electronics

MDPI

Article

The Method of Elementary Solvers in SPICE

Enrique Miranda

Departament d’Enginyeria Electrònica, Universitat Autònoma de Barcelona, 08193 Cerdanyola del Valles, Spain; enrique.miranda@uab.cat

Abstract: Circuit simulators are fundamentally used for solving electric circuit problems with different degrees of complexity in which node voltages and branch currents are the unknowns. This is fully understandable since they were originally created for this specific task. However, behind the curtains, powerful simulation engines based on a variety of numerical techniques operate so as to always comply with Kirchhoff’s current and voltage laws. In this paper, it is shown how a simple circuital configuration, referred to as the elementary solver, consistent in two behavioral current sources in series, can be used to solve mathematical problems that go beyond electronics. Of course, the intention is not to substitute mathematical packages with well-proven calculus capacity but to increase the scope of circuit simulators for their application in other areas of research or simply for educational purposes. It is worth mentioning that no special programming skills are required (except a basic knowledge of the available tools and language) and, furthermore, that the user can operate exclusively in a graphical environment. It is shown, throughout a series of selected examples, how the method of elementary solvers (MES) works, providing a new and practical dimension to the applicability of circuit simulators.

Keywords: circuit simulator; LTSpice; SPICE; modeling; simulation; differential equations; problem solving

check for updates

Citation: Miranda, E. The Method of Elementary Solvers in SPICE. *Electronics* **2024**, *13*, 2480. https://doi.org/10.3390/electronics13132480

Academic Editor: Paolo Stefano Crovetti

Received: 21 May 2024
Revised: 9 June 2024
Accepted: 20 June 2024
Published: 25 June 2024

1. Introduction

This paper is not about electric circuits. It is about the use of a specific electric circuit for solving mathematical problems that can be expressed as a single equation or as a system of equations. However, before getting into the topic, it is worth mentioning that, to our knowledge, the approach we are going to follow is different from previous proposals. The idea that we can represent equations associated with mathematical, physical, engineering, biological, epidemiological, etc., problems in terms of electric components is not new of course and the literature has plenty of cases of such applications [1–4]. For instance, very often, if we need a derivator or an integrator we think of a combination of components such as resistors, capacitors, inductances, opamps, etc. But why do we not make direct use of the behavioral components circuit simulators offer instead? In this way, we can straightforwardly benefit from the heart of the simulation engines upon which circuit simulators are built without establishing correspondence with any specific electric circuit. At the end, a circuit simulator is nothing but an equations solver. However, the question is whether it is possible or not to define a very elemental electrical cell suitable for solving a wide variety of problems without requiring specialized programming skills. Of course, we are not thinking of substituting mathematical packages of proven calculus capacity to this purpose. We are thinking about something simple and adaptable, and in any case, expressible by means of a few commands and directives. In order to accomplish this objective, we are going to use the LTSpice XVII simulator from Analog Devices [5], but any other circuit simulator would be appropriate in principle. LTSpice is a free SPICE simulator software that offers a powerful simulation engine and, perhaps what is relevant to our aim, an easy to use graphical interface which can be combined with a waveform viewer. LTSpice is available both for Windows and MacOS. Apart from the numerous papers, webpages, tutorials, and videos on LTSpice, some books have been recently published [6–9].

We will refer to the proposed electric cell as the elementary solver (ES). This cell simply consists in two behavioral current sources (B-sources B1 and B2) connected in series with a central node x as illustrated in Figure 1. F and G are two functions of the voltage V(x), i.e., the unknown. The triangle at the bottom of the mesh indicates the connection to ground (this is always required!). Because of the particular arrangement considered, the cell automatically complies with Kirchhoff's current law (ΣI = 0) at node x.

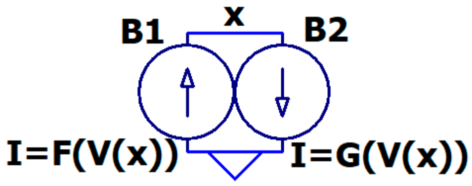
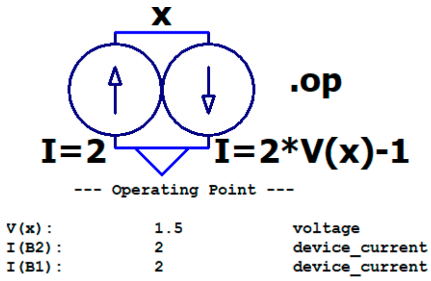


Figure 1. Schematics for the elementary solver (ES).

However, since B1 and B2 can be somehow regarded as a parallel combination too, the voltage drops across them must also coincide because of Kirchhoff's voltage law (ΣV = 0). At the end, V(x) is the value required to equalize the currents (F[V(x)] = G[V(x)]), i.e., the solution of the problem we want to solve.

For the sake of clarity, Figure 2 shows an elementary introductory example. The equation to be solved is 2 = 2·x − 1, with x as the unknown. Since x is a label in the circuit, we have to refer to the voltage value at node x, i.e., V(x). Of course, we can always rewrite the proposed equation as 0 = 2·x − 3 if necessary. **.op** is the LTSpice directive for the DC operating point (stationary solution). For time-dependent problems, a transient analysis **.tran** must be invoked instead. The solution V(x) = 1.5, i.e., x = 1.5 can be found in the bottom panel of Figure 2. Notice also that I(B1) = I(B2) as expected (this must always be checked!). If the user prefers to work in text mode, the script corresponding to the circuit shown in Figure 2 can be found in the "SPICE Netlist" window.



```
       --- Operating Point ---

V(x):           1.5          voltage
I(B2):          2            device_current
I(B1):          2            device_current
```

Figure 2. Solution of a 1D linear equation using the ES.

If, because of the complexity of the problem we want to solve, the operating point cannot be found with successive linear approximations as is conducted using the Newton–Raphson (NR) iteration method, it is always advisable to try a different method. The corresponding directives are indicated in Table 1 and they can be added to the circuit diagram.

Table 1. List of directives corresponding to the numerical methods used by LTSpice.

Method	Directive to Disable
Direct Newton Iteration	.options noopiter
Adaptive Gmin Stepping	.options gminsteps = 0
Adaptive Source Stepping	.options srcsteps = 0
Pseudo Transient	.options ptrantau = 0

It is also worth mentioning that in many cases, the variable **time**, which is always assumed positive and reserved for transient problems, can be used with an alternative meaning, for example, as the abscissa in a 2D coordinate problem. If a negative x-axis is required, the **.step** directive and discretized versions of the derivatives must be considered for **.op** conditions. Even though LTSpice can deal with complex numbers, mainly for frequency analysis, in this paper we will exclusively focus on the real axis.

2. Selected Examples

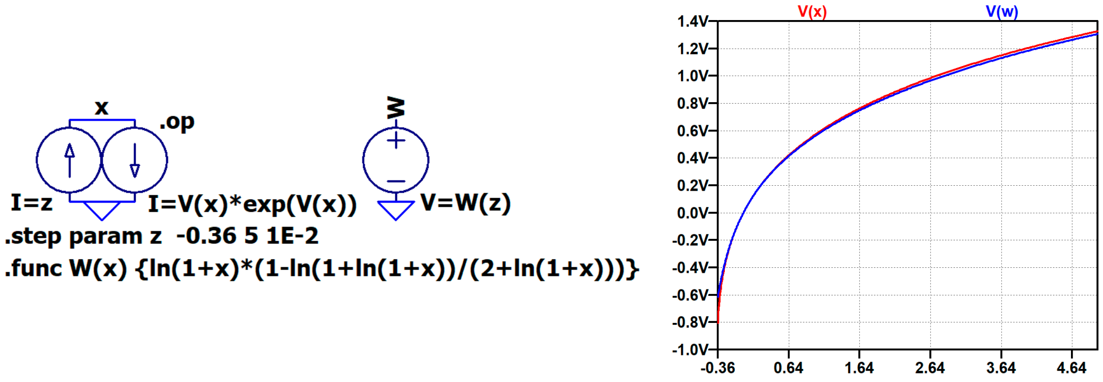
In this section, a number of selected problems will be presented and discussed, mainly in connection with their practical implementation. The examples reported below do not pretend to cover all cases and situations. They are just presented for illustrative purposes and many of them are widely discussed in the literature. In some cases, exact solutions or approximations are included for comparison. We will also make extensive use of behavioral voltage sources as function plotters. Since this is not a course on mathematical calculus (neither on any other discipline!), for the sake of simplicity, we will skip derivations and details. Please pay attention to the definition of the current sources B1 and B2. If inadvertently exchanged, computational errors can arise.

2.1. Solution of a Nonlinear Equation: Lambert W Function

We start this selection with an extension of the problem presented in Figure 2. Here, we will focus the attention on the solution of a transcendental equation. Contrary to the previous example, the solution to the equation is as follows:

$$W{\cdot}\exp(W) = z \tag{1}$$

where $z > -1/e$ is a real number which cannot be expressed in terms of elementary functions. $W(z)$ is called the Lambert W function and can be found in combinatory analysis, combustion problems, electric circuits, population dynamics, etc. [10]. When z changes, the principal branch of W is generated. The corresponding ES with the stepped z parameter is shown in Figure 3a. A closed-form Hermite–Padé approximation for $W(z)$ is also included in the schematics [11]. Figure 3b illustrates both the numerical and approximate results for $W(z)$.



```
.step param z  -0.36 5 1E-2
.func W(x) {ln(1+x)*(1-ln(1+ln(1+x))/(2+ln(1+x)))}
```

(**a**) (**b**)

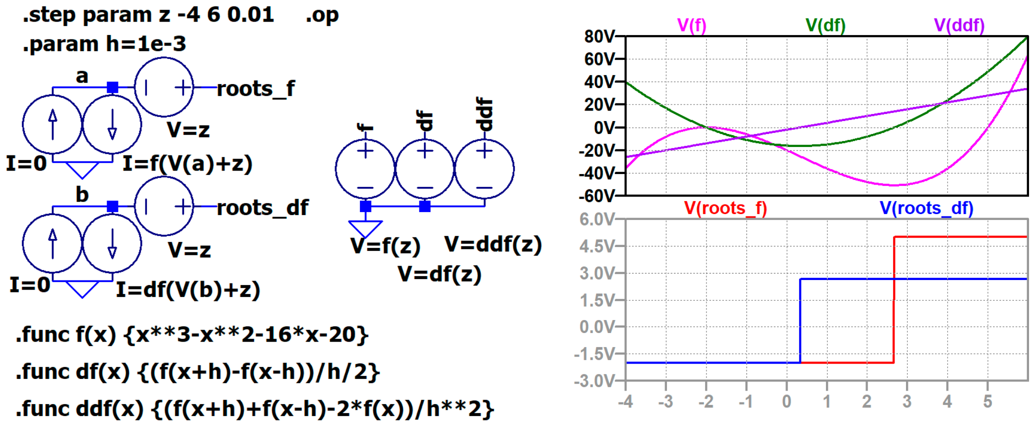
Figure 3. (**a**) ES for $W(z)$ and Hermite–Padé approximation. (**b**) Curves corresponding to the numerical solution (x) and analytic approximation (w).

2.2. Analysis of a Function: Polynomial Case

This second selected example focuses on the analysis of a cubic polynomial function such as the one below:

$$f(x) = x^3 - x^2 - 16x - 20 \tag{2}$$

In this case, not only the function f but also its first and second derivatives are plotted as a function of the argument (see the set of voltage plotters in Figure 4a). First, notice that the discrete forms of the derivatives are used (central case difference). h is the discretization step. Second, in order to find the roots of the function (zeroes of f) and of its first derivative (local extremes of f), a shift z in the function argument is introduced. Recall that the circuit simulator only provides a single solution (we are dealing with a conventional electric circuit!). The shift is compensated by a second voltage source of value z at the ES output port. The roots must be read in the y-axis of the plot as indicated by the horizontal lines (see Figure 4b). A reasonable range for z must be initially provided. Notice that horizontal asymptotes can also be identified as minimums/maximums since the first derivative could reach values close to zero in those regions. Switching between the root values can also occur randomly because of the NR iteration method. Trying different z steps is also advisable.



```
.step param z -4 6 0.01    .op
.param h=1e-3
.func f(x) {x**3-x**2-16*x-20}
.func df(x) {(f(x+h)-f(x-h))/h/2}
.func ddf(x) {(f(x+h)+f(x-h)-2*f(x))/h**2}
```

(**a**) (**b**)

Figure 4. (**a**) ES for function analysis. (**b**) Plots of the function (f), first derivative (df), and second derivative (ddf). The bottom panels show the roots of the function (f) and of its first derivative (df).

2.3. Indirect Integration: Gamma Function

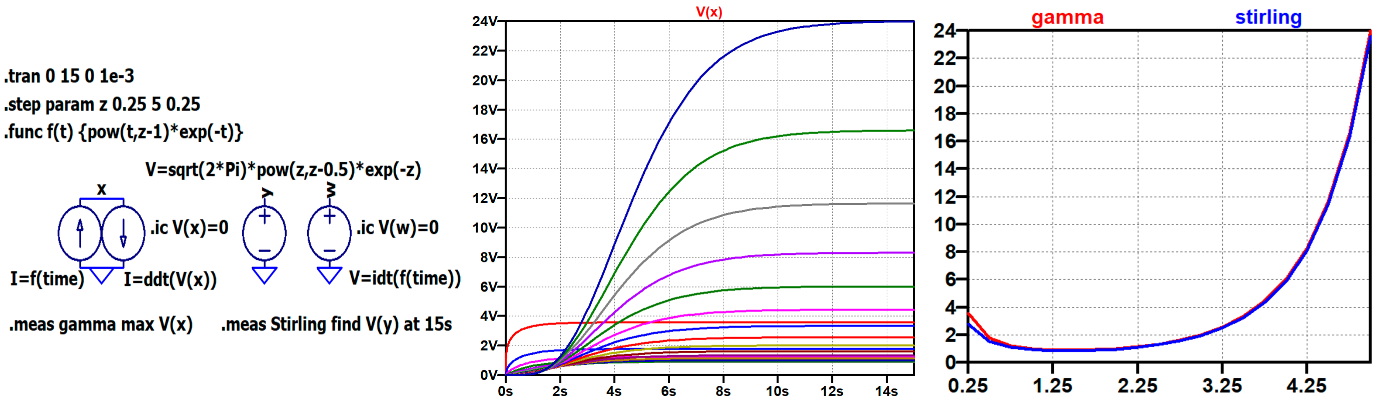
This exercise deals with the fundamental theorem of calculus which states that the integral of a function f over a fixed interval is equal to the difference of the antiderivative F evaluated at the ends of the interval. We will use the theorem to calculate the gamma function, which can be considered an extension of the factorial function to non-integer values [12,13]. We will limit the analysis here to the positive real axis. Notice that the intention here is not to find an approximate expression for the gamma function but to play with the ES. The gamma function is the integral from 0 to ∞ of the function (see Figure 5a):

$$f(z) = t^{z-1} \cdot \exp(-t) \tag{3}$$

Using **time** as the integration variable, we construct the gamma function through the stepped parameter z. In fact, Figure 5b shows the so-called lower incomplete gamma function which is a function of the upper integration limit t. The gamma function corresponds to the last point in each curve, once the curves level off. These points are plotted in Figure 5c as a function of z. The results are compared with Stirling's approximation for the factorial function given by the following equation:

$$F(z) \sim (2\pi)^{1/2} \cdot z^{z-1/2} \cdot \exp(-z) \tag{4}$$

Although we have used the ES as an indirect integrator, we can also integrate the function f using the integration command **idt** directly, which can be obtained by clicking on the w port shown in Figure 5a.

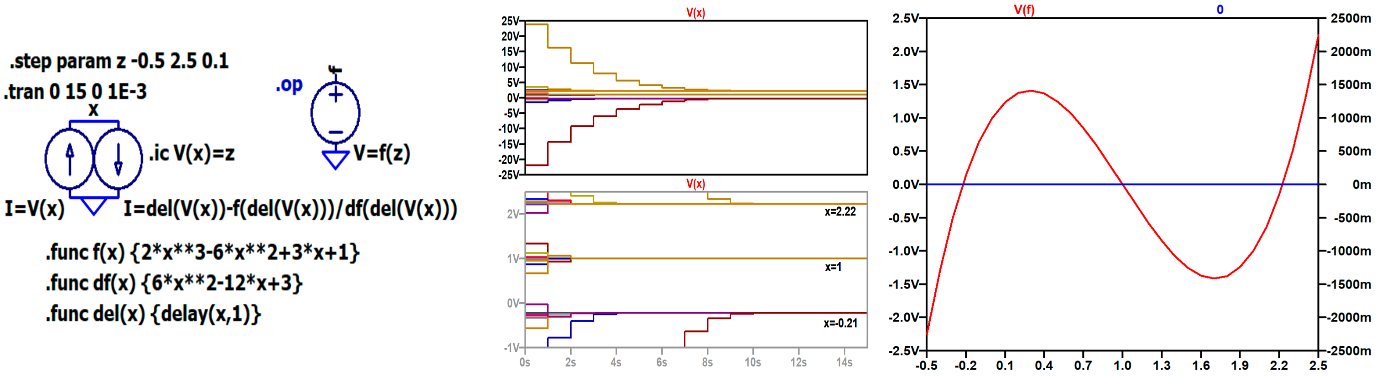


(**a**) (**b**) (**c**)

Figure 5. (**a**) ES for an indirect integration exercise. Colors indicate different *z* values (**b**) Plot of the incomplete gamma function (max x). (**c**) Gamma function (gamma) and comparison with Stirling's formula (Stirling).

2.4. Recursion: Explicit Newton–Raphson Method

In this exercise, the NR method is explicitly used [14]. The delay function is used to generate the recursive computation required to find the roots of a polynomial of degree 3 (Figure 6a). The roots of *f* are plotted in Figure 6b (zoomed plot). The stepped parameter *z* is used for shifting the initial condition of the iterative process. For completeness, a voltage plotter is included in order to generate the function graph. Figure 6c describes the analyzed function and indicates its roots.



(**a**) (**b**) (**c**)

Figure 6. (**a**) ES for an explicit evaluation of the NR method. Colors indicate different values of *z*. (**b**) Evolution of the iteration changing the initial condition of the problem. General view and detailed view. (**c**) Plot of the polynomial and its roots.

2.5. System of Linear Equations: Matrix Inversion

This simple algebra exercise consists in finding the inverse of a 2 × 2 matrix with the coefficients $b_{11} = 5$, $b_{12} = 2$, $b_{21} = -7$, and $b_{22} = -3$ (see Figure 7). This is nothing but the solution of a linear equation system. The exercise can easily be extended to higher dimensions. The solution is given by the matrix $a_{11} = 3$, $a_{12} = 2$, $a_{21} = -7$, and $a_{22} = -5$. The blue numbers on top of the ESs are obtained by clicking on the corresponding connection node. The method can also be adapted for the computation of the eigenvalues and eigenvectors of a matrix if an additional equation for the determinant (det[A-λI] = 0) is added.

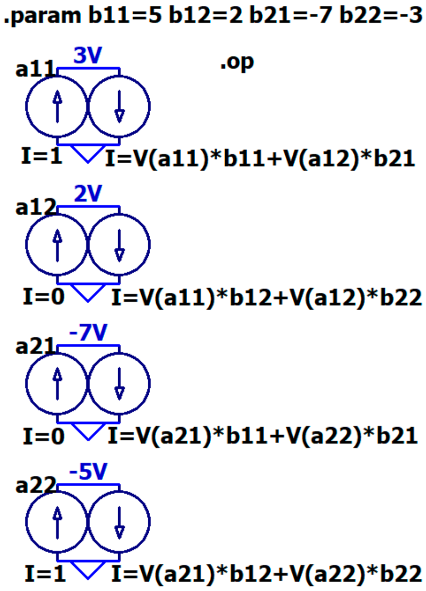
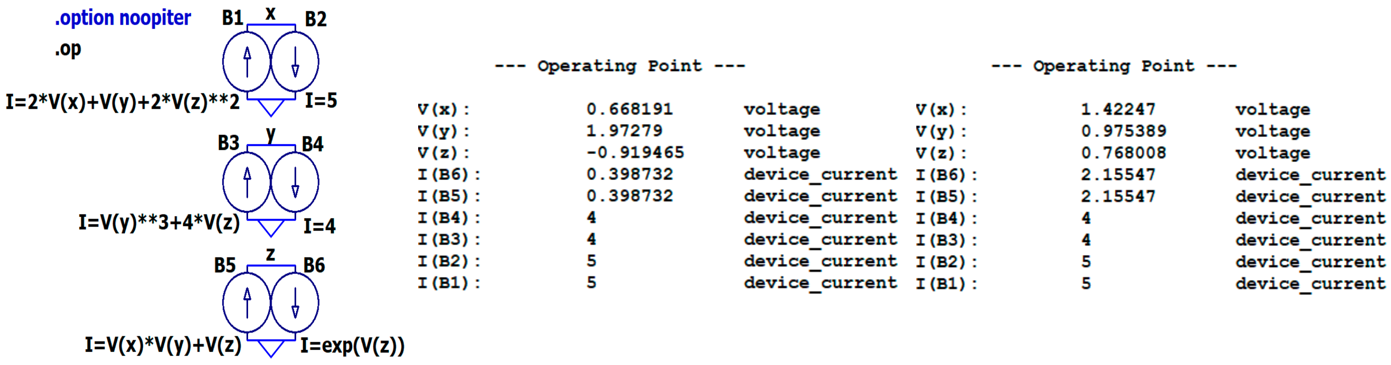


Figure 7. Inversion of a 2 × 2 matrix using a system of ESs.

2.6. System of Nonlinear Equations: Optimum Solution

In this example, a system of nonlinear equations needs to be solved. The optimum solution (minimum error) is looked for. As shown in Figure 8a, the problem consists of solving three coupled nonlinear equations with three unknowns. If solved under the **.op** directive, the solution provided by the simulator seems to be *a priori* correct (see Figure 8b) though a message in the Error Log window indicates that the "Direct Newton iteration failed to find .op point (Use .option noopiter to skip)". If the noopiter option is enable then, the solution shown in Figure 8c is obtained. This is the solution we were looking for since it provides the smallest error. The main message is that even if the solution seems to be correct (because of the currents matching), always substitute the solution into the equations and check the resulting error. The other alternative numerical methods provide the same optimum solution. Notice that, in this case, LTSpice does not require an initial guess for solving the problem. Instead, commercial software arrives at the same solution using the initial guess (1,1,0).



```
       --- Operating Point ---

V(x):          0.668191      voltage
V(y):          1.97279       voltage
V(z):          -0.919465     voltage
I(B6):         0.398732      device_current
I(B5):         0.398732      device_current
I(B4):         4             device_current
I(B3):         4             device_current
I(B2):         5             device_current
I(B1):         5             device_current
```

```
       --- Operating Point ---

V(x):          1.42247       voltage
V(y):          0.975389      voltage
V(z):          0.768008      voltage
I(B6):         2.15547       device_current
I(B5):         2.15547       device_current
I(B4):         4             device_current
I(B3):         4             device_current
I(B2):         5             device_current
I(B1):         5             device_current
```

(**a**) (**b**) (**c**)

Figure 8. (**a**) The MES applied to a nonlinear system of equations. (**b**) Approximate solution. (**c**) Correct solution.

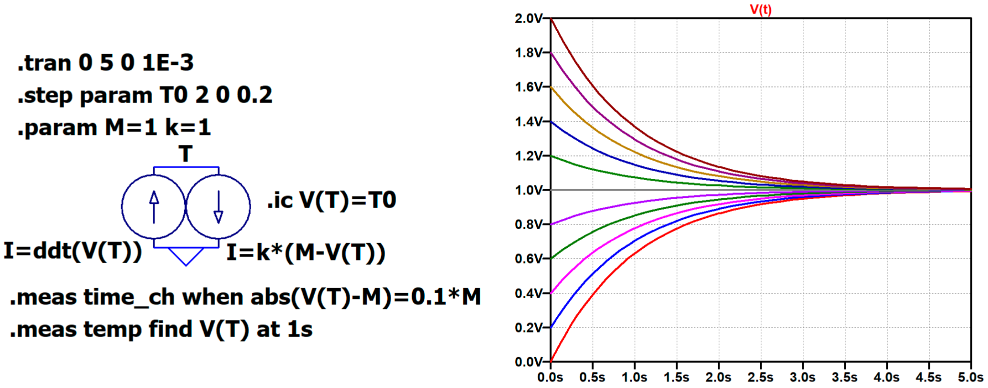
2.7. First-Order Autonomous Differential Equation: Newton's Law of Heat Transfer

With this simple exercise, we start the series of differential equation problems. In this case, we have to solve the first-order autonomous differential equation:

$$\mathrm{d}T/\mathrm{d}t = k\cdot(M - T) \tag{5}$$

which can be linked to Newton's law of cooling and heating [15]. In this context, t is the time, T is the temperature, M the equilibrium temperature, and k a positive constant.

Additionally, $T(t = 0) = T_0$ is the initial temperature ($T_0 > M$: cooling, $T_0 < M$: heating). Figure 9a illustrates the corresponding circuit schematic. **ddt** is the derivative with respect to time. Two post-processing directives are included here: one measures the time (time_ch) required for the system to reach a ±10% value of the equilibrium temperature and the other directive computes the temperature (temp) reached after 1 s (see Figure 9a's insets). Temperatures are plotted as a function of the initial condition T_0 in Figure 9b.



```
.tran 0 5 0 1E-3
.step param T0 2 0 0.2
.param M=1 k=1
.ic V(T)=T0
I=ddt(V(T))
I=k*(M-V(T))
.meas time_ch when abs(V(T)-M)=0.1*M
.meas temp find V(T) at 1s
```

(**a**) (**b**)

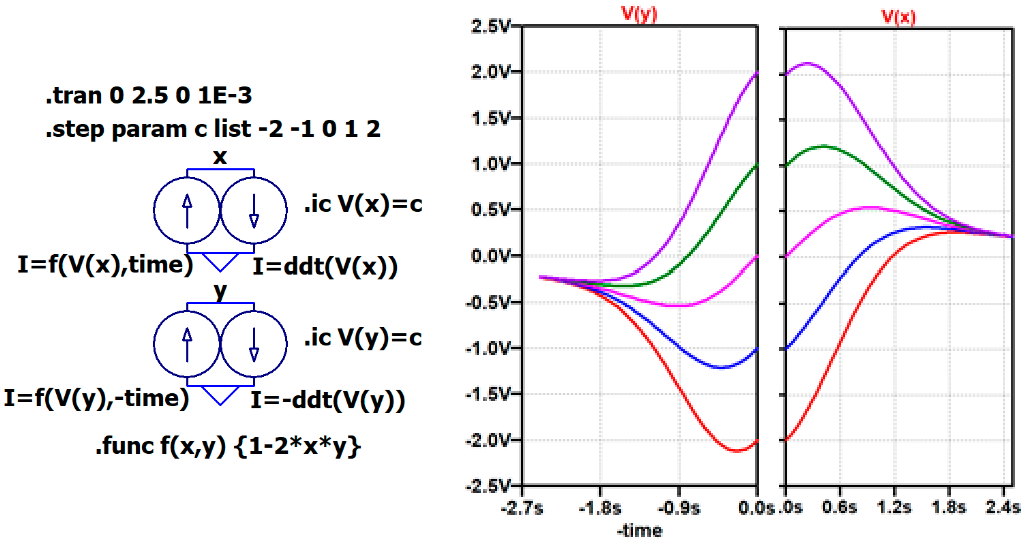
Figure 9. (**a**) Schematics for the Newton's law of heat transfer equation. Colors are for different vales of T0. t is the temperature T. (**b**) Solution to the differential as a function of the initial temperature equation.

2.8. First-Order Non-Autonomous Differential Equation: Redefinition of the Time Coordinate

In this case, the differential equation to be solved is as follows:

$$\mathrm{d}y/\mathrm{d}x = 1 - 2\cdot y\cdot x \tag{6}$$

Notice that not only the function y but also the variable x appear now in the right-hand side of (6). In order to solve this equation, we will use the variable **time** as the x coordinate for positive values of x and −**time** for negative values of x. The corresponding circuital diagram is shown in Figure 10a. Notice that the derivative with respect to time must also be reversed. The solutions for positive and negative values of x as a function of the integration constant c are reconstructed in Figure 10b.



```
.tran 0 2.5 0 1E-3
.step param c list -2 -1 0 1 2
.ic V(x)=c
.ic V(y)=c
.func f(x,y) {1-2*x*y}
```

(**a**) (**b**)

Figure 10. (**a**) Circuit diagram for solving a non-autonomous differential equation problem. Colors are for different values of c. (**b**) Solution to the differential equation as a function of the integration constant. Notice that time and -time are used as the coordinate x.

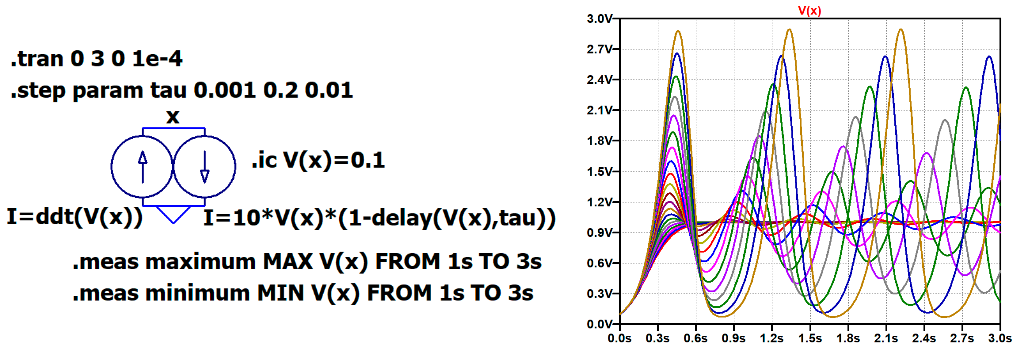
2.9. Delayed Differential Equation: Logistic Model

This exercise consists in finding the solution of the delayed logistic equation (Hutchinson's equation) [16]:

$$dx/dt = k \cdot x(t) \cdot [1 - x(t - \tau)] \tag{7}$$

where x is the magnitude of interest, t the time, k a constant, and τ the time delay.

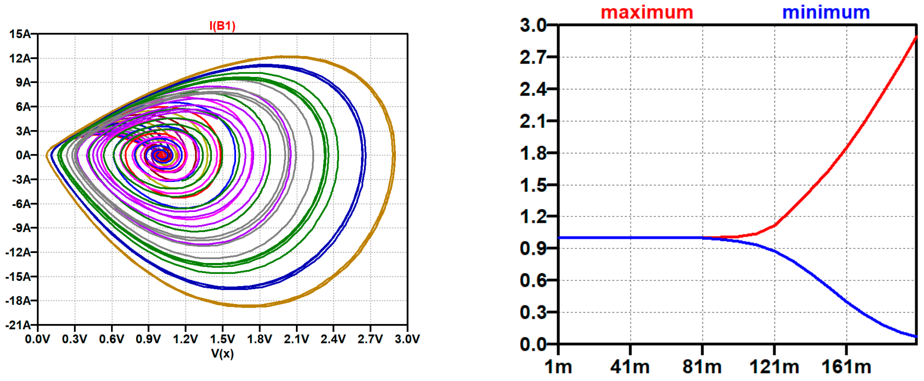
Equation (7) is frequently found in biological problems that require the representation of resource regeneration times, maturation periods, feeding times, reaction times, etc. Time delays introduce perturbations in the solution which deviate the system state from its equilibrium condition. The corresponding schematic is shown in Figure 11a along with its solution as a function of τ in Figure 11b. For $\tau \approx 0$, the logistic function (Verhulst's equation) is recovered. Figure 11c illustrates the evolution of the system state in the phase space using, as the derivative function, the current flowing through one of the generators. Notice that different behaviors are obtained: from closed orbits, there are decaying orbits to an attraction point. The schematic includes two measurement directives for the maximum and minimum of the oscillation amplitudes around the stationary state. Figure 11d shows the expected Hopf bifurcation (transition to a periodic solution) as a function of the delay time [17]. Measurements are taken after the transient regime.



```
.tran 0 3 0 1e-4
.step param tau 0.001 0.2 0.01
.ic V(x)=0.1
.meas maximum MAX V(x) FROM 1s TO 3s
.meas minimum MIN V(x) FROM 1s TO 3s
```

(**a**) (**b**)

Figure 11. *Cont.*



(**c**) (**d**)

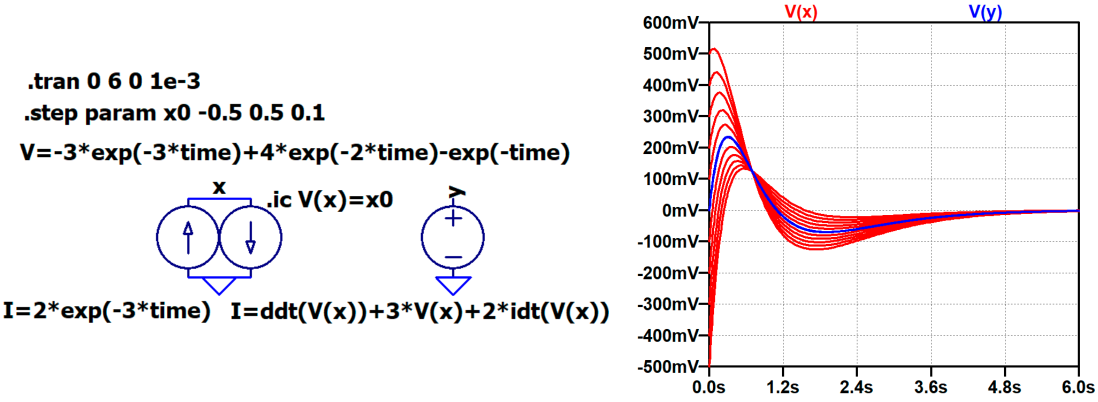
Figure 11. (**a**) Circuit diagram for solving the delay logistic equation. (**b**) Solution to the differential equation as a function of time. (**c**) Evolution of the system state in the phase space. Notice the orbits as well as the attraction point. (**d**) Hopf bifurcation indicating the appearance of an oscillatory behavior.

2.10. First-Order Integro-Differential Equation: When Integrals and Derivatives Meet

In this exercise, integrals and derivatives are combined in the same expression to form an integro-differential equation. This is a nonlinear problem which can be analytically solved as a sum of exponential functions. The equation to be solved is as follows:

$$2{\cdot}\exp(-3{\cdot}t) = \text{deriv}(x) + 3{\cdot}x + 2{\cdot}\text{integ}(x) \tag{8}$$

deriv() and integ() stand for the time derivative and integral, respectively. Figure 12a shows the corresponding circuit diagram using the ES, and Figure 12b shows the solution to the proposed equation as a function of the initial condition (red curves). The solution corresponding to the initial condition $x(t = 0) = 0$ is shown in blue.



```
.tran 0 6 0 1e-3
.step param x0 -0.5 0.5 0.1
V=-3*exp(-3*time)+4*exp(-2*time)-exp(-time)
.ic V(x)=x0
I=2*exp(-3*time)  I=ddt(V(x))+3*V(x)+2*idt(V(x))
```

(**a**) (**b**)

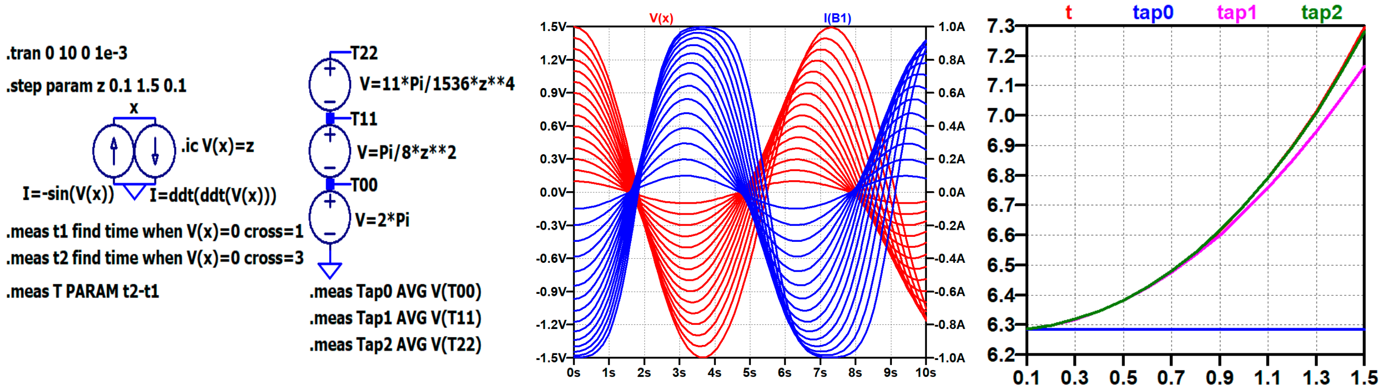
Figure 12. (**a**) Circuit diagram for solving the integro-differential equation. (**b**) Solution to the equation as a function of the initial condition. In blue the solution corresponding to x0 = 0.

2.11. Second-Order Differential Equation: Harmonic Oscillator

Second-order differential equations can also be solved using MES. This kind of problem requires two initial conditions, one for the function and one for the derivative. If the second derivative operator **ddt(ddt())** is used, then we just need to define the initial condition of f. The initial condition for the derivative is assumed to be null. This is the case illustrated in Figure 13a. The equation to be solved corresponds to the classical pendulum formula:

$$\mathrm{d}^2x/\mathrm{d}t^2 = -\sin(x) \tag{9}$$

where x is the angle with respect to the vertical. Notice that the equation is solved without invoking the linear approximation for the angular amplitude of the oscillation. The oscillation period T is measured for different initial angles. In addition, the numerical results are compared with approximations of increasing order (see the series combination of voltages sources). The results are illustrated in Figure 13b.



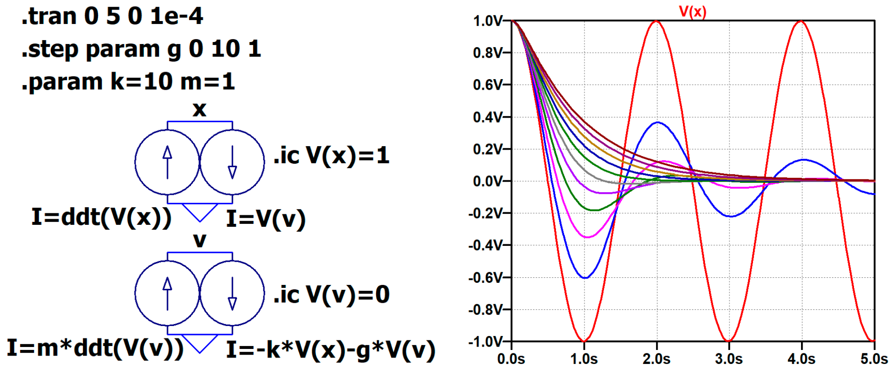
```
.tran 0 10 0 1e-3
.step param z 0.1 1.5 0.1
x
.ic V(x)=z
I=-sin(V(x))  I=ddt(ddt(V(x)))
.meas t1 find time when V(x)=0 cross=1
.meas t2 find time when V(x)=0 cross=3
.meas T PARAM t2-t1

T22
V=11*Pi/1536*z**4
T11
V=Pi/8*z**2
T00
V=2*Pi
.meas Tap0 AVG V(T00)
.meas Tap1 AVG V(T11)
.meas Tap2 AVG V(T22)
```

(**a**) (**b**) (**c**)

Figure 13. (**a**) Circuit diagram for solving the pendulum equation. (**b**) Solution to the equation and its second derivative as a function of the initial condition. (**c**) Computation of the oscillation period corresponding to approximations of increasing order as a function of the initial angle. In red, the numerical solution (x).

Instead, Figure 14a shows the case in which initial conditions are provided both for the function and the derivative. The example corresponds to a linear oscillator with a drag force proportional to the velocity:

$$\mathrm{m}\cdot \mathrm{d}^2 x/\mathrm{d}t^2 = -\mathrm{k}\cdot x - \mathrm{g}\cdot \mathrm{d}x/\mathrm{d}t \tag{10}$$



```
.tran 0 5 0 1e-4
.step param g 0 10 1
.param k=10 m=1
x
.ic V(x)=1
I=ddt(V(x))  I=V(v)
v
.ic V(v)=0
I=m*ddt(V(v))  I=-k*V(x)-g*V(v)
```

(**a**) (**b**)

Figure 14. (**a**) Circuit diagram for solving the oscillator equation with drag force. (**b**) Solution to the equation as a function of drag force magnitude. In red, the solution with no drag force.

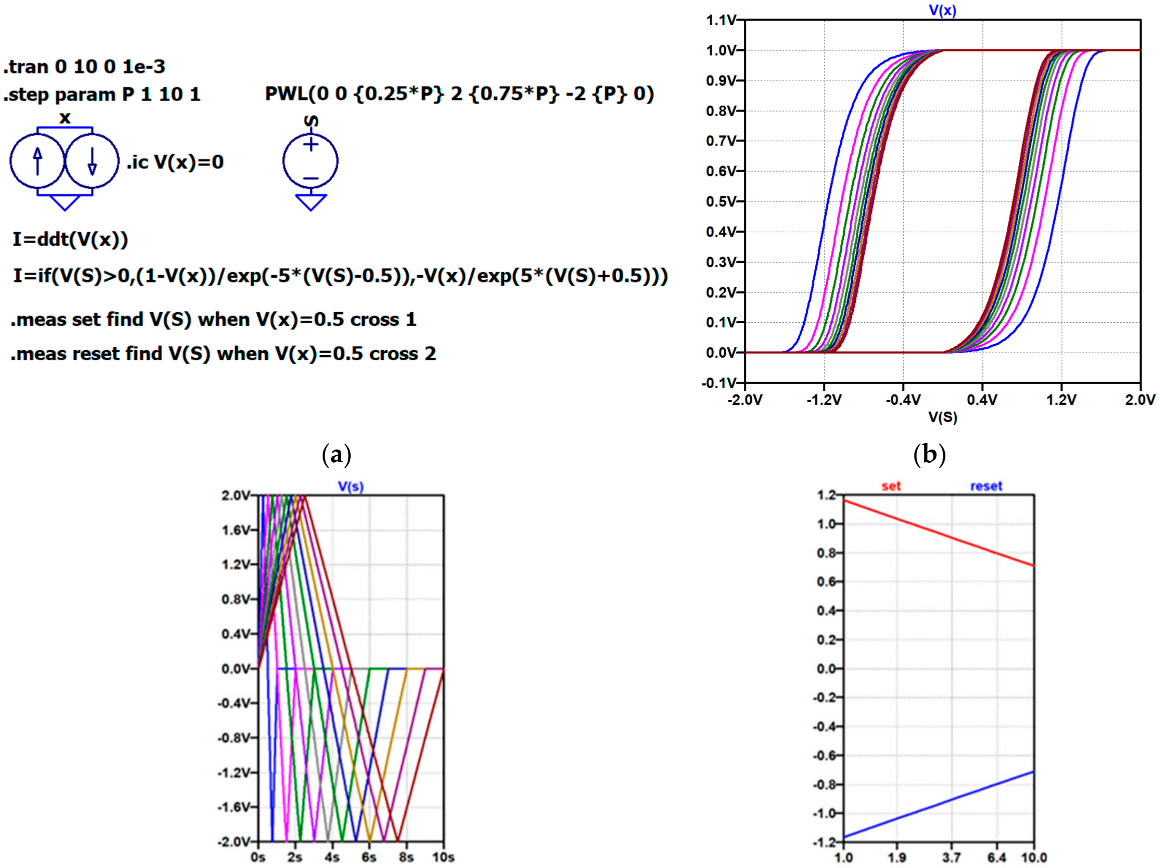
Notice that the second-order differential equation is decomposed now into two first-order differential equations. Figure 14b illustrates the cases corresponding to the free, underdamped, and overdamped responses. This problem is also associated with linear circuit theory (RLC circuits).

2.12. Differential Equation with Control Parameter: Frequency-Dependent Memristor Model

The memory state of a memristor (the so-called fourth element of circuit theory [18]) can be described by a balance-type differential equation of the following form [19]:

$$\mathrm{d}x/\mathrm{d}t = (1 - x)/\mathrm{T_S} - x/\mathrm{T_R} \tag{11}$$

where $\mathrm{T_S} = \exp[-\eta_\mathrm{S}\cdot(S - \mathrm{V_S})]$ and $\mathrm{T_R} = \exp[\eta_\mathrm{R}\cdot(S + \mathrm{V_R})]$ are called the set and reset characteristic times, respectively. η_S and η_R are called the transition rates, and $\mathrm{V_S}$ and $\mathrm{V_R}$ are called the set and reset voltages, respectively (see Figure 15a). Here, the memory state is driven by a triangular signal S (voltage) with a variable period P (see Figure 15c). A conditional expression in the memory equation expressed by the ES is introduced so as to avoid interference between positive and negative voltages.



```
.tran 0 10 0 1e-3
.step param P 1 10 1
PWL(0 0 {0.25*P} 2 {0.75*P} -2 {P} 0)
.ic V(x)=0
I=ddt(V(x))
I=if(V(S)>0,(1-V(x))/exp(-5*(V(S)-0.5)),-V(x)/exp(5*(V(S)+0.5)))
.meas set find V(S) when V(x)=0.5 cross 1
.meas reset find V(S) when V(x)=0.5 cross 2
```

(**a**) (**b**)

(**c**) (**d**)

Figure 15. (**a**) Circuit diagram for solving the balance-type differential equation for the memory state of a memristor. (**b**) Solution of the memory equation as a function of the signal period (hysteron plot). (**c**) Control parameter (voltage applied to the device). (**d**) Effect of the signal period on the set and reset edges (voltage) of the hysteron plot. The *x*-axis is in the log scale.

Figure 15b illustrates the hysteretic behavior of the memristor memory state as a function of the applied stimulus (control parameter). Figure 15c,d show that when the signal period increases (signal frequency decreases), the half-way memory state (S-value@$x = 1/2$) shifts towards lower values both for the set and reset processes. This means that if we kept the maximum voltage constant and we increase the frequency, then, the memristor behavior coincides with that of a resistor (collapse of the memory window).

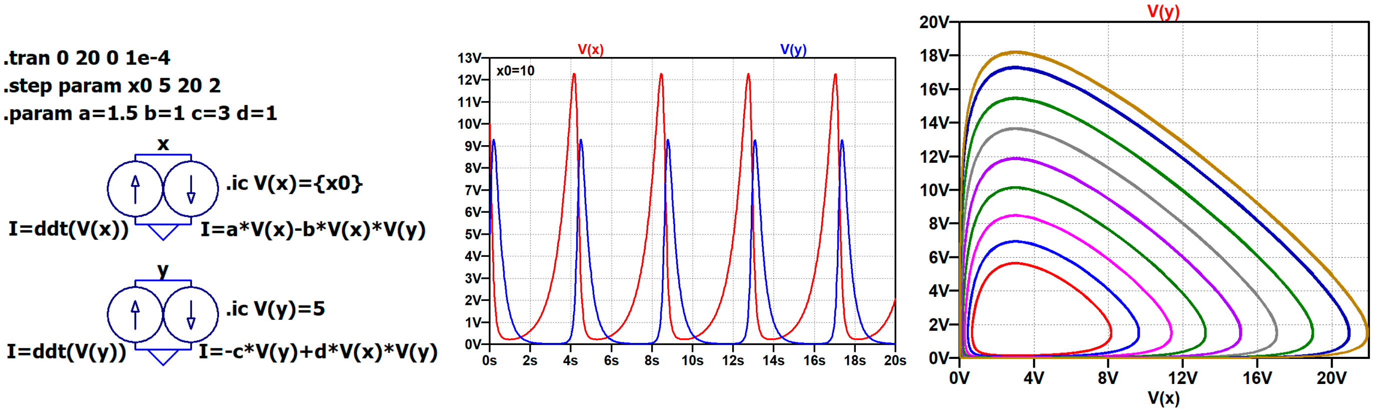
2.13. System of Nonlinear Differential Equations: Lotka–Volterra Model

The Lotka–Volterra equations are a system of first-order nonlinear differential equations that describe the dynamics of a biological system in which two species interact [20]. This is also called the predator–prey model. The population dynamics are described by a system of differential equations:

$$\mathrm{d}x/\mathrm{d}t = \mathrm{a}\cdot x - \mathrm{b}\cdot x\cdot y \tag{12}$$

$$\mathrm{d}y/\mathrm{d}t = -\mathrm{c}\cdot y + \mathrm{d}\cdot\mathrm{x}\cdot\mathrm{y} \tag{13}$$

where x represents the prey and y the predator. The derivatives describe the instantaneous growth rates of both populations. a, b, c, and d are positive constants (see Figure 16a). The solution corresponding to the initial condition $x(0) = 10$ is shown in Figure 16b. Figure 16c illustrates the orbits in the phase space.



```
.tran 0 20 0 1e-4
.step param x0 5 20 2
.param a=1.5 b=1 c=3 d=1
```

(**a**) (**b**) (**c**)

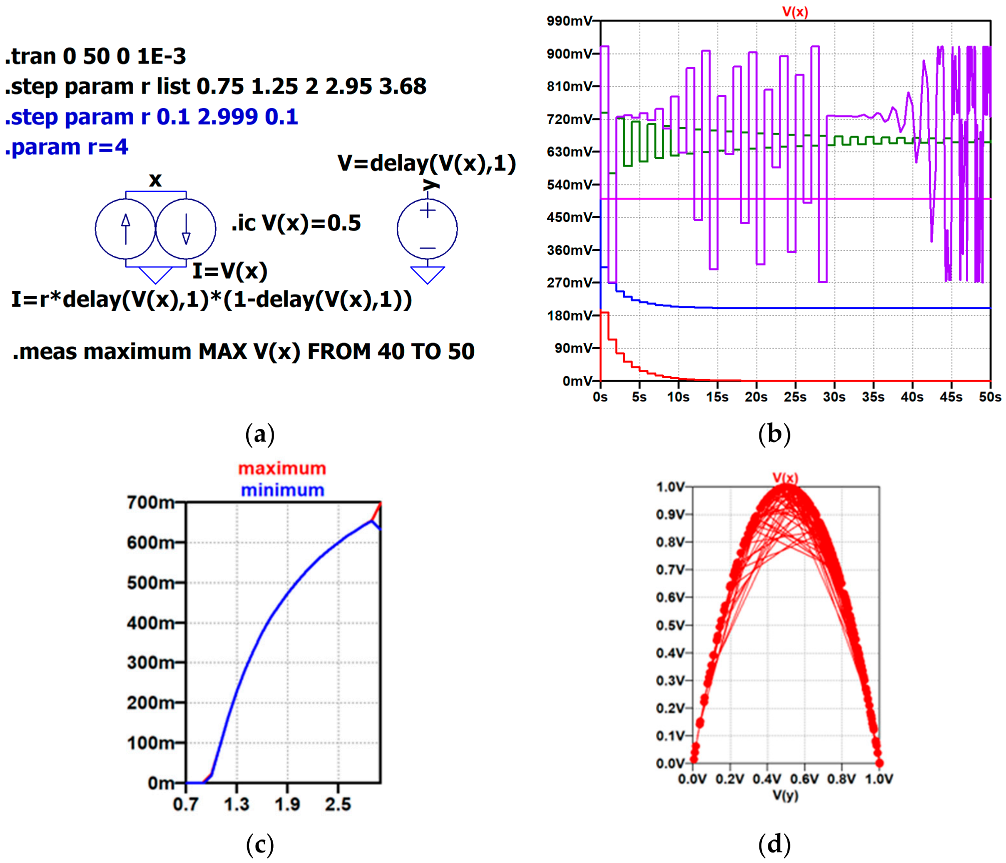
Figure 16. (**a**) Circuit diagram for solving the Lotka–Volterra model. (**b**) Temporal evolution of the prey and predator populations. (**c**) Orbits in the phase space as a function of the initial condition.

2.14. Recurrence Relation: Logistic Map

The logistic map is a recurrence relation often used as a reference to illustrate how complex chaotic behavior can arise from a simple nonlinear equation [21]. The logistic map is expressed as follows:

$$x_{\mathrm{n}+1} = r\cdot x_{\mathrm{n}}\cdot(1 - x_{\mathrm{n}}) \tag{14}$$

where x_{n} is a number between zero and one and r a positive number in the range [0,4]. As r changes, the amplitude and frequency content of the logistic map exhibits different behaviors. The iteration is carried out with the help of the delay function (see Figure 17a). Figure 17b shows the variety of behaviors than can be observed for different values of r. Figure 17c shows the onset of the first bifurcation. Unfortunately, the evolution of the chaotic behavior cannot be fully visualized using the proposed scheme. Nevertheless, some specific curves can indeed be obtained such as the 2D Poincaré plot for $r = 4$ (see Figure 17d). In this case, use **tran** 0 25 5 10E3 for a better resolution.



```
.tran 0 50 0 1E-3
.step param r list 0.75 1.25 2 2.95 3.68
.step param r 0.1 2.999 0.1
.param r=4
.ic V(x)=0.5
I=r*delay(V(x),1)*(1-delay(V(x),1))
I=V(x)
V=delay(V(x),1)
.meas maximum MAX V(x) FROM 40 TO 50
```

Figure 17. (**a**) Circuit diagram for solving the logistic map. (**b**) Evolution of the equation's solution as a function of the iteration number. (**c**) Onset of the first bifurcation as a function of r. (**d**) Poincaré map for r = 4.

2.15. System of Nonlinear Differential Equations: SIR Model with Vital Dynamics

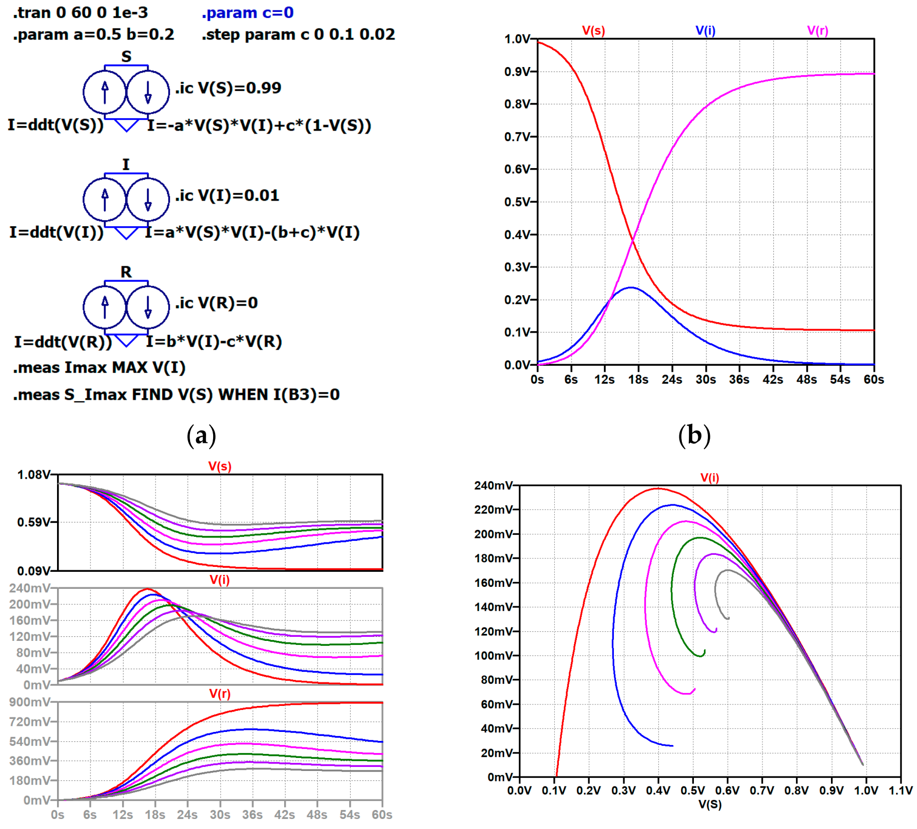
Mathematical models of epidemics are often expressed as a system of nonlinear differential equations. The SIR model represents the evolution of different population groups: *S*, susceptible; *I*, infected or infectious; and *R*, recovered [22]. Recovered people are considered immune so that they can no longer be infected again. The SIR model with vital dynamics includes birth (c in (15)) and natural death (c in (17)) processes. The equations (normalized) governing the whole process are as follows:

$$\mathrm{d}S/\mathrm{d}t = -\mathrm{a}\cdot S\cdot I + \mathrm{c}\cdot(1 - S) \tag{15}$$

$$\mathrm{d}I/\mathrm{d}t = \mathrm{a}\cdot S\cdot I + (\mathrm{b} + \mathrm{c})\cdot I \tag{16}$$

$$\mathrm{d}R/\mathrm{d}t = \mathrm{b}\cdot I - \mathrm{c}\cdot R \tag{17}$$

The model circuital diagram is shown in Figure 18a. Typical curves obtained with Equations (12)–(14) are illustrated in Figure 18b. The curves shown in Figure 18c were obtained by sweeping the vital dynamics parameter c. Figure 18d shows the infectious–susceptible plot and how its maximum shifts towards lower values as the maximum of *S* increases. This information is obtained through the measurement directives in the bottom line of the model diagram.



```
.tran 0 60 0 1e-3          .param c=0
.param a=0.5 b=0.2         .step param c 0 0.1 0.02
S
.ic V(S)=0.99
I=ddt(V(S))   I=-a*V(S)*V(I)+c*(1-V(S))
I
.ic V(I)=0.01
I=ddt(V(I))   I=a*V(S)*V(I)-(b+c)*V(I)
R
.ic V(R)=0
I=ddt(V(R))   I=b*V(I)-c*V(R)
.meas Imax MAX V(I)
.meas S_Imax FIND V(S) WHEN I(B3)=0
```

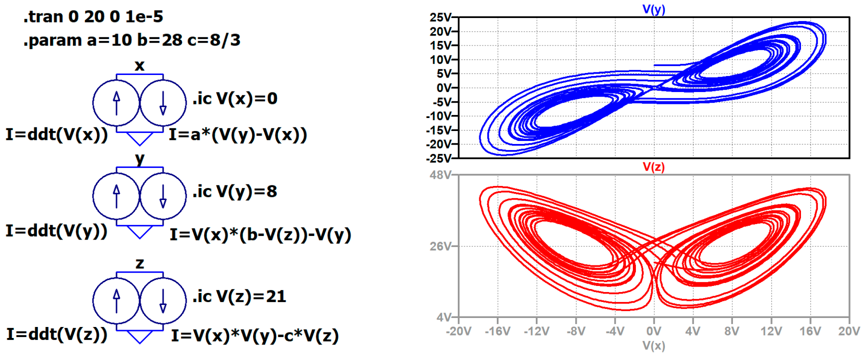
(**a**) (**b**)

(**c**) (**d**)

Figure 18. (**a**) Circuit diagram for solving the SIR model with vital dynamics. (**b**) Solution to the model equations. (**c**) Different model trajectories obtained by sweeping the vital dynamics parameter c. (**d**) Infectious-susceptible trajectories for different values of c.

2.16. System of Nonlinear Differential Equations: Lorenz Model

Another example of a coupled set of nonlinear time-dependent differential equations is that of the Lorenz system [23]. The Lorenz equations arise in a number of areas such atmospheric convection problems, lasers, dynamos, chemical reactions, electric circuits, etc. In particular, the Lorenz attractor is a set of chaotic solutions that when plotted in the phase space exhibits a typical "butterfly" plot. Figure 19a shows the model diagram for the three differential equations that constitute the Lorenz model. The trajectories obtained for a particular combination of parameters (those often used in the literature) are shown in Figure 19b.



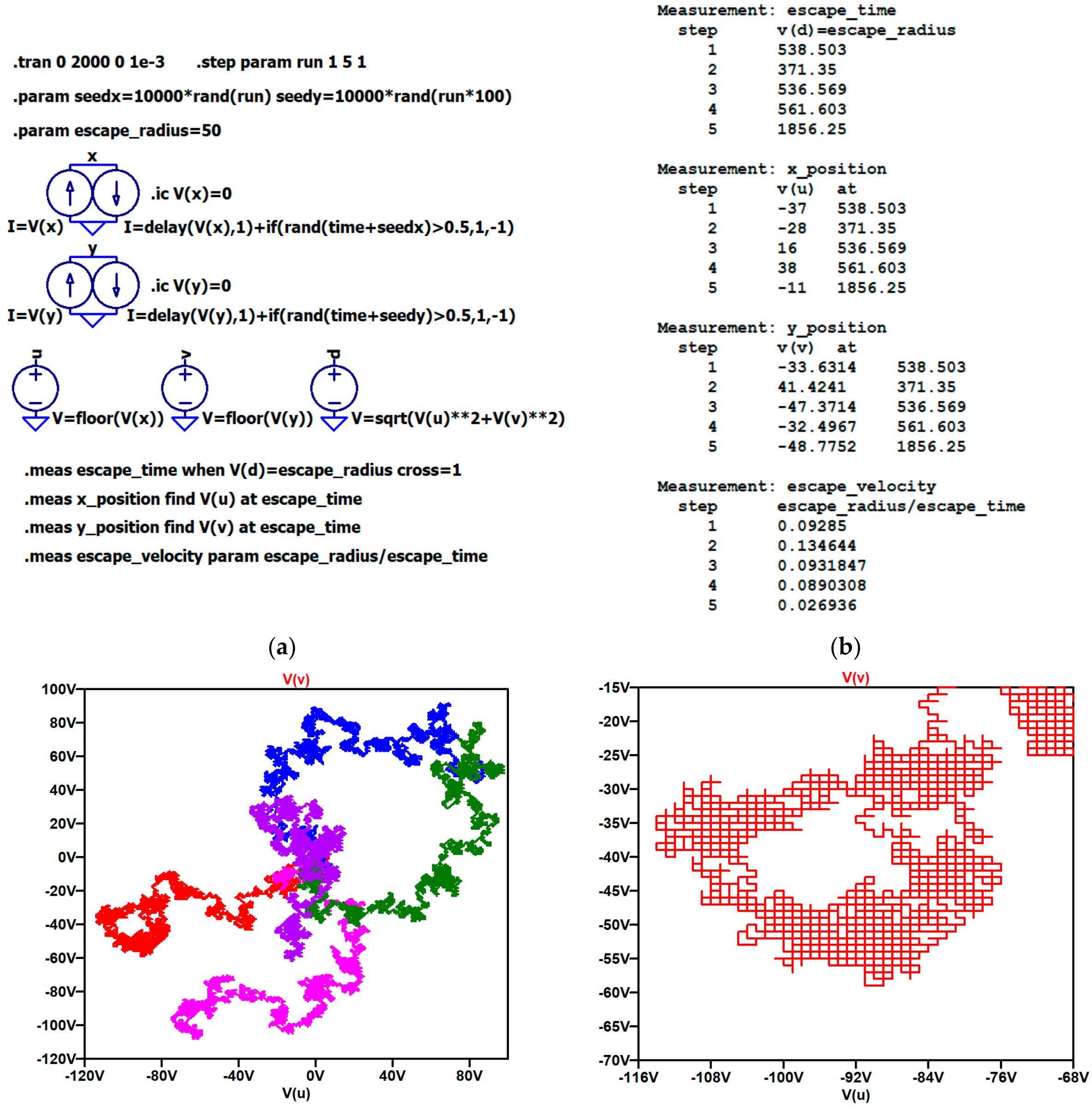
```
.tran 0 20 0 1e-5
.param a=10 b=28 c=8/3
x
.ic V(x)=0
I=ddt(V(x))   I=a*(V(y)-V(x))
y
.ic V(y)=8
I=ddt(V(y))   I=V(x)*(b-V(z))-V(y)
z
.ic V(z)=21
I=ddt(V(z))   I=V(x)*V(y)-c*V(z)
```

(**a**) (**b**)

Figure 19. (**a**) Circuit diagram for solving the Lorenz model. (**b**) Butterfly-like orbits obtained for a particular combination of parameters.

2.17. Discrete Fractal Curve: 2D Random Walk

A random walk is a stochastic process that describes a path in the space consistent in a succession of random steps [24,25]. In this case, a 2D random walk with steps ±1 was generated. Recurrence is carried out using the delay function again and uncorrelated randomness is introduced in the *x* and *y* directions through an *if* statement (see Figure 20a). Discretization of the variables is performed using the *floor* function. The model schematic calculates the escape time for a given distance (radius) as well as the location of the escape point. The escape velocity is defined as the escape radius divided by the escape time. All this information is obtained from the post-processing directives at the bottom of the model schematic and can be found in the "SPICE error log" window under the view tab (see Figure 20b). Figure 20c illustrates five different trajectories with origin at (0,0). Figure 20d shows a detail of one of the trajectories.



```
.tran 0 2000 0 1e-3     .step param run 1 5 1
.param seedx=10000*rand(run) seedy=10000*rand(run*100)
.param escape_radius=50
.ic V(x)=0
I=V(x)    I=delay(V(x),1)+if(rand(time+seedx)>0.5,1,-1)
.ic V(y)=0
I=V(y)    I=delay(V(y),1)+if(rand(time+seedy)>0.5,1,-1)
V=floor(V(x))    V=floor(V(y))    V=sqrt(V(u)**2+V(v)**2)
.meas escape_time when V(d)=escape_radius cross=1
.meas x_position find V(u) at escape_time
.meas y_position find V(v) at escape_time
.meas escape_velocity param escape_radius/escape_time
```

(**a**)

```
Measurement: escape_time
  step       v(d)=escape_radius
     1       538.503
     2       371.35
     3       536.569
     4       561.603
     5       1856.25

Measurement: x_position
  step       v(u)   at
     1       -37    538.503
     2       -28    371.35
     3       16     536.569
     4       38     561.603
     5       -11    1856.25

Measurement: y_position
  step       v(v)   at
     1       -33.6314     538.503
     2       41.4241      371.35
     3       -47.3714     536.569
     4       -32.4967     561.603
     5       -48.7752     1856.25

Measurement: escape_velocity
  step       escape_radius/escape_time
     1       0.09285
     2       0.134644
     3       0.0931847
     4       0.0890308
     5       0.026936
```

(**b**)

(**c**)

(**d**)

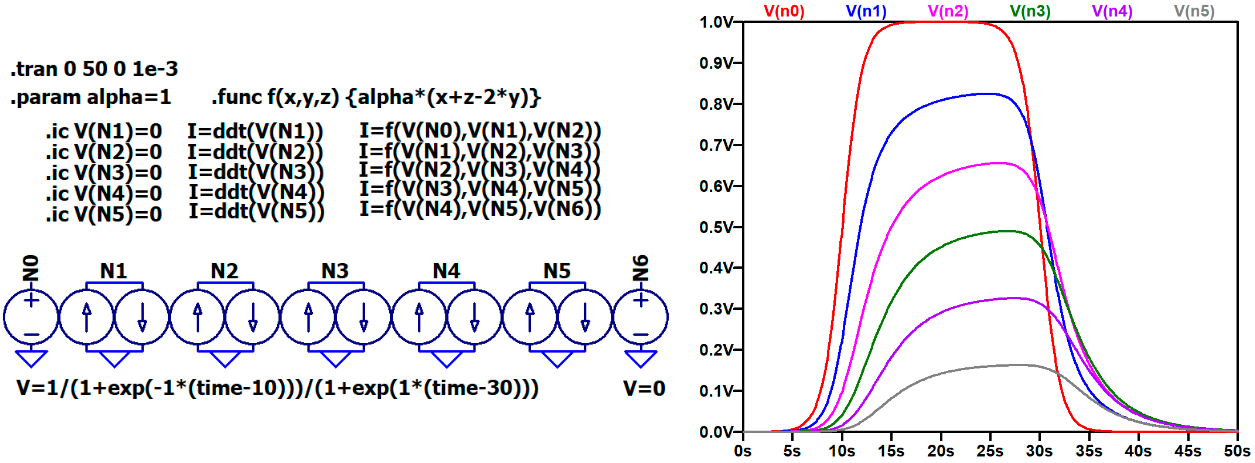
Figure 20. (**a**) Circuit diagram for a 2D random walk simulation. (**b**) Results of the measurement directives. (**c**) Generated random walks. (**d**) Detail of one of the random walks.

2.18. Parabolic Partial Differential Equation: 1D Heat Equation

In this example, the 1D heat equation is analyzed [26]. To solve this problem, we are going to use for the first time a 1D chain of ESs. This equation is strongly connected to other several problems including probability theory (Brownian motion) and financial mathematics (Black–Scholes equation). For instance, this problem can be associated with a rod heated at one extreme with the other extreme subjected to a fixed temperature. The equation can be modified so as to incorporate internal heat generation. The corresponding equation is given below:

$$\mathrm{d}T/\mathrm{d}t = \alpha \cdot \mathrm{d}^2T/\mathrm{d}x^2 \tag{18}$$

where α is a constant. In this exercise, we have assumed a temperature pulse with logistic rise and fall times for the boundary condition and null temperatures for the initial conditions (see Figure 21a). Each ES corresponds to a different location along the rod being heated at one extreme (N0). The second derivative was discretized using the central difference scheme. As shown in Figure 20b, the temperature increases and decreases at each node with a certain delay related to its position along the rod.



```
.tran 0 50 0 1e-3
.param alpha=1     .func f(x,y,z) {alpha*(x+z-2*y)}
   .ic V(N1)=0   I=ddt(V(N1))   I=f(V(N0),V(N1),V(N2))
   .ic V(N2)=0   I=ddt(V(N2))   I=f(V(N1),V(N2),V(N3))
   .ic V(N3)=0   I=ddt(V(N3))   I=f(V(N2),V(N3),V(N4))
   .ic V(N4)=0   I=ddt(V(N4))   I=f(V(N3),V(N4),V(N5))
   .ic V(N5)=0   I=ddt(V(N5))   I=f(V(N4),V(N5),V(N6))

V=1/(1+exp(-1*(time-10)))/(1+exp(1*(time-30)))          V=0
```

(**a**) (**b**)

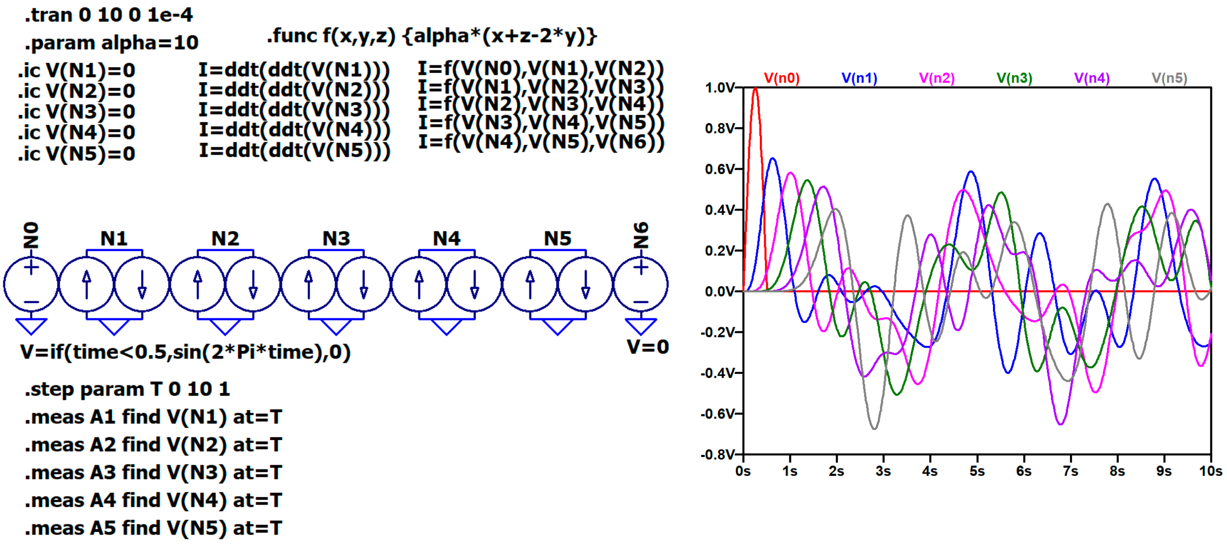
Figure 21. (**a**) Circuit diagram for a 1D rod subject to a temperature pulse at one extreme. (**b**) Solution of the differential equation at the different nodes.

2.19. Hyperbolic Partial Differential Equation: 1D Wave Equation

The wave equation describes the evolution of waves (sound, water, seismic, electromagnetics) as a function of space and time. The equation requires initial and/or boundary conditions depending on the problem we want to solve. The equation reads as follows:

$$\mathrm{d}^2A/\mathrm{d}^2t = \alpha \cdot \mathrm{d}^2A/\mathrm{d}x^2 \tag{19}$$

where A can represent the displacement of a string and α is a parameter related to the velocity of the wave. In our case, a sinusoidal pulse is applied to one extreme of the string (N0) while the opposite extreme is kept fixed (N6) (see Figure 22a). The solution at each node as a function of time is shown in Figure 22b. Notice that we have deliberately used a discretization step $h = 1$ (included in the constant α). Appropriate boundary conditions at N6 can represent the vibration of a free string. The displacement amplitude as a function of the position at various times (like a photograph of the string) can be found using post-processing directives.



```
.tran 0 10 0 1e-4
.param alpha=10
.func f(x,y,z) {alpha*(x+z-2*y)}
.ic V(N1)=0
.ic V(N2)=0
.ic V(N3)=0
.ic V(N4)=0
.ic V(N5)=0
I=ddt(ddt(V(N1)))
I=ddt(ddt(V(N2)))
I=ddt(ddt(V(N3)))
I=ddt(ddt(V(N4)))
I=ddt(ddt(V(N5)))
I=f(V(N0),V(N1),V(N2))
I=f(V(N1),V(N2),V(N3))
I=f(V(N2),V(N3),V(N4))
I=f(V(N3),V(N4),V(N5))
I=f(V(N4),V(N5),V(N6))
V=if(time<0.5,sin(2*Pi*time),0)
V=0
.step param T 0 10 1
.meas A1 find V(N1) at=T
.meas A2 find V(N2) at=T
.meas A3 find V(N3) at=T
.meas A4 find V(N4) at=T
.meas A5 find V(N5) at=T
```

(**a**) (**b**)

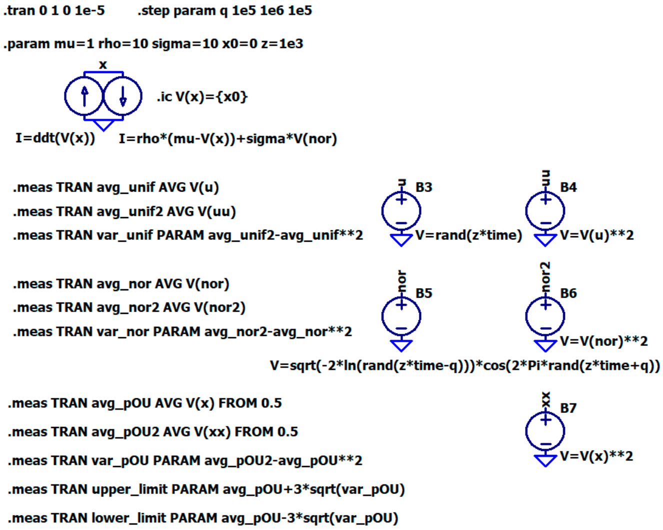
Figure 22. (**a**) Circuit diagram for the oscillations of a string with one extreme fixed. (**b**) Solution of the 1D wave equation at each node as a function of time.

2.20. Stochastic Differential Equation: Ornstein–Uhlenbeck Process

A stochastic differential equation (SDE) is a differential equation which includes a noise term [27]. The result is a stochastic process. SDEs find application in a plethora of fields including physics, probability, mathematical finance, evolutionary biology, etc. In physics, the Ornstein–Uhlenbeck (OU) process describes the velocity of a Brownian particle under the influence of friction. The process is also called mean-reverting since the trajectories tend to fluctuate around the mean value μ with reversion velocity ρ. The OU SDE is sometimes written as the Langevin equation:

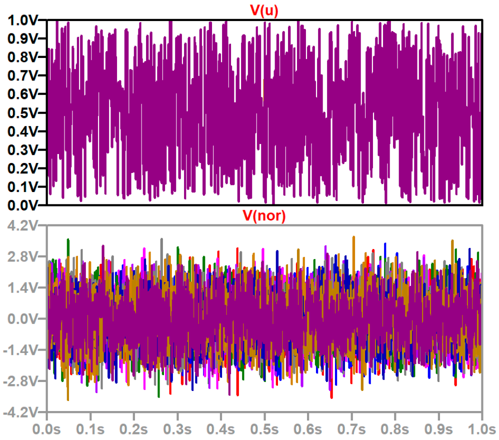
$$\mathrm{d}x/\mathrm{d}t = \rho\cdot(\mu - x) + \sigma\cdot\eta(t) \tag{20}$$

where σ is the magnitude of the fluctuations (volatility) and $\eta(t)$ is white noise (time derivative of a Wiener process). In the framework of SPICE, we identify (17) with a pseudo-OU (pOU) process since the differential time dt is not accessible. Caution should be exercised in considering dt as a fixed increment. We assume that $\eta(t)$ is a normally distributed random number with zero mean and unit variance. Figure 23a presents the corresponding ES: q is the random number generator seed, z is the sampling number per unit time (maximum timestep << 1/z), and x0 is the process initial value. Three sets of measurements are considered. They are used for calculating and checking the average value and variance of the individual processes. In all the cases, an auxiliary generator is used to calculate the square of the average value. In the case of a uniform distribution in the range [0,1], these values are 0.5 and 1/12. In the case of a normal distribution, these values are 0 and 1 (see Figure 22b). The normally distributed random numbers are calculated using the Box–Muller algorithm [28]. Again, care has to be taken with the generated data since they do not strictly follow a normal distribution (we use marked data points to see the correlations). The last directives calculate the statistical features of the pOU process including a ± 3 standard deviation confidence band (measured from 0.5 s to 1 s after the transient). The simulation results are shown in Figure 23c. Notice how the different trajectories fluctuate around the average value. This is the mean-reverting effect. A more detailed discussion about the solution of SDEs using MES will be reported elsewhere.

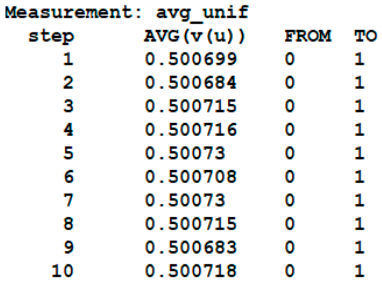


```
.tran 0 1 0 1e-5        .step param q 1e5 1e6 1e5

.param mu=1 rho=10 sigma=10 x0=0 z=1e3

.ic V(x)={x0}

I=ddt(V(x))    I=rho*(mu-V(x))+sigma*V(nor)

.meas TRAN avg_unif AVG V(u)
.meas TRAN avg_unif2 AVG V(uu)
.meas TRAN var_unif PARAM avg_unif2-avg_unif**2

.meas TRAN avg_nor AVG V(nor)
.meas TRAN avg_nor2 AVG V(nor2)
.meas TRAN var_nor PARAM avg_nor2-avg_nor**2

V=rand(z*time)
V=V(u)**2
V=V(nor)**2
V=sqrt(-2*ln(rand(z*time-q)))*cos(2*Pi*rand(z*time+q))

.meas TRAN avg_pOU AVG V(x) FROM 0.5
.meas TRAN avg_pOU2 AVG V(xx) FROM 0.5
.meas TRAN var_pOU PARAM avg_pOU2-avg_pOU**2
.meas TRAN upper_limit PARAM avg_pOU+3*sqrt(var_pOU)
.meas TRAN lower_limit PARAM avg_pOU-3*sqrt(var_pOU)

V=V(x)**2
```

(**a**)

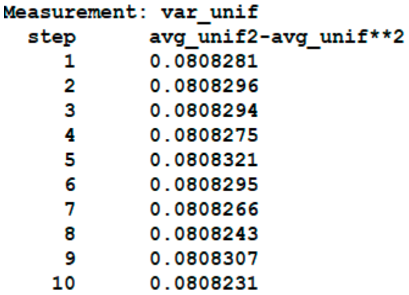


(**b**)



```
Measurement: avg_unif
  step        AVG(v(u))    FROM   TO
     1        0.500699     0      1
     2        0.500684     0      1
     3        0.500715     0      1
     4        0.500716     0      1
     5        0.50073      0      1
     6        0.500708     0      1
     7        0.50073      0      1
     8        0.500715     0      1
     9        0.500683     0      1
    10        0.500718     0      1
```

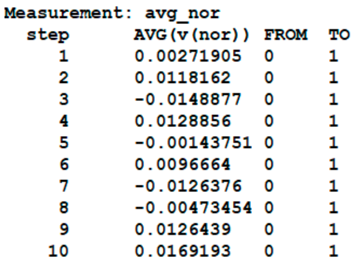
Average uniform distribution V(u) = 0.5



```
Measurement: var_unif
  step        avg_unif2-avg_unif**2
     1        0.0808281
     2        0.0808296
     3        0.0808294
     4        0.0808275
     5        0.0808321
     6        0.0808295
     7        0.0808266
     8        0.0808243
     9        0.0808307
    10        0.0808231
```

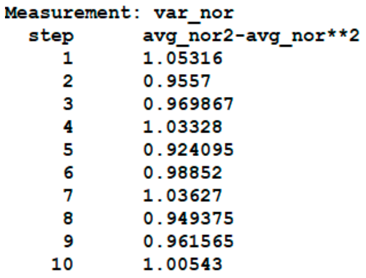
Variance standard distribution V(u) = 0.0833

Figure 23. *Cont.*



```
Measurement: avg_nor
  step        AVG(v(nor)) FROM   TO
     1        0.00271905  0      1
     2        0.0118162   0      1
     3        -0.0148877  0      1
     4        0.0128856   0      1
     5        -0.00143751 0      1
     6        0.0096664   0      1
     7        -0.0126376  0      1
     8        -0.00473454 0      1
     9        0.0126439   0      1
    10        0.0169193   0      1
```

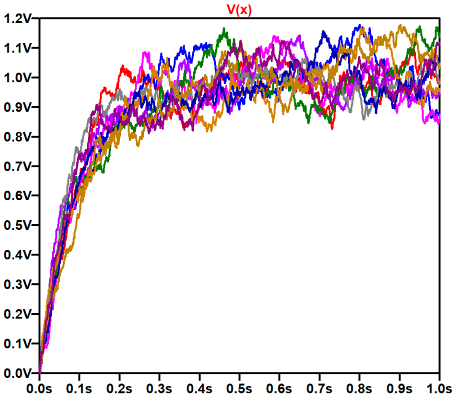
Average normal distribution V(nor) = 0

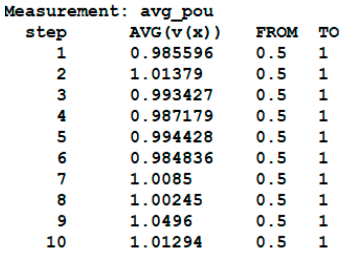


```
Measurement: var_nor
  step        avg_nor2-avg_nor**2
     1        1.05316
     2        0.9557
     3        0.969867
     4        1.03328
     5        0.924095
     6        0.98852
     7        1.03627
     8        0.949375
     9        0.961565
    10        1.00543
```

Variance normal distribution V(nor) = 1

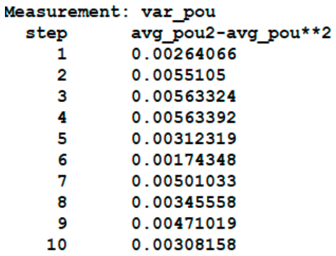
(**c**)





```
Measurement: avg_pou
  step        AVG(v(x))   FROM   TO
     1        0.985596    0.5    1
     2        1.01379     0.5    1
     3        0.993427    0.5    1
     4        0.987179    0.5    1
     5        0.994428    0.5    1
     6        0.984836    0.5    1
     7        1.0085      0.5    1
     8        1.00245     0.5    1
     9        1.0496      0.5    1
    10        1.01294     0.5    1
```

Average OU distribution V(x)



```
Measurement: var_pou
  step        avg_pou2-avg_pou**2
     1        0.00264066
     2        0.0055105
     3        0.00563324
     4        0.00563392
     5        0.00312319
     6        0.00174348
     7        0.00501033
     8        0.00345558
     9        0.00471019
    10        0.00308158
```

Variance OU distribution V(x)

Figure 23. (**a**) Circuit diagram for the pseudo-OU process. (**b**) Computation of the uniform and normal distributions including their averages and variances. (**c**) Simulation results for ten trajectories including a statistical description of the average value and variance of the associated process. These results can be used for calibrating the model parameters.

2.21. Control System: When Physics Meets Mathematics Meets Electronics

In this last example, we solve a physical problem using the method of elementary solvers and we combine it with electronics. This example is just for illustrative purposes and shows the potentiality of the proposed method when different aspects of a complex problem are combined.

Let us consider a 2D furnace with a liquid inside with its top edge open to the air (room temperature, *rt*) and the rest of the edges at a given temperature (maximum temperature, *maxt* or *rt*). The region of interest (blue region) is discretized as illustrated in Figure 24. We are not going to pay attention to details such as the discretization step or the properties of the liquid. The problem requires solving the heat equation with changing boundary conditions. The graphic approach is still valid although it is a bit cumbersome because the finite difference method in 2D needs to be used. Instead, the text-mode method is preferred here and of course is the best option when working with higher dimensions or larger problems. The idea here is that the temperature at the center of the furnace is measured using a constant current biased diode. The initial condition is room temperature *rt* everywhere inside the furnace. Then, the walls of the furnace are heated up to a temperature *maxt*. When the threshold temperature *thrt* is reached at the center of the structure (yellow spot), then the wall's temperature is set back to *rt*. When *T* drops below *thrt*, the heating source is switched on again. Of course, this is in the end a very simple control problem. The process dynamics is illustrated in Figure 25, where the liquid temperature, the wall's temperature, and the reference temperature are shown. The oscillations are a consequence of the thermal inertia of the system.

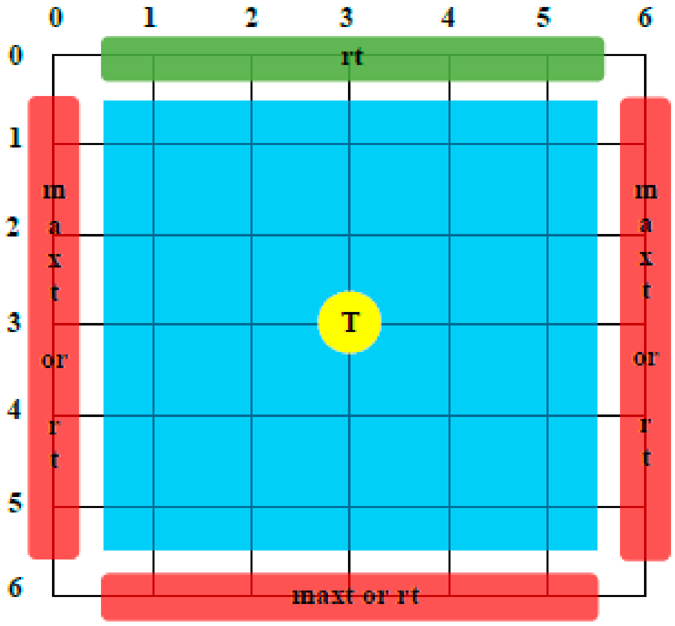


Figure 24. Two-dimensional furnace with its discretization mesh. The temperature of the lateral and bottom walls changes depending on the temperature measured at the center of the structure.

For completeness, Figure 26 shows the transductor circuit. The scheme illustrates the circuit and the fitting result for the voltage-temperature dependence of the particular diode considered (1N4148). LTSpice does not allow a dynamic change in the device temperature. Notice that the relationship is linear except at the highest temperatures. The temperature of the diode is finally found using an ES. The complete script is reported in Figure 27. The script can be further simplified using subcircuits, but it is presented this way for clarity. The script can be opened as .txt file in LTSpice and executed. No circuit schematic is required.

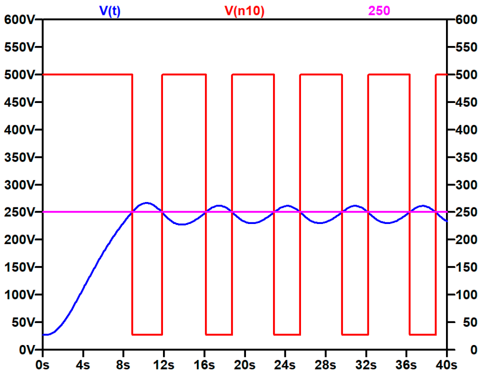


Figure 25. Evolution of the temperature at the center of the furnace. Notice how the wall's temperature is increased and decreased following a control process.

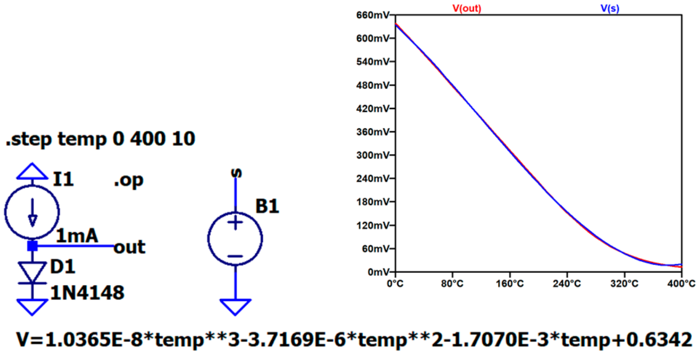
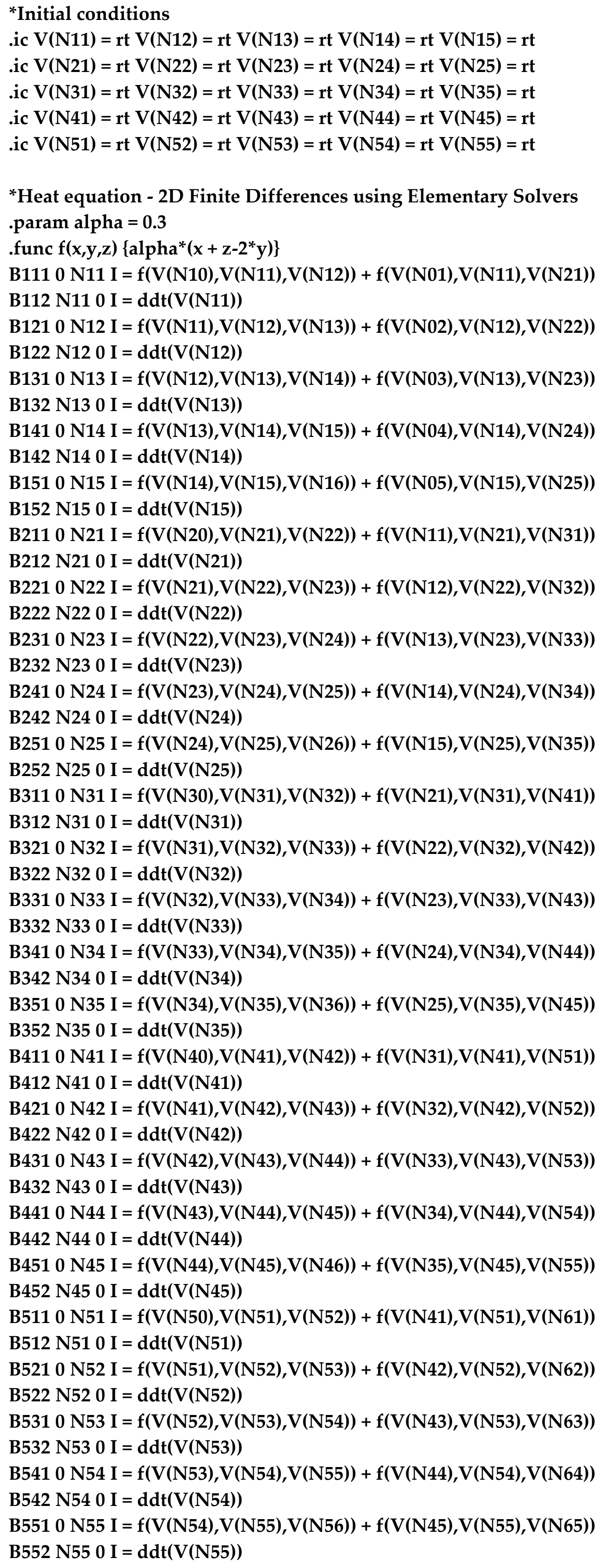


Figure 26. Transductor circuit. The plot illustrates the potential drop across the diode as a function of the temperature (red solid line). The blue line corresponds to the fitting model.

```
*Furnace control system

*Control specification
.param maxt = 500 thrt = 250 rt = 27
.func h(x) {if(V(T) < thrt,maxt,rt)}

*Boundary conditions
B01 N01 0 V = rt
B02 N02 0 V = rt
B03 N03 0 V = rt
B04 N04 0 V = rt
B05 N05 0 V = rt
B10 N10 0 V = h(V(T))
B20 N20 0 V = h(V(T))
B30 N30 0 V = h(V(T))
B40 N40 0 V = h(V(T))
B50 N50 0 V = h(V(T))
B16 N16 0 V = h(V(T))
B26 N26 0 V = h(V(T))
B36 N36 0 V = h(V(T))
B46 N46 0 V = h(V(T))
B56 N56 0 V = h(V(T))
B61 N61 0 V = h(V(T))
B62 N62 0 V = h(V(T))
B63 N63 0 V = h(V(T))
B64 N64 0 V = h(V(T))
B65 N65 0 V = h(V(T))
```

Figure 27. *Cont.*



```
*Initial conditions
.ic V(N11) = rt V(N12) = rt V(N13) = rt V(N14) = rt V(N15) = rt
.ic V(N21) = rt V(N22) = rt V(N23) = rt V(N24) = rt V(N25) = rt
.ic V(N31) = rt V(N32) = rt V(N33) = rt V(N34) = rt V(N35) = rt
.ic V(N41) = rt V(N42) = rt V(N43) = rt V(N44) = rt V(N45) = rt
.ic V(N51) = rt V(N52) = rt V(N53) = rt V(N54) = rt V(N55) = rt

*Heat equation - 2D Finite Differences using Elementary Solvers
.param alpha = 0.3
.func f(x,y,z) {alpha*(x + z-2*y)}
B111 0 N11 I = f(V(N10),V(N11),V(N12)) + f(V(N01),V(N11),V(N21))
B112 N11 0 I = ddt(V(N11))
B121 0 N12 I = f(V(N11),V(N12),V(N13)) + f(V(N02),V(N12),V(N22))
B122 N12 0 I = ddt(V(N12))
B131 0 N13 I = f(V(N12),V(N13),V(N14)) + f(V(N03),V(N13),V(N23))
B132 N13 0 I = ddt(V(N13))
B141 0 N14 I = f(V(N13),V(N14),V(N15)) + f(V(N04),V(N14),V(N24))
B142 N14 0 I = ddt(V(N14))
B151 0 N15 I = f(V(N14),V(N15),V(N16)) + f(V(N05),V(N15),V(N25))
B152 N15 0 I = ddt(V(N15))
B211 0 N21 I = f(V(N20),V(N21),V(N22)) + f(V(N11),V(N21),V(N31))
B212 N21 0 I = ddt(V(N21))
B221 0 N22 I = f(V(N21),V(N22),V(N23)) + f(V(N12),V(N22),V(N32))
B222 N22 0 I = ddt(V(N22))
B231 0 N23 I = f(V(N22),V(N23),V(N24)) + f(V(N13),V(N23),V(N33))
B232 N23 0 I = ddt(V(N23))
B241 0 N24 I = f(V(N23),V(N24),V(N25)) + f(V(N14),V(N24),V(N34))
B242 N24 0 I = ddt(V(N24))
B251 0 N25 I = f(V(N24),V(N25),V(N26)) + f(V(N15),V(N25),V(N35))
B252 N25 0 I = ddt(V(N25))
B311 0 N31 I = f(V(N30),V(N31),V(N32)) + f(V(N21),V(N31),V(N41))
B312 N31 0 I = ddt(V(N31))
B321 0 N32 I = f(V(N31),V(N32),V(N33)) + f(V(N22),V(N32),V(N42))
B322 N32 0 I = ddt(V(N32))
B331 0 N33 I = f(V(N32),V(N33),V(N34)) + f(V(N23),V(N33),V(N43))
B332 N33 0 I = ddt(V(N33))
B341 0 N34 I = f(V(N33),V(N34),V(N35)) + f(V(N24),V(N34),V(N44))
B342 N34 0 I = ddt(V(N34))
B351 0 N35 I = f(V(N34),V(N35),V(N36)) + f(V(N25),V(N35),V(N45))
B352 N35 0 I = ddt(V(N35))
B411 0 N41 I = f(V(N40),V(N41),V(N42)) + f(V(N31),V(N41),V(N51))
B412 N41 0 I = ddt(V(N41))
B421 0 N42 I = f(V(N41),V(N42),V(N43)) + f(V(N32),V(N42),V(N52))
B422 N42 0 I = ddt(V(N42))
B431 0 N43 I = f(V(N42),V(N43),V(N44)) + f(V(N33),V(N43),V(N53))
B432 N43 0 I = ddt(V(N43))
B441 0 N44 I = f(V(N43),V(N44),V(N45)) + f(V(N34),V(N44),V(N54))
B442 N44 0 I = ddt(V(N44))
B451 0 N45 I = f(V(N44),V(N45),V(N46)) + f(V(N35),V(N45),V(N55))
B452 N45 0 I = ddt(V(N45))
B511 0 N51 I = f(V(N50),V(N51),V(N52)) + f(V(N41),V(N51),V(N61))
B512 N51 0 I = ddt(V(N51))
B521 0 N52 I = f(V(N51),V(N52),V(N53)) + f(V(N42),V(N52),V(N62))
B522 N52 0 I = ddt(V(N52))
B531 0 N53 I = f(V(N52),V(N53),V(N54)) + f(V(N43),V(N53),V(N63))
B532 N53 0 I = ddt(V(N53))
B541 0 N54 I = f(V(N53),V(N54),V(N55)) + f(V(N44),V(N54),V(N64))
B542 N54 0 I = ddt(V(N54))
B551 0 N55 I = f(V(N54),V(N55),V(N56)) + f(V(N45),V(N55),V(N65))
B552 N55 0 I = ddt(V(N55))
```

Figure 27. *Cont.*

```
*Voltage@1mA for diode 1N4148 as a function of T
.func g(x) {1.0365E-8*x**3-3.7169E-6*x**2-1.7070E-3*x + 0.6342}

*Voltage divider output(resistor + diode)
I1 0 out 1m
R1 out 0 R = g(V(N33))/1m

*Voltage-temperature conversion using an Elementary Solver
B1 0 T I = g(V(t))
B2 T 0 I = V(out)

*Simulation time and timestep
.tran 0 40 0 1e-3
.backanno
.end
```

Figure 27. Model script for the 2D furnace and control process.

3. Conclusions

This paper reports a method for numerically solving an equation or a system of equations using a circuit simulator. In particular, LTSpice was considered in this work for its great versatility. The use of a basic circuit cell formed by two current sources connected in series, called an elementary solver, is demonstrated through the presentation of several examples. The simple and graphical way of programming are among the main advantages of the proposed method. The method of elementary solvers extends the range of applicability of circuit simulators to other areas of research and education.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/electronics13132480/s1.

Funding: The author acknowledges Project PID2022-139586NB-C41 funded by MCIN/AEI/10.13039/501100011033, Spain and FEDER, EU.

Data Availability Statement: All the examples are included in the Supplementary File.

Acknowledgments: The author thanks the engineer Alberto Villa for reviewing all the scripts reported in this work.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Zueco, J. An electric simulator to solve education engineering problems in fluid mechanics. *Comp. App. Eng. Educ.* **2013**, *21*, 748–757. [CrossRef]
2. Alfaki, H.; Dauda, M.; Gimba, A.; Ahmed, M. On modelling and simulation of electric circuit problems. *Malays. J. Com. Appl. Math.* **2020**, *3*, 21–26. [CrossRef]
3. Xie, Z.; Harrison, S.; Torti, S.; Torti, F.; Han, J. Application of circuit simulation method for differential modeling of TIM-2 iron uptake and metabolism in mouse kidney cells. *Front. Physiol.* **2013**, *4*, 52725. [CrossRef] [PubMed]
4. Madec, M.; Lallement, C.; Haiech, J. Modeling and simulation of biological systems using SPICE language. *PLoS ONE* **2017**, *12*, e0182385. [CrossRef] [PubMed]
5. Available online: https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html (accessed on 1 January 2023).
6. Gilles, B. *The LTSpice XVII Simulator: Manual, Methods and Applications*; Würth Elektronik: Niedernhall-Waldzimmern, Germany, 2022.
7. Asadi, F. *Essential Circuit Analysis Using Ltspice*; Springer: Cham, Switzerland, 2023. [CrossRef]
8. May, C. *Passive Circuit Analysis with Ltspice*; Springer: Cham, Switzerland, 2020. [CrossRef]
9. Svoboda, J. *Ltspice for Linear Circuits*; Wiley: Hoboken, NJ, USA, 2023.
10. Corless, R.M.; Gonnet, G.H.; Hare, D.E.G.; Jeffrey, D.J.; Knuth, D.E. On the LambertW function. *Adv. Comput. Math.* **1996**, *5*, 329–359. [CrossRef]
11. Winitzki, S. Uniform Approximations for Transcendental Functions. In *Computational Science and Its Applications—ICCSA 2003*; Kumar, V., Gavrilova, M.L., Tan, C.J.K., L'Ecuyer, P., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2667. [CrossRef]

12. Chen, C. A more accurate approximation for the gamma function. *J. Number Theory* **2016**, *164*, 417–428. [CrossRef]
13. Yang, Z.; Tian, J. An accurate approximation formula for gamma function. *J. Inequalities Appl.* **2018**, *2018*, 56. [CrossRef] [PubMed]
14. Deuflhard, P. *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*; Springer Series in Computational Mathematics; Springer International: Berlin/Heidelberg, Germany, 2004; Volume 35.
15. Grockenbach, M.; Schmidtke, K. Newton's law of heating and the heat equation. *Involve* **2009**, *2*, 419–437. [CrossRef]
16. Kashchenko, S. Asymptotics of the solutions of the generalized Hutchinson equation. *Autom. Control Comput. Sci.* **2013**, *47*, 470–494. [CrossRef]
17. Guckenheimer, J.; Myers, M.; Sturmfels, B. Computing Hopf Bifurcations I. *SIAM J. Numer. Anal.* **1997**, *34*, 1–21. [CrossRef]
18. Chua, L. Resistance switching memories are memristors. *Appl. Phys. A* **2011**, *102*, 765–783. [CrossRef]
19. Aguirre, F.L.; Suñe, J.; Miranda, E. SPICE implementation of the dynamic memdiode model for bipolar resistive switching devices. *Micromachines* **2022**, *13*, 330. [CrossRef] [PubMed]
20. Bartlett, M.; Hiorns, R.W. *Mathematical Theory of the Dynamics of Biological Populations: v.1*; Academic Press Inc.: Cambridge, MA, USA, 1973; Volume I.
21. May, R.M. Simple mathematical models with very complicated dynamics. *Nature* **1976**, *261*, 459–467. [CrossRef] [PubMed]
22. Okabe, Y.; Shudo, A. A mathematical model of epidemics—A tutorial for students. *Mathematics* **2020**, *8*, 1174. [CrossRef]
23. Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
24. Peitgen, H.; Jurgens, H.; Saupe, D. *Chaos and Fractals*; Springer: New York, NY, USA, 2004.
25. Popov, S. *Two-Dimensional Random Walk*; Cambridge University Press: Cambridge, UK, 2020.
26. Carslaw, H.S.; Jaeger, J.C. *Conduction of Heat in Solids, Oxford Science Publications*, 2nd ed.; The Clarendon Press, Oxford University Press: New York, NY, USA, 1988; ISBN 978-0-19-853368-9.
27. Oksendal, B. *Stochastic Differential Equations: An Introduction with Applications*, 6th ed.; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
28. Scott, D. Box-Muller transformation. *WIREs Comput. Stat.* **2011**, *3*, 177–179. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.