the converter.

Author Contributions: Conceptualization, A.M.R.A.; methodology, A.M.R.A.; software, A.M.R.A.; validation, A.M.R.A.; formal analysis, A.M.R.A. and A.J.M.C.; investigation, A.M.R.A. and A.J.M.C.; resources, A.M.R.A.; data curation, A.M.R.A.; writing—original draft preparation, A.M.R.A.; writing—review and editing, A.M.R.A. and A.J.M.C.; visualization, A.M.R.A. and A.J.M.C.; supervision, A.M.R.A. and A.J.M.C.; project administration, A.M.R.A. and A.J.M.C.; funding acquisition, A.M.R.A. and A.J.M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the European Regional Development Fund (ERDF) through the operational program for competitiveness and internationalization (COMPETE 2020), under project POCI-01-0145-FEDER-029494 and by national funds through the FCT—Portuguese Foundation for Science and Technology, under projects PTDC/EEI-EEE/29494/2017, UIDB/04131/2020 and UIDP/04131/2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jung, G.; Hosani, K.; Song, B.; Seo, D.; Kim, J.; Cho, D. Semi-Dynamic Wireless Power Charging System for Autonomous Electric Vehicle. In Proceedings of the 2021 IEEE Wireless Power Transfer Conference, San Diego, CA, USA, 1–4 June 2021.
2. Mohammad, K.; Rashid, M.F.; Rahat, H.; Khan, F.; Rahman, K. Detailed Analysis of DC-DC Converters Fed with Solar-PV System with MPPT. In Proceedings of the 2022 International Conference for Advancement in Technology, Goa, India, 21–22 January 2022.
3. Dias, N.; Naik, A.J. Design, Modeling and Simulation of Bidirectional Buck and Boost Converter for Electric Vehicles. In Proceedings of the 2022 International Conference for Advancement in Technology, Goa, India, 21–22 January 2022.
4. Jeelani, N.; Bhat, A.H.; Mir, T.N. Comparative Analysis of Voltage Control Techniques for Single Phase AC-AC Buck Converter. In Proceedings of the 2022 IEEE International Conference on Power Electronics, Smart Grid, and Renewable Energy, Trivandrum, India, 2–5 January 2022.
5. Chapparya, V.; Singh, S.P.; Dey, A. Novel Non-isolated Boost-Zeta Interleaved DC-DC Converter for Low Voltage Bipolar DC Micro-grid Application. In Proceedings of the 2022 IEEE International Conference on Power Electronics, Smart Grid, and Renewable Energy, Trivandrum, India, 2–5 January 2022.
6. Stala, R.; Waradzyn, Z.; Folmer, S. Input Current Ripple Reduction in a Step-Up DC–DC Switched-Capacitor Switched-Inductor Converter. *IEEE Access* **2022**, *10*, 19890–19904. [CrossRef]
7. Andries, V.; Goras, L.; Buzo, A.; Pelz, G. Automatic tuning for a DC-DC Buck Converter with adaptive controller. In Proceedings of the 2017 International Symposium on Signals, Circuits and Systems, Iasi, Romania, 13–14 July 2017.
8. Prag, K.; Woolway, M.; Celik, T. Data-Driven Model Predictive Control of DC-to-DC Buck-Boost Converter. *IEEE Access* **2021**, *9*, 101902–101915. [CrossRef]
9. Andino, J.; Ayala, P.; Llanos-Proaño, J.; Naunay, D.; Martinez, W.; Arcos-Aviles, D. Constrained Modulated Model Predictive Control for a Three-Phase Three-Level Voltage Source Inverter. *IEEE Access* **2022**, *10*, 10673–10687. [CrossRef]
10. Nupur, N.; Nath, S. Effect of Coupling on Discontinuous Conduction Mode of Coupled Inductor SIDO Boost Converter. *IEEE Trans. Power Electron.* **2022**, *37*, 4991–5002. [CrossRef]
11. Navamani, J.D.; Geetha, A.; Almakhles, D.; Lavanya, A.; Ali, J.S. Modified LUO High Gain DC-DC Converter With Minimal Capacitor Stress for Electric Vehicle Application. *IEEE Access* **2021**, *9*, 122335–122350. [CrossRef]

12. Castiglia, V.; Campagna, N.; Tommaso, A.; Miceli, R.; Nevoloso, C.; Pellitteri, F.; Puccio, C.; Viola, F. Modeling, Simulation, and Characterization of a Supercapacitor in Automotive Applications. *IEEE Trans. Ind. Appl.* **2022**, *58*, 2421–2429. [CrossRef]
13. Khader, S.; Hadad, A.; Abu-aisheh, A.A. The application of PSIM & Matlab/Simulink in power electronics courses. In Proceedings of the 2011 IEEE Global Engineering Education Conference, Amman, Jordan, 4–6 April 2011.
14. Showers, O.; Raji, A. Modelling and Simulation of Fuel Cell Hybrid Electric Vehicle Powertrain. In Proceedings of the 30th Southern African Universities Power Engineering Conference, Durban, South Africa, 25–27 January 2022.
15. Ranjani, J.; Sheela, A.; Meena, K.P. Combination of NumPy, SciPy and Matplotlib/Pylab-a good alternative methodology to MATLAB-A Comparative analysis. In Proceedings of the 1st International Conference on Innovations in Information and Communication Technology, Chennai, India, 25–26 April 2019.
16. Janarthanam, K.; Kamalesk, P.; Basil, V.; Judeson, J. Electrical Faults-Detection and Classification using Machine Learning. In Proceedings of the International Conference on Electronics and Renewable Systems, Tuticorin, India, 16–18 March 2022.
17. Khond, S.; Kale, V.; Ballal, M. A combined Data Mining and Machine Learning approach for High Impedance Fault Detection in Micrograids. In Proceedings of the IEEE International Conference on Power Electronics, Smart Grid, and Renewable Energy, Trivandrum, India, 2–5 January 2022.
18. Sun, Q.; Wang, Y.; Jiang, Y. A Novel Fault Diagnostic Approach for DC-DC Converters Based on CSA-DBN. *IEEE Access* **2018**, *6*, 6273–6285. [CrossRef]
19. Amaral, A.; Sahraqui, M. Capacitors. In *Diagnosis and Fault Tolerance of Electrical Machines Power Electronics and Drives*; Chapter 5; Cardoso, A.J.M., Ed.; IET-The Institution of Engineering and Technology: London, UK, 2018; pp. 195–286.
20. Xia, Y.; Xu, Y. A Transferrable Data-Driven Method for IGBT Open-Circuit Fault Diagnosis in Three-Phase Inverters. *IEEE Trans. Power Electron.* **2021**, *36*, 13478–13488. [CrossRef]
21. Li, Z.; Gao, Y.; Zhang, X.; Wang, B.; Ma, H. A Model-Data-Hybrid-Driven Diagnosis Method for Open-Switch Faults in Power Converters. *IEEE Trans. Power Electron.* **2021**, *36*, 4965–4970. [CrossRef]
22. Zhao, M.; Li, M.; Song, S.; Hu, Y.; Yao, Y.; Bai, X.; Hu, R.; Wu, X.; Tan, Z. An Ultra-Low Quiescent Current Tri-Mode DC-DC Buck Converter With 92.1% Peak Efficiency for IoT Application. *IEEE Trans. Circuits Syst.* **2022**, *69*, 428–439. [CrossRef]
23. Malik, M.S.; Khan, H.A.; Zaffar, N.A. Evaluation of a Single Inductor based Single-Input Dual-Output Buck Converter for DC Microgrid Applications. In Proceedings of the IEEE 7th World Conference on Photovoltaic Energy Conversion, Waikoloa, HI, USA, 10–15 June 2018.
24. Garg, A.; Das, M. High Efficiency Three Phase Interleaved Buck Converter for Fast Charging of EV. In Proceedings of the 1st International Conference on Power Electronics and Energy, Bhubaneswar, India, 2–3 January 2021.
25. Venkatramanan, D.; John, V. Dynamic Modeling and Analysis of Buck Converter Based Solar PV Charge Controller for Improved MPPT Performance. *IEEE Trans. Ind. Appl.* **2019**, *55*, 6234–6246. [CrossRef]
26. Amaral, A.M.R.; Cardoso, A.J.M. A Simple Offline Technique for Evaluating the Condition of Aluminum–Electrolytic–Capacitors. *IEEE Trans. Ind. Electron.* **2009**, *56*, 3230–3237. [CrossRef]