PyAMS: A New Software for Modeling Analog Elements and Circuit Simulations

Fathi Dhiabi[1], M. Larbi Megherbi[1], Achour Saadoune[1], Riccardo Carotenuto[2], and Fortunato Pezzimenti[2]
[1] Department of Electrical Engineering, Biskra University, Biskra, Algeria
[2] DIIES-, Mediterranea University of Reggio Calabria, Reggio Calabria, Italy
Email: {f.dhiabi; mohamed.megherbi; a.saadoune}@univ-biskra.dz; {r.carotenuto; fortunato.pezzimenti}@unirc.it

***Abstract*—All technology instruments use electrical and electronic systems that, before their production, need to be verified via simulation software. A new simulation software called Python for Analog and Mixed Signals (PyAMS) has been programmed. As presented in this paper, the main objective of this software is to simplify the modeling of analog elements and circuits by using the python language to describe design schematics involving libraries, packages, and symbols. PyAMS would be a free software (GNU license). The circuit simulation in PyAMS allows a detailed frequency-domain analysis, DC analysis, and time-domain analysis. The output signals are acquired in different operating points and they are displayed by means of a dedicated waveform editor. The behavioral modeling of analog elements and the simulations results of different test circuits are reported in the text.**

***Index Terms*—Analog circuits, spice, PyAMS, modeling, newton raphson, CAD system**

I. Introduction

With the rapid development of devices technology, the design of electrical circuits is becoming more and more computer-dependent. The circuit outputs cannot be determined by manual calculations which lead to a high probability of error and are too many time consuming. The increasing complexity of circuits makes the use of Computer Aided Design (CAD) tool an advanced methodology playing a key role in this field. Nowadays, for example, Spice is largely used by designers [1]-[3]. In fact, after more than four decades of operation with different generations of computers in simulating critical problems in highly complex circuits, Spice is considered the best circuit simulator around the world. Also, Spice inspires different commercial and free electronic design automation (EDA) programs, such as Qucs, XSPICE, Xyce, NgSPICE, LTSpice, and OPUS [4]-[10]. They all utilize the Spice core, namely the Newton-Raphson (NR) iterative algorithm with the standard sparse solving method [3], [11], [12] to find the operating points of nonlinear circuits.

The NR algorithm is the best numerical method applied in circuit analysis due to its principle of converting nonlinear elements to linear elements depending on Newton's iteration [3], [13]-[15]. However, the NR method does not always converge to a solution.

Manuscript received December 16, 2020; revised March 25 2021; accepted March 30, 2021.
Corresponding author: Fathi Dhiabi (email: f.dhiabi@univ-biskra.dz).

In this context, a new CAD software, which we called Python for Analog and Mixed Signals (PyAMS) is presented for the first time in this paper to simplify the modeling of analog elements and circuits. In particular, the behavioral modeling of systems by PyAMS is based on a revised computing technique involving the Python language and specific software packages and libraries. PyAMS can account for detailed circuit analyses in the frequency-domain (AC), time-domain (TR), and direct-current (DC) operating conditions. In more detail, the circuit simulation using PyMAS is based on a solving system of nonlinear equations by a continuous method which eliminates any convergence problem [16]-[25]. It could be useful to improve the convergence problem or to accelerate the findings of the standard NR method. The continuous method is based on three basic steps; (a) transforming nonlinear equations to homotopy equations; (b) estimating the direction of the solution by a prediction method; (c) correcting the process to get the final solution.

Different examples of modeling of electrical elements as well as the simulation results of various test circuits are presented in the following to verify the correct operating of the proposed PyAMS software.

II. Graphical User Interface of PyAMS

The graphical user interface (GUI) of PyAMS is presented in Fig. 1.

The objectives of this interface are:

- Drawing circuit by schematic (CAD approach);
- Creating new PyAMS models of electrical elements by using Python language;

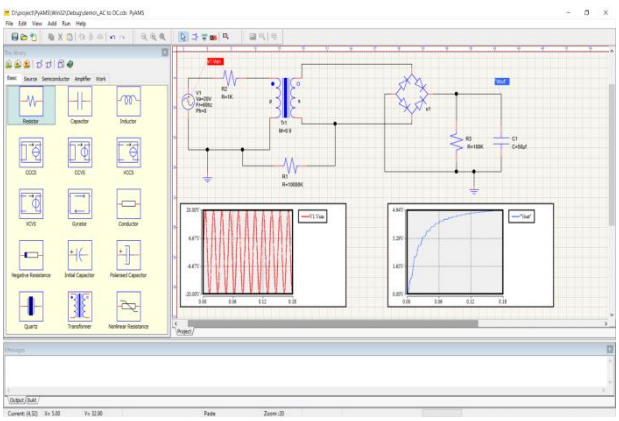


Fig. 1. GUI of PyAMS.

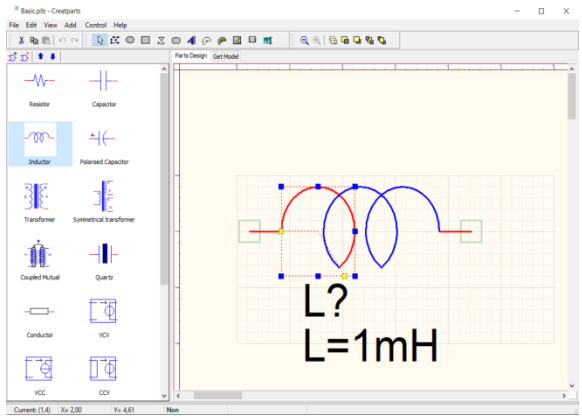
doi: 10.18178/ijeetc.10.4.233-242

- Creating new symbols for models;
- Modifying parameters of elements by using an intuitive language;
- Simulating the circuit in the selected mode of operation;
- Presenting simulation results in a dedicated waveform editor.

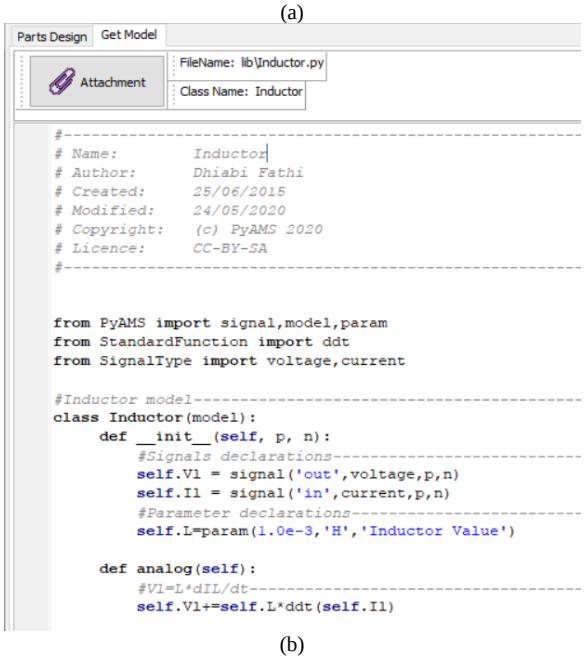
As shown in Fig. 2, the graphical interface of the editor of symbols includes: (a) the page for drawing analog elements, (b) the page for attaching symbols with a PyAMS description model based on the Python language.

In more detail, the creating of a symbol in the editor is based on the following steps:

- Drawing the electrical elements by lines, rectangles, ellipse, etc.;
- Adding ports;
- Adding references;
- Adding parameters;
- Joining the symbol with a PyAMS model.



(a)



(b)

Fig. 2. Graphical interface of the editor of symbols. (a) Page for drawing analog elements; (b) page for joining a symbol with a description model.

```
from PyAMS import model,signal   #Declaration of library

class name_of_element(model):    #Model name

   def __init__(self,ports):     #Initial function
                                 #Her signles and
                                 #parameters dclarations
   def analog(self):             #Analog function
                                 #Her equations or
                                 #analog operation
```

Fig. 3. Global form of a PyAMS model.

III. Modeling of Analog Elements

The modeling of analog elements in PyAMS is based on writing their description using the Python language respecting the following structure:

- Declaration of the library;
- Creation of the name of the model;
- Adding parameters with an initial value;
- Adding type of signals (current or voltage);
- Adding sub-models if available.
- Definition of relations between signals and parameters.

The global form for modeling any analog element in PyAMS is shown in Fig. 3.

The initial function (*def _init_*) is used to create ports for connections to other units, signals, variables, and parameters with default values. The analog function (*def analog*) is the behavioral modeling of the analog element by giving the relation between signals and parameters.

The *_init_* function and the *analog* function represent the principal functions of the model. However, there are two other secondary functions used to describe the model, i.e. the *sub* function, which is used to construct a sub-circuit in the model, and the *start* function, which is used to initialize values while starting a simulation.

IV. Signals and Parameters in PyAMS

Signals in PyAMS represent the voltage or current in the analog model with the direction and position of the connection. The signal declaration in PyAMS follows the statement: *signal ([Direction, Type, Port_p, Port_n])* where *Direction* is the direction of the signal (in or out), *Type* is the type of the signal (current or voltage), and *Port_p* and *Port_n* are where the positive terminal and the negative terminal are connected in the model, respectively.

Any electrical or analog element contains parameters represented by constant or variable values used for operation between voltages and currents. The parameter declaration defines its value, unit, and description as follows: *param([Value, Unit, Description])*.

The fundamental local functions of the behavioral modeling of analog elements referring to a specific PyAMS library are listed in Table I.

TABLE I: THE FUNDAMENTAL LOCAL FUNCTIONS BY PYAMS LIBRARY

Function	Application
ddt()	ddt([*Signal*]) is used to calculate the signal derivative by time.
idt()	idt([*Signal*]) is used to calculate signal integrative dependent by time.
limexp()	limexp([*Expression*]) is the limitation for an exponential function.
acSim()	acSim([*mag*, *phase*]) is the AC stimulus function. It returns 0 during DC analyses. During AC analyses the source becomes active and it is modeled with a magnitude, *mag*, and a phase, *phase*: *Mag* $e^{-j\ phase}$.
itDC()	It returns the true value when analysis by mode DC.
itAC()	It returns the true value when analysis by mode AC.
itTR()	It returns the true value when analysis by mode TR.
display()	display([*string*]) is used to display messages.
realTime()	It is the simulation time function which provides an access to current simulation time or returns a value of time as a real number.
temperature()	It is the simulation temperature in Kelvin.
vt()	vt([*temperature_expression*]) returns the thermal voltage at a given temperature. If the *temperature expression* is not supplied, the thermal voltage is calculated by using the standard expression *KT/q* where *K* is Boltzmann's constant, *T* is the simulation temperature, and *q* is the electron charge.

V. EXAMPLES OF MODELING ANALOG ELEMENTS

In this section, we explain how to model some analog elements that have two terminals (e.g. resistors and diodes), three poles (e.g. OP-AMPs), and four poles (e.g. transformers) by PyAMS library.

A. Resistors

As shown in Fig. 4, a resistor can be represented by a current source I_r, which is dependent on the voltage V_r applied between the nodes *p* and *n*, and the resistor value *R*. As well known, the current in the resistor is expressed by Ohm's law. The model of a resistor by PyAMS is shown in Fig. 5.

It initializes the parameter *R* and two types of signals. The first signal is defined by the current I_r flowing between the port *p* and the port *n*; the second signal is defined by applying a voltage V_r. In the analog function, we simply implement the expression:

$$I_r = V_r / R \tag{1}$$

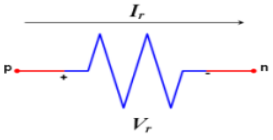


Fig. 4. Resistor element.

```
class Resistor(model):
    def __init__(self, p, n):
        #Signals declarations-----------------------------------
        self.Vr = signal('in',voltage,p,n)
        self.Ir = signal('out',current,p,n)

        #Parameter declarations---------------------------------
        self.R=param(1000.0,'Ohm','Resistance multiplier')


    def analog(self):
        #Resistor equation-Low hom (Ir=Vr/R)--------------------
        self.Ir+=self.Vr/self.R
```

Fig. 5. PyAMS model of a resistor.

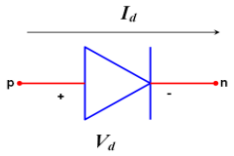


Fig. 6. Diode element.

```
class  Diode(model):
    def __init__(self, a, b):
        #Signals declarations-----------------------------------
        self.Vd = signal('in',voltage,a,b)
        self.Id = signal('out',current,a,b)
        #Parameter declarations---------------------------------
        self.Iss=param(1.0e-15,'A','Saturation current')
        self.Vt=param(0.025,'V','Thermal voltage')


    def analog(self):
        #Mathematical equation between Id and Vd----------------
        self.Id+=self.Iss*(explim(self.Vd/self.Vt)-1)
```

Fig. 7. PyAMS model of a diode.

B. Diodes

The diode is a nonlinear semiconductor device. It can be represented as shown in Fig. 6 by the I_d of the equivalent nonlinear current, which is dependent on the voltage V_d applied between the nodes *p* and *n*, i.e.

$$I_d = I_s\left(\exp\left(\frac{V_d}{Vt}\right)-1\right) \tag{2}$$

where V_t is the thermal voltage and I_s is the reverse bias saturation current.

The diode model in PyAMS is shown in Fig. 7. There are two types of signals in the model. The first signal is the current I_d flowing between the port *p* and the port *n*; the second signal is the applied voltage V_d. These two signals are declared in the initial function. The mathematical equation between V_d and I_d is implemented in the analog function declaration referring to (2).

C. OP-AMPs

The OP-AMP is a dc-coupled high-gain electronic voltage amplifier with a differential input voltage that, as shown in Fig. 8, can be represented by its equivalent voltage source V_{out} in the node *o*, depending on the voltage V_{in} applied between the nodes *p* and *n*, and its gain value *G*. The positive power supply (V_{s+}) and the negative power supply (V_{s-}) limit the effective device output voltage. The fundamental nonlinear OP-AMP expressions are:

$$V_{out} = \begin{cases} V_{s+}\dfrac{2}{\pi}\arctan\left(\dfrac{\pi}{2(1+V_{s+})}GV_{in}\right) & \text{for} \quad V_{in} > 0 \\ -V_{s-}\dfrac{2}{\pi}\arctan\left(\dfrac{\pi}{2(1-V_{s-})}GV_{in}\right) & \text{for} \quad V_{in} \le 0 \end{cases}. \tag{3}$$

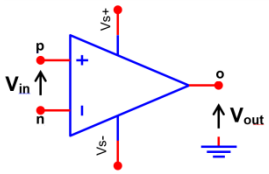


Fig. 8. The OP-AMP element.

```
class OpAmp(model):
  def __init__(self,p,n,sp,sn,o):
     #Signals declarations-------------------------------------------
     self.Vin = signal('in',voltage,p,n)
     self.Vout = signal('out',voltage,o)
     self.Vsp = signal('in',voltage,sp)
     self.Vsn = signal('in',voltage,sn)

     #Pramatre declarations------------------------------------------
     self.G =param(1000,' ','gain amplifier')

  def analog(self):

     if  self.Vin >0:
        self.Vout+=self.Vsp*2*atan(self.G*self.Vin*pi/((1+self.Vsp)*2))/pi;
     else:
        self.Vout+=-self.Vsn*2*atan(self.G*self.Vin*pi/((1-self.Vsn)*2))/pi;
```

Fig. 9. PyAMS model of an OP-AMP.

The device model in PyAMS is presented in Fig. 9. It has four types of signals. The first signal is the voltage V_{in} applied between the ports p and n with input direction. The second signal is the input voltage V_{sp} in the port sp. The third signal is the input voltage V_{sn} in the port sn. The last signal is the output voltage V_{out} obtained in the port o. These four signals are declared in the initial function. Equation (3) is properly defined in the analog function.

D. Transformers

A transformer is a passive electrical device that transfers electrical energy from an inductance to another one as schematized in Fig. 10.

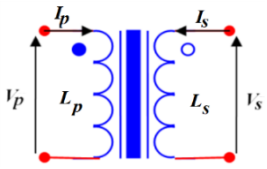


Fig. 10. Transformer element.

This model is based on two inductances, namely the primary (L_p) and the secondary (L_s), also accounting for their mutual inductance contribution M. The relation between the input and output voltages (V_p, V_s) and the currents (I_p, I_s) is expressed by the following system of equations:

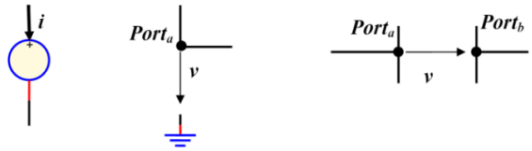
$$\begin{cases} V_p(t) = L_p \dfrac{\partial I_p}{\partial t} + M \dfrac{\partial I_s}{\partial t} \\ V_s(t) = L_s \dfrac{\partial I_s}{\partial t} + M \dfrac{\partial I_p}{\partial t} \end{cases}. \quad (4)$$

```
class Transformer(model):
     def __init__(self, P1, P2,S1, S2):
         #Signals declarations-------------------------------
         self.Vp = signal('out',voltage,P1, P2)
         self.Ip = signal('in',current,P1, P2)
         self.Vs = signal('out',voltage,S1, S2)
         self.Is = signal('in',current,S1, S2)

        #Parameter declarations-----------------------------
         self.Lp=param(1.0,'H','Primary inductance Value')
         self.Ls=param(1.0,'H','Secondary inductance Value')
         self.M=param(0.5,'H','Coupling inductance Value')


     def analog(self):
         #the equation of transformer-----------------------
         self.Vp+=self.Lp*ddt(self.Ip)+self.M*ddt(self.Is)
         self.Vs+=self.Ls*ddt(self.Is)+self.M*ddt(self.Ip)
```

Fig. 11. PyAMS model of a transformer.

(a) (b)

Fig. 12. Equivalent outputs of current (a) and voltage (b) signals when applied to a circuit.



(a) (b) (c)

Fig. 13. Equivalent inputs of current (a) and voltage (b, c) signals when applied to a circuit.

The model of the transformer in PyAMS is shown in Fig. 11. It has two currents signals, i.e. I_p and I_s with input direction, and two voltages signals, i.e. V_p and V_s with output direction. In particular, we consider I_p as the current across port P_1 to port P_2; I_s as the current across port S_1 to port S_2; V_p as the voltage between ports P_1 and P_2; V_s as the voltage between ports S_1 and S_2. These signals are declared in the initial function as shown in Fig. 11. The mathematical equations between the voltages and the currents (4) are defined in the analog function.

VI. Structure of Analog Elements in PyAMS

In PyAMS there are four types of signals, namely the output voltage, the output current, the input voltage, and the input current. The behavioral modeling of an analog component is expressed by these signals and parameters to obtain its function when applied in the circuit simulation.

It is important to note that during the circuit simulations the signals in the models are converted into sources and nodes to simplify the creation of the overall circuit equations. In more detail, we can state that

- The output current signal is equivalent to a dependent current source as shown in Fig. 12 (a);
- The output voltage signal is equivalent to a dependent voltage source as shown in Fig. 12 (b);
- The input current signal is equivalent to the current across a voltage source as shown in Fig. 13 (a);
- The input voltage signal is equivalent either to the voltage applied in a node or to the voltage between two nodes as shown in Fig. 13 (b) and (c), respectively.

As an example, we can consider the OP-AMP shown in Fig. 14 (a). By replacing the signals in the circuit with the dependent voltage sources as well as the signals in the resistors with the dependent current sources, we get the equivalent circuit shown in Fig. 14 (b). The relationships between I_{R1}, I_{R2}, and V_{out} are represented by the following set of equations:

$$\begin{cases} I_{R1}(v_2, v_3) = (v_2 - v_3)/R_1 \\ I_{R2}(v_2) = v_2/R_2 \\ V_{out}(v_1, v_2) = G(v_1 - v_2) \end{cases} \quad (5)$$

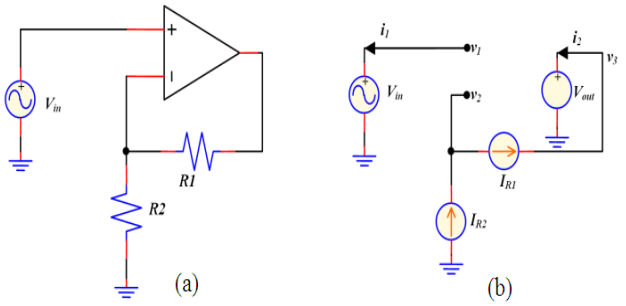


Fig. 14. OP-AMP with negative feedback (non-inverting amplifier). (a) Schematic circuit; (b) equivalent circuit for simulations.

VII. Circuit Global Equations in PyAMS

The aim of converting the circuit elements to their dependent sources is to simplify the generation of the global equations of the circuit by applying the two Kirchhoff's laws as recalled here in brief.

Kirchhoff Current Law (KCL) [9]:

$$\sum_k d_I I_k(v,i) + \sum_p d_i i_p = 0 \tag{6}$$

where d_I is the sign (±) of the direction of the dependent current source I, d_i is the sign of the direction of the current across the dependent voltage source, and k and p are the numbers of terms in each node.

Kirchhoff Voltage Law (KVL) [9], simplified by the modified nodal analysis (MNA) [10]:

$$v_a - v_b - V(v,i) = 0 \tag{7}$$

where the dependent voltage source is equal to the potential difference at the two conduction regions of the circuit.

Starting from (6) and (7), we obtain a general equation for the circuit in the form of

$$F(v,i) = \begin{cases} \sum_p d_i i_p + \sum_k d_I I_k(v,i) \\ v_a - v_b - V(v,i) \end{cases} \tag{8}$$

where

$$F(v,i) = 0, \ (v, i) = (v_1, v_2, ..., v_n, i_1, i_2, ..., i_m) \tag{9}$$

is a system of equations, namely $F: \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n+m}$ where n is the number of nodes in the circuit and m is the number of voltage sources. For example, we can write:

$$F(v_1, v_2, v_3, i_1, i_2) = \begin{cases} i_1 \\ i_2 - I_{R1}(v_2, v_3) \\ I_{R1}(v_2, v_3) + I_{R2}(v_2) \\ v_1 - V_{in} \\ v_3 - V_{out}(v_1, v_2) \end{cases} \tag{10}$$

by applying the KCL and MNA laws in the three nodes and two voltage sources of the equivalent circuit shown in Fig. 14 (b).

VIII. Method of Finding Operating Points

The proposed PyAMS simulator uses the NR (Newton Raphson) method to find the operating points by solving the equations systems for analog circuits [3]. The NR method and its variations are widely used for systems of nonlinear equations [26]. In more detail, the NR method for (9) is an iterative expression given by

$$\begin{pmatrix} v_1^{k+1} \\ i_1^{k+1} \end{pmatrix} = \begin{pmatrix} v_1^{k} \\ i_1^{k} \end{pmatrix} - \frac{F(v,i)}{J_F(v,i)} \tag{11}$$

$$J_F(v,i) = \begin{bmatrix} \frac{\partial f_1}{\partial v_1} & \frac{\partial f_1}{\partial v_2} & \cdots & \frac{\partial f_1}{\partial i_m} \\ \frac{\partial f_2}{\partial v_1} & \frac{\partial f_2}{\partial v_2} & \cdots & \frac{\partial f_2}{\partial i_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{n+m}}{\partial v_1} & \frac{\partial f_{n+m}}{\partial v_2} & \cdots & \frac{\partial f_{n+m}}{\partial i_m} \end{bmatrix} \tag{12}$$

where k is the iteration index, v_k and i_k are the relative approximate solutions, and J_F is the Jacobian matrix of the term F. The iteration in (11) stops according to the Newton method when

$$\begin{cases} \|x_{k+1} - x_k\| \le \text{reltol} \times \max(\|x_{k+1}\|, \|x_k\|) + \text{abstol} \\ x \in (v,i) \end{cases} \tag{13}$$

assuming reltol and abstol as the constants which guarantee a good converge result.

If a convergence problem occurs, the best way to accelerate the solution of a system of equations as in (9) is to use a practical continuous method. The continuous method which is used by PyAMS [14] is, in average, about three times faster than the variable gain Newton homotopy method (VGNH) [27] implemented in SPICE as highlighted in Table II.

Table II: Comparison between the VGNH and the Continuous method

Circuit	Computation time (ms)	
	VGNH (SPICE)	Continuous (PyAMS)
VRef	3.36	1.12
6sLA	3.20	1.92
µA741	4.48	1.12
RCA3040	3.36	1.28
2sOA	3.20	0.80
Average	3.52	1.28

The continuous method allows to convert F to a homotopy equation (H) well suited to use a predictor corrector solver [14]. In particular, the homotopy equation H is based on embedding a continuation parameter λ into $F(v, i)$. As a result, a new equation

$$H(v,i,\lambda) = 0 \tag{14}$$

where $\lambda \in \mathbb{R}$ and $H: \mathbb{R}^{n+m} \times \mathbb{R} \rightarrow \mathbb{R}^{n+m}$ is considered. At present, the most widely used continuous method is the Newton homotopy mapping (NH) [14] which is in the form of

$$H(v,i,\lambda)=F(v,i)+(1-\lambda)F(0,0). \quad (15)$$

According to this expression, the homotopy equation to be solved becomes

$$H(v,i,\lambda)=\begin{cases}\sum_n d_n i+\sum_k d_I\left(I_k(v,i)+(\lambda-1)I_k(0,0)\right)\\ v_a-v_b-\left(V(v,i)+(\lambda-1)V(0,0)\right)\end{cases} \quad (16)$$

and by applying NR we can write

$$\begin{pmatrix}v_1^{k+1}\\ i_1^{k+1}\end{pmatrix}=\begin{pmatrix}v_1^{k}\\ i_1^{k}\end{pmatrix}-\frac{H(v,i,\lambda)}{J_H(v,i)} \quad (17)$$

for any λ ranging from 0 to 1, if the homotopy equation solution (v, i, λ) exists, the corresponding curve of (v, i, λ) starts from (v_0, i_0, 0) and ends in the solution (v^*,i^*,1) [14].

IX. Circuit Analysis

The simulation of analog circuits by means of PyAMS can be performed in three different domains:

- Quiescent domain, i.e. direct current (DC) mode;
- Time-domain, i.e. transient analysis (TR) mode;
- Frequency domain, i.e. harmonic analysis (AC) mode.

In this section, we discuss the finding of the operating points (OPs) for each algorithm mentioned above.

A. DC Analysis

The DC analysis is based on the algorithm schematized in Fig. 15. To find the OPs, this analysis gets the operation of the circuit by variation one parameter in turns (e.g., the resistor value, current value, voltage value, temperature value, etc.). In this analysis, the nonlinear dynamic signals are replaced by sources with a zero value.

B. Transient Analysis

The flow chart of the TR analysis by PyAMS is shown in Fig. 16. This analysis involves analog elements, which have derivative or integrative signals. It uses a numerical integration method to convert nonlinear differential equations to nonlinear algebraic equation in order to simplify the solving by the NR method or the continuous method if convergence problems occur [14].

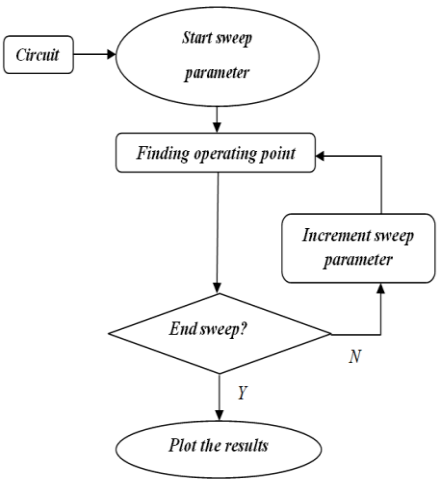


Fig. 15. DC analysis flow chart.

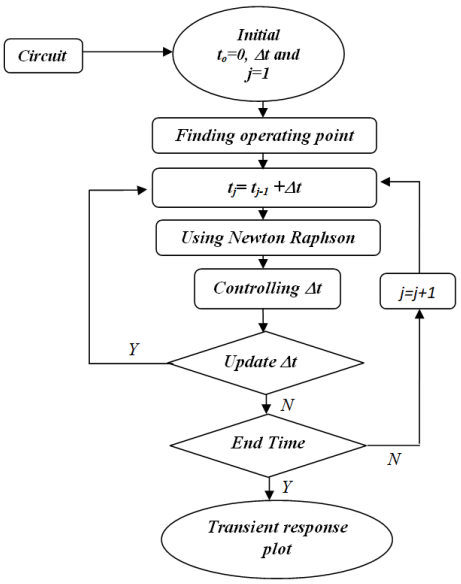


Fig. 16. TR analysis flow chart.

In Fig. 16, Δt is the step time and its value is minimized when having problems of solving of differential equations by using a controlling step time method as in Spice [3], [7].

C. Small Signal AC Analysis

The AC analysis algorithm is represented in the flow diagram shown in Fig. 17.

The fundamental steps are (a) finding the operating points, (b) converting the analog elements into impedance terms, and (c) formulating the matrix system equation for the circuit. In this analysis, the obtained system equation is linear and we solve it according to the KLU algorithm [12], [28].

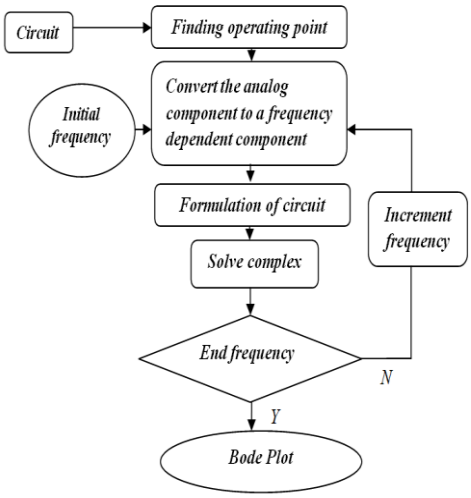


Fig. 17. AC analysis flow chart.

X. SPICE and PyAMS Overview

The structure of an electrical circuit and its analysis in SPICE are defined by the Netlist form where we can find a textual description of the name of each analog element, with its interconnection in the circuit, and the analysis commands. The applied system of equations that SPICE

adopts for solutions is based on the MNA (Modified Nodal Analysis) method. The objective of the MNA method is to simplify the structure of the passive elements, active elements, and current and voltage sources in a matrix equation which can be solved using linear methods as the LU decomposition method [3]. The nonlinear elements are converted to linear elements by using the NR method. Convergence problems, however, could need specific techniques of solving [3]. Finally, VHDL-AMS and Verilog-AMS are languages integrated with SPICE that can be used for modeling analog, digital, and mixed elements simplifying the building of new elements [10], [29].

Although the SPICE has a great utility, it has some drawbacks:

- The elements models are unchangeable standard models and the user can not add new elements [1];
- The models created by VHDL-AMS are not used directly in SPICE but they are first compiled to find errors of description and then converted to be used in the Netlist form (e.g., Vhdl2Spice [29] and Verilog-a [10]). Thus, convergence problems can occur for nonlinear elements;
- The convergence remains a problem in the solution of the matrix equation representing the circuit, especially in large complex circuits;
- The circuit analysis can be time consuming due to the slowness of the equation solving [6];

In this context, with the aim to overcome the mentioned drawbacks also introducing some new features, in this paper the new PyAMS software is proposed for creating analog elements and constructing circuits with the use of the Python language. In particular,

- The user can create new models applicable directly in the circuits;
- The software uses the continuous method based on the homotopy equation to accelerate the solutions finding;
- The circuit analysis is faster.
- The simplicity in creating sub-models for any compound element is pursued;
- The element description is based on dependent sources;
- Circuits can be designed by using an intuitive graphical mode description by schematic diagram.

XI. Obtained Results

In this section, we present the PyAMS simulations of some examples of standard analog circuits. In particular, starting from the creation of the proper PyAMS models, the design of the symbols, and the design of the circuits, we have performed the simulations of the following different circuits:

- Five-stage ring oscillator (Fig. 18).
- Chua's circuit (Fig. 19).
- State variable filter (Fig. 20).
- Frequency-shifted keyed (FSK) demodulation (Fig. 21).
- Peltz oscillator (Fig. 22).
- AC/DC converter (Fig. 23).
- Non-inverting amplifier circuit (Fig. 24).

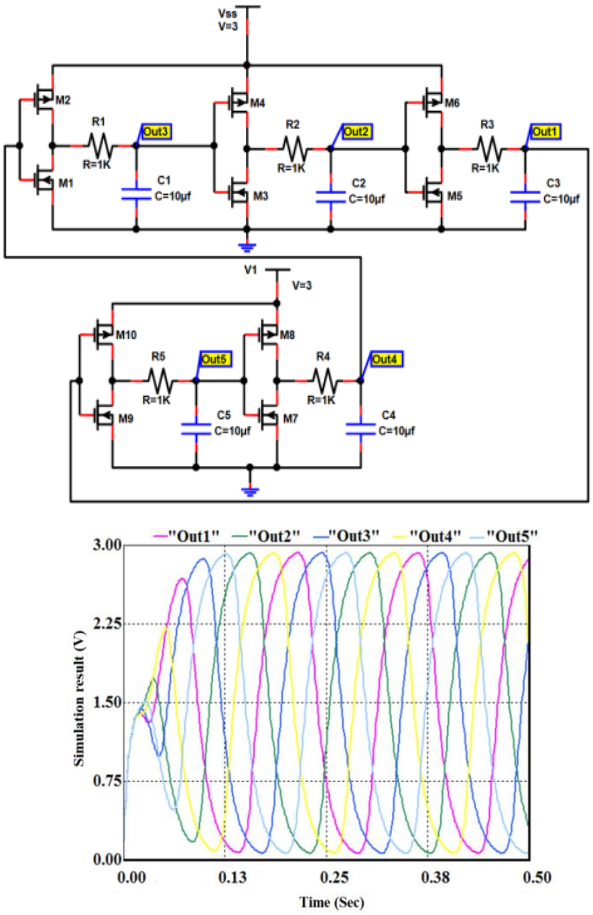


Fig. 18. Five-stage ring oscillator and relative simulation results.

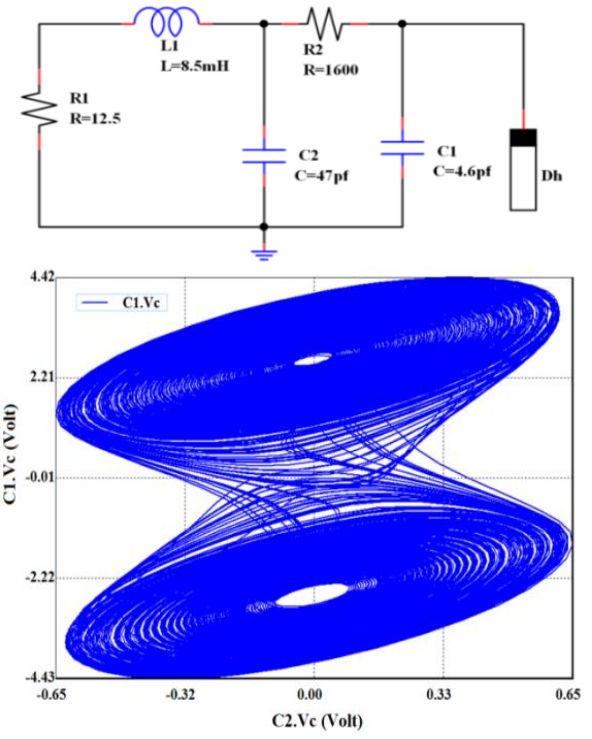


Fig. 19. Chua's circuit and relative simulation results.

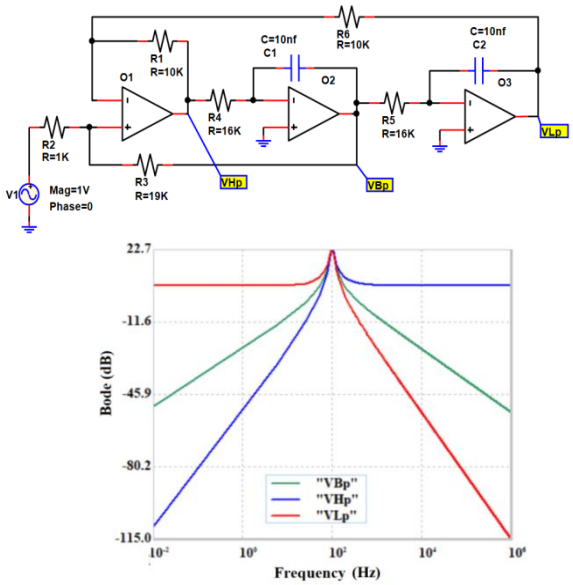


Fig. 20. Circuit of a state variable filter and relative simulation results.

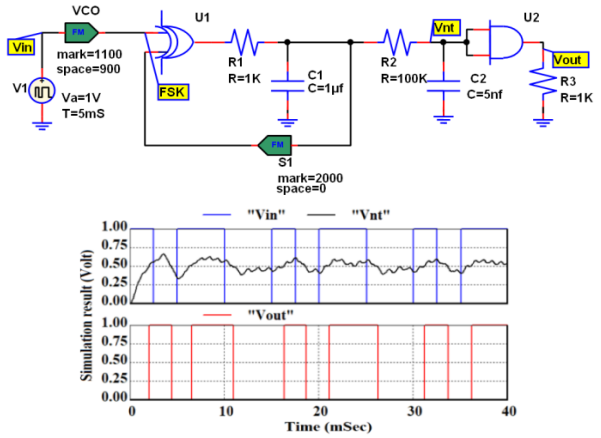


Fig. 21. FSK demodulation circuit with phase locked loop (PLL) and relative simulation results.

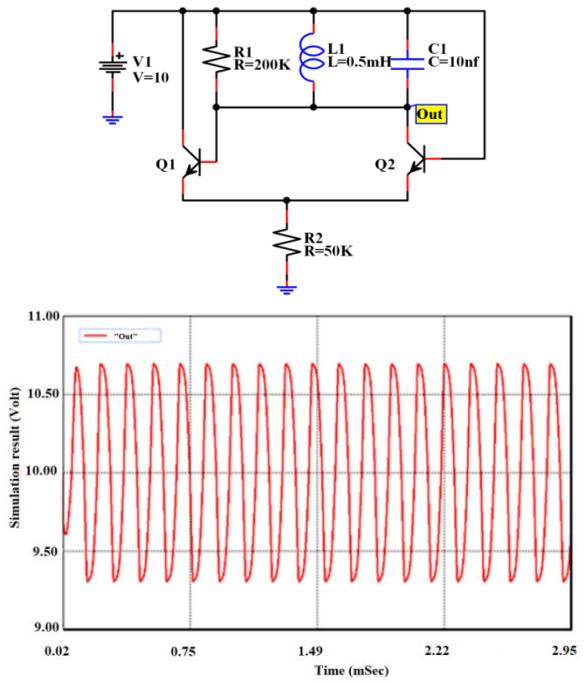


Fig. 22. Peltz oscillator circuit and relative simulation results.

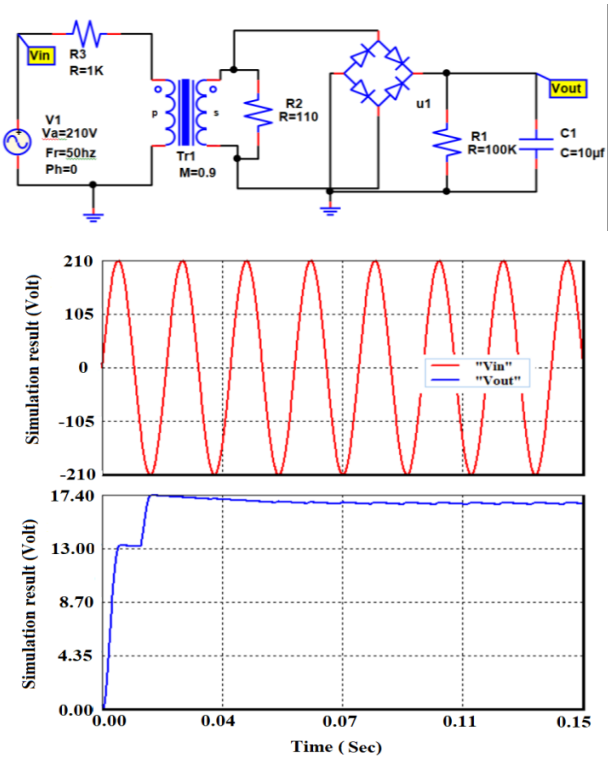


Fig. 23. Circuit of an AC/DC converter and relative simulation results.

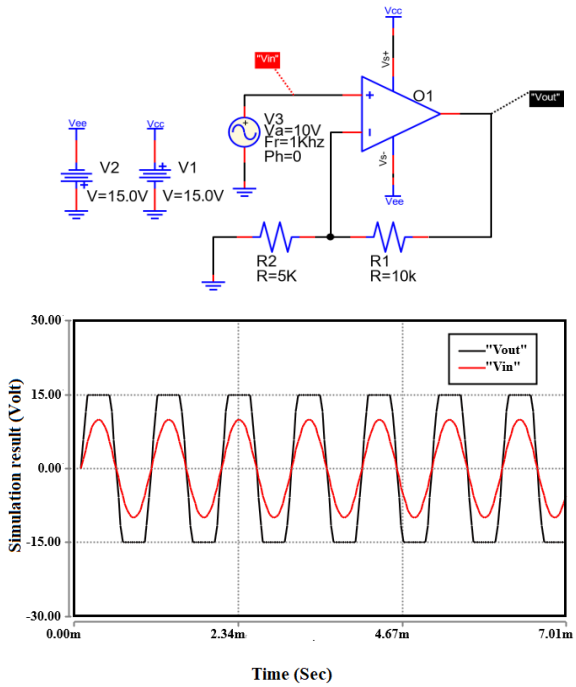


Fig. 24. Non-inverting amplifier circuit.

From the simulations, we can appreciate the effectiveness of the proposed PyAMS software in analyzing universal and complex circuits starting from a careful modeling of the analog elements.

XII. Conclusion

A new software based on the Python language that we have called PyAMS has been proposed for modeling analog elements and investigating electrical circuits in a

simply way. Thanks to a revised computing approach to solve systems of nonlinear equations by using a continuous method, which eliminates any convergence problem, PyAMS allows to find correct solutions minimizing the computational times with respect to the standard NR methods as used for example in SPICE. The Python language is practical to describe specific simulation models where an initial function is defined for declaration of signals and parameters whereas an analog function specifies the different operations accounted for.

In the different sections of the paper, we have presented the graphical user interface of PyAMS, the modeling and structure of fundamental analog elements, the equivalent global equations to be solved for a given circuit, and the methods of finding the operation points according to the selected analysis mode. Finally, the simulation results of different test circuits have proven the effectiveness of the proposed software. This work could be a good basis for further developments which include, for example, noise effects in the circuits.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Dhiabi Fathi designed and tested the PyAMS software and wrote the original draft of the manuscript. M.Larbi Megherbi and Saadoune Achour contributed to the methodology for the research plan and the manuscript organization. Fortunato Pezzimenti and Riccardo Carotenuto rewrote the manuscript accordingly to their analysis of the results enhancing the integrity of any parts of the work. All authors had approved the final version.

REFERENCES

[1] M. Brinson and V. Kuznetsov, "Extended behavioural device modelling and circuit simulation with Qucs-S," *Int. J. Electron.*, vol. 105, no. 3, pp. 412-425, July 2017.

[2] A. N. Shatunov, A. V. Zubarev, M. Yermekova, and O. I. Shurdukova, "Electrical circuits simulation in free GPL software," in *Proc. IEEE Conf. of Russian Young Researchers in Electrical and Electronic Engineering*, Petersburg and Moscow, 2020, pp. 173-175.

[3] A. Vladimirescu, *The SPICE Book*, New York, USA: Wiley, 1994, ch. 1, pp. 1-5.

[4] M. Brinson and S. Jahn, "Compact device modeling for established and emerging technologies with the Qucs GPL circuit simulator," in *Proc. MIXDES-16th Int. Conf. Mixed Design of Integrated Circuits & Systems*, Lodz, 2009, pp. 39-44

[5] M. Brinson and V. Kuznetsov, "Qucs-0.0.19S: A new open-source circuit simulator and its application for hardware design," in *Proc. International Siberian Conference on Control and Communications*, Moscow, 2016, pp. 1-5.

[6] A. Bürmen, J. Puhan, T. Tuma, I. Fajfar, and A. Nussdorfer, "Model parameter identification with SPICE OPUS: A comparison of direct search and elitistic genetic algorithm," in *Proc. 15th European Conference on Circuit Theory and Design*, Helsinki University, 2001, pp. 61-64.

[7] T. Tuma and Á. Buermen, *Circuit Simulation with SPICE OPUS: Theory and Practice*, Boston, USA: Springer Science & Business Media, 2009, ch. 1.

[8] F. Paulů and J. Hospodka, "GEEC: Graphic editor of electrical circuits," in *Proc. Int. Conf. on Applied Electronics*, Pilsen, 2017, pp. 1-4.

[9] M. Brinson and V. Kuznetsov, "Recent developments in Qucs-s equation-defined modelling of semiconductor devices and IC's," *Int. J. Microelectron. Comput. Sci.*, vol. 8, no. 1, pp. 29-35, April 2017.

[10] M. Brinson and V. Kuznetsov, "A new approach to compact semiconductor device modelling with Qucs Verilog-A analogue module synthesis," *Int. J. Numer. Model. El.*, vol. 29, no. 6, pp. 1070-1088, 2016.

[11] K. Kundert, *The Designer's Guide to Spice and Spectre*, California, USA: Springer Science & Business Media, 2006.

[12] F. Lannutti, P. Nenzi, and M. Olivieri, "KLU sparse direct linear solver implementation into NGSPICE," in *Proc. 19th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES*, Warsaw, Poland, 2012, pp. 69-73

[13] M. Tadeusiewicz and A. Kuczyński, "A very fast method for the DC analysis of diode–transistor circuits," *Circuits Syst. Signal Process.*, vol. 32, no. 2, pp. 433-451, Aug. 2012.

[14] F. Dhiabi and M. Boumehraz, "Accelerating the solving of nonlinear equations using the homotopy method: Application on finding the operating point of complex circuits," *Turk. J. Elec. Eng. & Comp. Sci.*, vol. 25, no. 3, pp. 1864-1880, May 2017.

[15] K. Yamamura, "SPICE-oriented numerical methods for solving nonlinear problems where formulas are described by circuits," *J. IEICE*, vol. 88, no. 4, pp. 825-831, April 2005.

[16] D. Niu, G. He, Q. Ye, X. Wang, and X. Zhou, "An effective embed algorithm for newton fixed-point homotopy method in MOS transistor circuits," *Int. J. Inf. Electron. Eng.*, vol. 6, no. 3, pp. 32-36, Jan. 2016.

[17] K. Yamamura and T. Miyamoto, "DC operating point analysis of transistor circuits using the variable-gain homotopy method," *J. IEICE*, vol. 97, no. 5, pp. 1042-1050, May 2014.

[18] L. Trajkovic, "DC operating points of transistor circuits," *Nonlinear Theory and Its Applications*, vol. 3, no. 3, pp. 287-300, July 2012.

[19] F. Dhiabi and M. Boumehraz, "Solving nonlinear equations by applying homotopy method to find operating point in analog circuit," in *Proc. Conf. on Electrical Engineering*, Biskra, 2014, pp. 7-8.

[20] X. Wu, Z. Jin, D. Niu, and Y. Inoue, "A PTA method using numerical integration algorithms with artificial damping for solving nonlinear DC circuits," *Nonlinear Theory and Its Applications*, vol. 5, no. 4, pp. 512-522, Oct. 2014.

[21] M. Jereminov, A. Terzakis, M. Wagner, A. Pandey, and L. Pileggi, "Robust and efficient power flow convergence with g-min stepping homotopy method," in *Proc. IEEE Int. Conf. on Environment and Electrical Engineering Industrial and Commercial Power Systems Europe*, Genova, 2019, pp. 1-6.

[22] L. Tang, J. Ruan, and R. Chen, "A homotopy continuation method for nonlinear electric field analysis," *IEEE T. Dielect. EL. IN.*, vol. 26, no. 4, pp. 1125-1133, Aug. 2019.

[23] W. Gu, W. Liu, and K. Zhang, "Variable step-size prediction–correction homotopy method for tracking high-dimension hopf bifurcations," *Int. J. Elec. Power. Energ. Syst.*, vol. 33, no. 6, pp. 1229-1235, July 2011.

[24] Y. Inoue, S. Kusanobu, K. Yamamura, and M. Ando, "An initial solution algorithm for globally convergent homotopy methods," *IEICE Trans. Fundamentals, Communications and Computer Sciences*, vol. 87, no. 4, pp. 780-786, April 2004.

[25] H. Vazquez-Leal, A. Sarmiento-Reyes *et al*., "A homotopy continuation approach for testing a basic analog circuit," *British J. Math. & Comp. Sci*., vol. 3, no. 3, pp. 226-240, 2013.

[26] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, Philadelphia, USA: SIAM, 2003, ch. 2, pp. 27-51.

[27] K. Yamamura and T. Miyamoto, "DC operating point analysis of transistor circuits using the variable-gain homotopy Method," *IEICE Trans. Fundamentals, Communications and Computer Sciences*, vol. 97, no. 5, pp. 1042-1050, May 2014.

[28] T. A. Davis and E. N. Palamadai, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Softw.*, vol. 37, no. 3, pp. 1-17, Sept. 2010.

[29] S. Li, B. Okoon, M. Hella, M. Ismail, and M. Rubeiz, "The implementation of a VHDL-AMS to SPICE converter," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 22, pp. 113-121, Sept. 1999.

Dhiabi Fathi was born in Biskra, Algeria, in 1977. He received the Elec. Eng. degree from the Faculty of Sciences and Technology, University of Biskra in 2002, the M.S. degree in 2004 and the Ph.D. degree in 2018 from the same university. From 2018, he worked as a researcher in Laboratory of Metallic and Semi- conducting Materials (LMSM), Department of Electrical Engineering, Faculty of Sciences and Technology, Biskra University. In 2013, he joined the Faculty of Sciences and Technology of Biskra as an assistant professor of electronics, Instrumentations and microelectronics. Currently, he is assistant professor of electronics and telecommunications.

M. Larbi Megherbi received the B.Sc. degree in electronic engineering from Batna University, Algeria in 1986, the M.Sc. degree in magnetic material from the University of Wales, Cardiff, UK, in 1988, and the Ph.D. degree in electrical engineering from Biskra University, Algeria. He is currently with the LMSM Laboratory for semiconductor material, Biskra. His current research interest including modeling and simulation of wide bandgap semiconductor device for high-temperature and high-power application.

Saadoune Achour received the Ing. degree in electronics, the M.Sc. degree in electronics, the Ph.D. degree in electrical engineering, and the HDR degree in electrical engineering from Biskra University, Biskra, Algeria, in 2001, 2004, 2009, and 2014, respectively. He is currently a Researcher with the Laboratory of Metallic and Semiconducting Materials, University of Biskra,

Riccardo Carotenuto was born in Rome, Italy. He received the Ph.D. degree in electronic engineering from the Sapienza University of Rome, Rome, Italy. He is currently associate professor of electronics with the Mediterranea University of Reggio Calabria, Reggio Calabria, Italy. His current research interests include indoor positioning, smart sensors, energy harvesting, electronic devices, ultrasound imaging.

Fortunato Pezzimenti received the Laurea degree in electronic engineering from Mediterranea University of Reggio Calabria, Italy, in 2000 and the Ph.D. degree in 2004. Since 2006, he is assistant professor of electronics at the Mediterranea University of Reggio Calabria. His current research interests include design, modeling, and electrical characterization of novel semiconductor devices and circuits.