On the Discretization of Continuous-Time Chaotic Systems for Digital Implementations

Ashraf A. Zaher
Department of Electrical and Computer Engineering – College of Arts and Sciences
American University of Kuwait – PO Box 3323, Safat 13060, Kuwait
E-mail: azaher@auk.edu.kw

Abstract. Recently, many continuous-time chaotic systems were synthesized using microcontrollers and FPGAs. This requires applying mathematical discretization to convert integration into recursion. Depending on the approximation algorithms, the speed of the numerical processors, and the number of bits used to represent data, different accuracies and stabilities could be obtained. This article explores the conditions necessary to faithfully generate signals that reflect the true behavior of the chaotic systems, while maintaining the same values for their Lyapunov exponents. This is very important for chaos control and synchronization, especially for applications in secure communication that rely on digital cryptography. The Lorenz system and the Duffing oscillator are investigated to illustrate the effect of having autonomous versus non-autonomous structures. In addition, the Nosé-Hoover dynamical model is investigated to detect the relationship between agility and numerical accuracy. The obtained results prove that identical deterministic chaotic systems can behave differently, for the same set of initial conditions, depending on the discretization algorithm. This added sensitivity necessitates careful design of the mathematical models required for digital implementations of chaotic systems. The article concludes with useful recommendations for best practices in designing, synthesizing, and implementing digital chaotic systems, while commenting on the best compromise between mathematical complexity and numerical accuracy.

1. Introduction

Chaotic systems represent a special type of nonlinear systems that exhibit high sensitivity to initial conditions. Despite being deterministic, their performance looks quite similar to stochastic and random systems [1]. This particular feature turned chaos from being harmful into useful. Recently, many research fields emerged that make use of chaos synchronization to construct useful applications, especially in the field of secure communication [2,3]. Chaotic systems are usually characterized by having positive Lyapunov exponents that reflect the rate of the separation of their infinitesimally close trajectories, in phase space. The maximum positive Lyapunov exponent is considered the best indication of chaos, or hyperchaos in case of chaotic systems with high dimensionality [4].

Since analytical solutions to the ODEs describing the dynamics of the chaotic systems are not available, considerable efforts focus on numerical solutions [5]. Varieties of numerical simulations are currently available using computer programming, e.g. MATLAB or Python codes, or graphical modeling, e.g. Simulink. In this paper, optimizing the settings of the least order discretized approximation is addressed, for possible implementations using digital hardware.

The rest of this paper is organized as follows. Section 2 investigates the discretization effects on
both accuracy and speed of simulations, using different mathematical algorithms, with different settings. In addition, it explores three different chaotic systems to exemplify both autonomous and non-autonomous structures with different orders and characteristics. Section 3 summarizes the results obtained in this paper and provides a comprehensive conclusion regarding the findings of the current study, in addition to recommendations for best practices in the future.

2. Investigating discretization effects

Due to the rapid advances in technology, many applications that stem from both engineering and science require using digital equipment. Therefore, many continuous-time systems that are usually implemented and synthesized using analog components need to be approximated by digital equivalents. Chaos-based secure communication systems are, perhaps, among the most important candidates for migrating from analog to digital implementations. Discretization is required to achieve such goal, via using mathematical techniques to approximate the differential equations that represent the dynamics of the continuous-time systems, using simpler difference equations. This has the effect of converting difficult calculus-based integration algorithms into much easier algebraic-based recursion algorithms. Many algorithms are reported in the literature, e.g. Euler, Heun, and Runge-Kutta, just to name a few [6]. Such algorithms are usually categorized according to their order, which reflects the number of points used to represent the approximation. The higher the order, the better the accuracy is, but at the expense of mathematical complexity and, consequently, the speed of executing the simulations.

When using digital hardware, e.g. FPGAs, one is usually limited by the number of programmable units that can be used. Surely, this will directly affect the number of adders, multipliers, and other logic functions to be used. Memory elements, represented by Flip-Flops are also a limiting factor, when it comes to implementing recursive algorithms that need to gain access to the history of the used signals. Moreover, maximum clock speed, number of bits used to represent data, and number of ports used for external connections between the inputs and the outputs, are limiting factors that constrain the required implementation of the digitized algorithm [7].

2.1. The Lorenz System

This is a typical example of autonomous chaotic systems [8]. Usually it is considered a benchmark for evaluating new synchronization and control techniques for chaos-based applications. Its mathematical model takes the form:

\[ \begin{align*}
\dot{x} &= -\sigma (x - y) \\
\dot{y} &= \rho x - y - xz \\
\dot{z} &= xy - \beta z
\end{align*} \]  

(1)

where \( \sigma, \rho, \) and \( \beta \) are three positive constants. For generating chaos, they take the values, 10, 28, and 2.667, respectively [1]. To find the discrete-equivalent model, Euler forward shift operator method [9] is used, such that:

\[ s \approx \frac{s[n+1] - s[n]}{T_s} \]  

(2)

where \( s \) represents a generic signal that could be replaced by either \( x, y, \) or \( z, T_s \) is the integration step, and \( n \) is the sampling number. Following the approximation of equation (2), the equivalent discrete model for equation (1) takes the form:

\[ \begin{align*}
x[n+1] &= (1 - \sigma T_s)x[n] + \sigma T_s y[n] \\
y[n+1] &= \rho T_s x[n] + (1 - T_s), y[n] - T_s x[n]z[n] \\
z[n+1] &= T_s x[n]y[n] + (1 - T_s), x[n]z[n]
\end{align*} \]  

(3)

Figure 1 illustrates the Simulink block diagrams for the continuous-time case, of equation (1), and the discrete-time case, of equation (3), in (a) and (b), respectively. A fixed value for \( T_s \) was used that is
adjusted internally, within the simulation parameters. For convenience, $T_s$ was generated as a variable in the discrete-time model to ease the connections of the block diagram. All initial condition for the three variables were set internally, within the integrators, corresponding to $x(0) = 1$ and $y(0) = z(0) = 0$. All signals were generated using 64-bit double format. The remaining MATLAB and Simulink simulation parameters were set to their default values.

Simulating the two systems, for a total of 100 s, and setting $T_s = 0.001$ s, produces the results, shown in figure 2. Examining figures 2-a and 2-b, the typical chaotic response of the Lorenz system is observed. However, when contrasting the two responses in figure 2-c, an obvious mismatch is observed, after $t \approx 42.80$ s, approximately. Although $T_s$ was chosen much smaller than the dominant time constant of the Lorenz system, the approximated discrete model is not producing a faithful replica of the original system. In all the subsequent figures, the subscripts “1” and “2” indicate the outputs of the continuous-time and the discrete-time systems, respectively.
In addition, examining the phase spaces of both the original continuous-time system and its approximated discrete-time system, in figures 3-a and 3-b, respectively, it is clear that the famous butterfly effect is produced. However, when comparing them to each other, in figure 3-c, they are shown to be almost identical for the first 42.80 s (illustrated in red solid line), and then completely different, with no sign of correlation between them (illustrated in blue dotted line). Simulating both systems for much longer times did not show any accumulation for the round-off errors, and the same response, outlined in figures 2-c and 3-c was always observed.

Figure 3. Comparative study between the phase spaces of the systems in figure 1.

An optimization is usually required to obtain the best compromise between a fast simulation, and an accurate result. Using different values for $T_s$, it was found that both systems produce unstable results, if $T_s > 0.0244$ s, which is considered to be the upper threshold for the integration step, for the continuous-time system. Figure 4-a illustrates this result, via showing that for $t < 19.52$ s, both systems were close to each other, though not identical (shown in red solid line), while after $t > 19.52$ s, they started to deviate from each other (shown in blue dotted line). By steadily lowering the value of $T_s$, better results were obtained and both systems exhibited almost identical response for longer periods of time, indicated by the red straight line, which has a unity slope. Figure 4-b depicts this result, for $T_s = 0.02$ s, which was found to be the optimal value, after which the straight line gets distorted, and below which no significant improvement in the simulation accuracy was observed. Figure 4-c illustrates an extreme case, when $T_s$ was set to 0.00001 s, showing slightly better match between the two systems, but at the expense of dramatically slowing down the simulation. This makes it impossible for real-time applications, and impractical for offline-asynchronous applications, as well. Both systems were in almost perfect agreement for $t < 40.01$ s, which is surprisingly less than the cases where $T_s$ was set to 0.001 s or 0.002 s, illustrated in figures 3-c and 4-b, respectively.

Figure 4. Optimizing the value of $T_s$. 

-20 -10 0 10 20
-20 -10 0 10 20
-20 -10 0 10 20

(a) (b) (c)
Compiling the results of figures 2, 3, and 4, indicates that the best compromise between accuracy and speed of the simulation was found to be equivalent to the case when $T_s = 0.002$ s. For real-time applications, it is required to use a real-time pacer to synchronize the simulation results with the real-time clock of the machine running the simulation. Decreasing the value of $T_s$ makes it harder to generate real-time results, as the simulation takes longer time to execute. In addition, this is too much dependent on the machine running the simulation. For the system at hand, it was found that real-time results could be generated up to $T_s = 0.001$ s, provided that the plotting functions are temporarily frozen.

Of course, for better simulation accuracy, it is required to use a higher order discrete model to represent the original continuous-time system. Consequently, Euler first order approximation, depicted in equation (2), is now replaced by the second order Heun method [10]. This method uses a two-step approximation, as illustrated in equation (4):

\[
\begin{align*}
\tau[n+1] &= \tau[n] + T_s \\
x[n+1] &= x[n] + T_s \left[ x'(n), x(n), y(n), z(n) \right] \\
y[n+1] &= y[n] + T_s \left[ y'(n), x(n), y(n), z(n) \right] \\
z[n+1] &= z[n] + T_s \left[ z'(n), x(n), y(n), z(n) \right]
\end{align*}
\]  

(4)

where the intermediate variables correspond to those generated by the first order Euler method. Now all the time derivatives of the variables $x$, $y$, and $z$, are evaluated at two points and their average is used to replace the corresponding variables of the Euler method. Unlike, equation (3), the closed-form solution for the Heun method is more complicated to be presented by a Simulink block diagram, with easy and readable connections between all the blocks. Therefore, a MATLAB script was written to run the simulations, using simple recursion. As expected, and since the Heun method is more accurate, it was found that larger integration steps could be applied, while maintaining stable simulations. The maximum value of $T_s$ was found to be 0.066 s, which is 2.7 times that of the Euler method.

For more accuracy, the fourth order Runge-Kutta (RK4) method is usually used. This method is the default for running simulations for many dynamical systems and it can be also formulated in sixth order format to address more complex systems with stiff and complicated nonlinearities, as well as heavily coupled MIMO systems [10]. For the RK4 method, stable, but distorted simulations were possible, up to $T_s = 0.107$ s. Figure 5 illustrates the differences in the phase spaces for the Euler ($T_s = 0.0244$ s), Heun ($T_s = 0.066$ s), and RK4 ($T_s = 0.107$ s) methods, at the edge of instability, in (a), (b), and (c), respectively. Obviously, Euler method resulted in the smoothest response, while RK4 resulted in the most distorted response. However, for the same value of $T_s$, RK4 resulted in the most accurate response that was very close to that of Heun method. Thus, to achieve the best accuracy and the fastest simulation, Heun method is the optimal choice, provided a relatively small value of $T_s$ is used.

![Figure 5](image-url)
2.2. The Duffing oscillator:
This is a typical example of nonautonomous chaotic systems [2,11]. Its dynamics has many different forms, depending on the type of the external excitation and the arrangement of its parameters. It has been widely used to model many physical systems [12]. The following second order model is used:

\[
\begin{align*}
    \dot{x} &= y \\
    \dot{y} &= px - x^3 - ay + q \cos(\omega t)
\end{align*}
\]

where the two positive parameters \( a \) and \( p \) represent the damping and the stiffness of the Duffing oscillator, respectively. In addition, \( q \) and \( \omega \) represent the amplitude and frequency of the external sinusoidal excitation, respectively. For generating a chaotic performance, \( a, p, q, \) and \( \omega \) assume the values 0.4, 1.1, 1.8, and 1.8, respectively. Again, applying Euler approximation, depicted in equation (2), the following discrete model is obtained:

\[
\begin{align*}
    x[n+1] &= x[n] + T_s x[n] \\
    y[n+1] &= pT_s x[n] - T_s x^3[n] + (1 - aT_s) y[n] + qT_s \cos(\omega T_s)
\end{align*}
\]

which could be easily converted into an equivalent Simulink block diagram. Figure 6 illustrates the original continuous-time model and its approximated discrete-time model, in (a) and (b), respectively.

![Figure 6](image)

(a) Continuous-time Simulink model of the Duffing oscillator.
(b) Discrete-time Simulink model of the Duffing oscillator.

Simulating the two systems \([x(0) = y(0) = 0]\), as shown in figure 6, using \( T_s = T/1,000, \) where \( T \) is the periodic time for the sinusoidal driving function results in the response, illustrated in figure 7. Again, a mismatch between the two responses is observed after \( t = 11.03 \) s, despite the small value of \( T_s \). In addition, it was observed that setting \( T_s = T/10,000, \) resulted in shifting the beginning of the mismatch to \( t \approx 22.69 \) s. The Duffing oscillator exhibits a similar performance to the Lorenz system, as decreasing \( T_s \) beyond a certain threshold results in no significant improvement in the response. Again, the phase spaces look similar, as illustrated in figure 8, for the case corresponding to \( T_s = T/1,000. \)

![Figure 7](image)

Figure 7. Comparative study between the responses of the systems in figure 6.
Due to the fact that the Duffing oscillator has less order than the Lorenz system, it was possible to achieve real-time compatibility for simulation times up to $T_s \approx 0.0008\text{s}$, which is slightly less than that of the Lorenz system. In addition, the maximum $T_s$ for obtaining stable performance, when applying the Euler approximation was found to be $T_s \approx 0.055\text{s}$, which is surprisingly much less that $T$. Figure 9 illustrates the case when $T_s = 0.055$, showing no existence of the straight line relationship for both the variable $x$ and $y$. Figure 10 illustrates the case corresponding to a very small value for $T_s$, showing no significant improvement, despite having matching responses for 22.69 s, approximately.
Following the same procedure, outlined in the previous section, using Heun algorithm is expected to result in better accuracy. Applying Heun approximation to equation (5) results in:

\[
\begin{align*}
\hat{t}[n+1] &= \hat{t}[n] + T_s \\
\hat{x}[n+1] &= x[n] + T_s \left[ \hat{x}(t[n], x[n], y[n]) \right] \\
\hat{y}[n+1] &= y[n] + T_s \left[ \hat{y}(t[n], x[n], y[n]) \right] \\
x[n+1] &= x[n] + 0.5 T_s \left[ \hat{x}(t[n], x[n], y[n]) + \hat{x}(t[n+1], \hat{x}[n], \hat{y}[n]) \right] \\
y[n+1] &= y[n] + 0.5 T_s \left[ \hat{y}(t[n], x[n], y[n]) + \hat{y}(t[n+1], \hat{x}[n], \hat{y}[n]) \right]
\end{align*}
\]

(7)

where, again, the intermediate \( \hat{x} \) and \( \hat{y} \) variables correspond to those generated by Euler method. As expected, larger values for \( T_s \) could be used to generate stable simulations. Using Heun approximation, \( T_s \) could be increased up to 0.31 s, which could be further increased to 0.734 s, using RK4 method. Figure 11 illustrates the phase spaces for the Euler, Heun, and RK4, at the maximum \( T_s \), in (a), (b), and (c), respectively. Comparing figure 11 to figure 8, it is obvious that seeking speed of response comes at a very high price, as the distorted responses are quite far from the accurate result. Compared to the Lorenz system, the Duffing oscillator exhibited more sensitivity to the maximum \( T_s \). It is believed that the best compromise for this particular case is to use the Heun method, with \( T_s = T/50 \) s.

![Figure 11](image)

Figure 11. Deciding on the maximum value of \( T_s \) to obtain a stable performance for the Duffing oscillator.

2.3. The Nosé-Hoover system:

The last case study to be analyzed corresponds to the Nosé-Hoover system (N-H model) that has many applications in molecular physics [13]. This model uses an external excitation, representing the temperature that can be cast in many different forms. In addition, its mathematical model can be extended to include multiple states to correspond to a hyperchaotic system. The phase space of the N-H model can follow different attractors, with different Poincaré sections, for different initial conditions [14]. This interesting feature is unique and is not found in either the Lorenz system or the Duffing oscillator, which makes it more complicated to analyze and simulate. In this paper, the simplest form of the N-H model is considered; which is represented by:

\[
\begin{align*}
\dot{x} &= y \\
\dot{y} &= -x - yz \\
\dot{z} &= y^2 - T
\end{align*}
\]

(8)

where \( T \) represents the constant temperature, which is assumed to be unity. Applying Euler approximation, depicted in equation (2), the following discrete model is obtained:

\[
\begin{align*}
x[n+1] &= x[n] + T_s y[n] \\
y[n+1] &= -x[n] + y[n] - T_s y[n] z[n] \\
z[n+1] &= T_s y^2[n] + z[n] - T_s T
\end{align*}
\]

(9)
Figure 12 illustrates the original continuous-time N-H model and its approximated discrete-time model, in (a) and (b), respectively.

Assuming unity initial conditions for $x(0)$, $y(0)$, and $z(0)$, and setting $T_s = 0.01$ s, the simulation results for the system in figure 12-a are obtained for the Euler, Heun, and RK4 methods, in (a), (b), and (c), respectively. The results, depicted in figures 13-b and 13-c, show great similarity between both Heun and RK4 methods, which are identical to the correct result [14]. On the other hand, the result in figure 13-a is different, and exhibits a limit cycle, instead of the expected chaotic response. The simulation was run for 2,000 s and only the last 1,000 s were plotted in figure 13. This means that having stable simulations, alone, is not enough to decide on the upper threshold of $T_s$; instead, there should be a reference solution to compare the simulation to it.

Another problem that needs to be checked is whenever the simulation is running for a long time, e.g. a real-time application that requires continuous operation. In such case, round-off errors might accumulate, even if signals were represented using sufficiently large number of bits. Although, Simulink used 64 bits to represent data, and $T_s$ was set to be 0.001 s, which is very small, compared to the dominant time constant of the N-H model, it was observed that allowing the simulation to run after 500 s, resulted in distorting the results and deviating from the correct attractor. This is illustrated in figure 14.
Figure 14-a shows the correct result, obtained using RK4, for $T_s = 0.01$ s. The responses of the Heun and Euler methods are shown in figure 14-b and 14-c, respectively, using $T_s = 0.001$ s. All responses looked almost similar for the first 500 s, and distortion happened thereafter. The simulation was run for a total of 2,000 s. This problem was not observed in the first two case studies, for the Lorenz system and the Duffing oscillator. Thus, depending on the complexity of the chaotic system at hand, careful analysis should be done to avoid arriving at wrong conclusions regarding thresholds and optimal values for the simulation parameters, especially $T_s$.

Following the two-step approximation for the Heun method, equation (10) is used to represent the new discretized N-H model:

$$
\begin{align*}
T[n+1] &= T[n] + T_s \\
\pi[n+1] &= \pi[n] + T_s \left[ x[n], x[n], y[n], z[n] \right] \\
\rho[n+1] &= \rho[n] + T_s \left[ x[n], x[n], y[n], z[n] \right] \\
\delta[n+1] &= \delta[n] + T_s \left[ x[n], x[n], y[n], z[n] \right]
\end{align*}
$$

Since the Euler approximation failed to generate a faithful replica for the original system, even while using small values for $T_s$, simulations using either Heun or RK4 algorithms should be used. For this system, a different conclusion was achieved, which is the RK4 algorithm, with $T_s = 0.01$ was the best setting for the simulations parameters to have accurate and stable results, while avoiding accumulation of the round off errors.

3. Discussion and conclusion

Simulation of three different chaotic systems was explored in this paper. MATLAB/Simulink package was used to conduct the simulations, using 64 bits for representing data. The Euler approximation is considered the simplest discrete-time model. It requires the minimum algebraic blocks to perform addition and multiplication, instead of integration that is required for the original continuous-time models. Therefore, it is the most suitable for implementation, using digital hardware, e.g. FPGAs. However, it was found that using Euler algorithm requires a very small sampling time to have a reasonable accuracy. Decreasing $T_s$ too much to guarantee accuracy will slow down simulations and make them impractical for applications that require real-time compatibility. Moreover, deciding on the upper threshold for $T_s$ might be tricky, as simulations can produce stable results that are very far away from the true performance. This was obvious in figure 13-a, when the N-H model exhibited a limit cycle, instead of the expected chaotic response. This is also possible, even with more accurate algorithms, due to the complicated nonlinearities of the chaotic systems.
Adjusting initial conditions is also important, before launching simulations. Knowledge of the range of the signals involved in the chaotic system is required, as some sets of initial conditions might force the simulator to fail, e.g. setting the initial conditions to all zeros for the Lorenz system will cause the simulation to stay at the origin, as this corresponds to an equilibrium point and the system is autonomous. This problem will not be observed in the Duffing oscillator, due to the existence of the external forcing function. Setting either of the variables of the Lorenz system or the Duffing oscillator to too high values will cause the simulator to diverge and to produce unstable results. For the N-H model, it might be even worse, as the shape of the attractor is strongly dependent on the initial conditions.

Adopting a programming or a graphical modeling approach is yet another important factor, when validating the simulation accuracy. Depending on the used software package, programming might be faster, but the script/code must address the recursion and the creation of the intermediate variables efficiently to avoid using too much memory. For graphical modeling, there will be an overhead involved when creating the necessary files for the simulation; thus, although easier, optimization of the simulation parameters and correct settings of each block are required. Choosing a target hardware for the digital implementation may dictate whether programming or graphical modeling is better. Many integrated design environments, such as MATLAB/Simulink, allow both options, where the necessary HDL code can be generated automatically from the script or the block diagram. When adopting a simple graphical modeling approach, Euler method should be the first choice. This requires using a small value for $T_s$, which could be easily accommodated, using current FPGA boards.

Generating the required HDL code that needs to be downloaded to the target FPGA can be done in many several ways. Depending on the used integrated development kit (IDK), different optimization methods could be applied to satisfy certain performance criteria [15-17]. To exemplify the synthesis and implementation phases of the discretized models of chaotic system, the EP2C70F896C6 FPGA that belongs to the Altera Cyclone II family was used to implement the Lorenz system, using the Euler method approximation that was discussed in the first case study. Figure 15 shows its typical layout and block diagram, in (a) and (b), respectively. The first method was done using the built-in HDL coder, which is an integral part of the MATLAB/Simulink package. The sampling time was set to 0.001 s and the constants data types of the system were converted to 16-bit fixed point format, using the Fixed-Point Tool in Simulink. The second method was done, using the IDK of Altera Quartus II, assuming the default 32-bit floating point (Single Precision) to represent data. Both generated VHDL codes that were run on ModelSim, resulting in an almost identical performance to the original continuous-time system. Although the two approaches were different, the performance was similar; however, looking at their analysis report reveals the structural differences between them that might be a limiting factor, when applying it in real-time scenarios. Optimizing the throughput of the implementation and accommodating real-time constraints are beyond the scope of this paper.

---

![Figure 15](image-url)  
**Figure 15.** The target FPGA and its board block diagram.
Tables 1 and 2 summarize the most important features and statistics of the two different designs, using the Simulink integrated HDL coder and the IDK of Altera Quartus II, for the synthesis and implementation of the Lorenz system, using Euler method approximation. In addition, figure 16 shows the typical ModelSim output for the last five samples (out of a total of 100,000 samples) of the three outputs of the discretized Lorenz system, \( x[n] \), \( y[n] \), and \( z[n] \), where \( n \) is the sample number, assuming a sampling rate of 1 KHz. Moreover, figure 17 shows the discrete signal \( x[n] \) of the Lorenz system, after converting the data into real numbers. It is obvious, that the result of figure 17 is similar to the results, obtained in figures 2-a and 2-b.

**Table 1.** Statistics for the Simulink-based HDL Coder

| Multipliers (16bits × 16bits) | 8 |
|------------------------------|---|
| 24×24-bit Adder             | 2 |
| 23×23-bit Adder             | 1 |
| 16×16-bit Subtractor        | 2 |
| 23×23-bit Subtractor        | 2 |
| 16-bit Registers            | 3 |

**Table 2.** Statistics for the IDK of Altera Quartus II

| Total logical elements | 248 |
|------------------------|-----|
| Total registers used   | 48  |
| 9-bit multiplier       | 10  |
| 16-bit multiplier      | 6   |
| 24×24-bit Adder        | 6   |
| 16×16-bit Adder/Subtractor | 4 |
| Total thermal power dissipation | 205.11mW |

**Figure 16.** ModelSim results for the last five samples (\( T_s = 0.001 \) s), of the Lorenz system.

**Figure 17.** ModelSim result for the 100,000 samples (\( T_s = 0.001 \) s), of \( x[n] \).
Finally, it should be highlighted that generating discrete-time models to approximate the continuous-time systems is becoming important and more demanding, due to the rapid advances in the technology and the limitations and disadvantages of the analog components. There is no unique answer to what the best discretization algorithm is; however, Euler and Heun methods should be the first choices for the discretization, due to their simplicity and the ease of synthesis, compared to the more accurate RK4 algorithm. However, this requires great effort for tuning the simulation parameters to arrive at the best settings, which can be very much dependent on the chaotic system at hand, the machine used to run the simulation, and the target digital hardware.

Acknowledgment
This work was supported by a research grant from the American University of Kuwait.

References
[1] J. C. Sprott, *Elegant Chaos*, World Scientific, 2010.
[2] A. Zaher, “Duffing Oscillators for Secure Communication”, *Computers and Electrical Engineering*, Vol. 71, 2018, pp. 77-92.
[3] A. Zaher and A. Abu-Rezq, “On the Designs of Chaos-Based Secure Communication Systems”, *Communications in Nonlinear Science and Numerical Simulation*, Vol. 6, No. 9, 2011, pp. 3721-3737.
[4] S. Boccalletti, J. Kurths, G. Osipov, D. Valladares, and C. Zhou, “The synchronization of chaotic systems” *Physical Reports*, Vol. 366, No. 1-2, 2002, pp. 1-101.
[5] S. Liao, “On the reliability of computed chaotic solutions of nonlinear differential equations”, *Tellus*, Vol. 61, No. 4, 2009, pp. 550-564.
[6] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*, 3rd Edition, Springer-Verlag, 1992.
[7] S. Liao, “On the clean numerical simulation (CNS) of chaotic dynamic systems”, *Journal of Hydrodynamics, Ser. B*, Vol. 29, No. 5, 2017, pp. 729-747.
[8] S. A. Eftekhari and A. A. Jafari, “Numerical simulation of chaotic dynamical systems by the method of differential quadrature”, *Scientia Iranica*, Vol. 19, No. 5, 2012, pp. 1299-1315.
[9] W. Song and J. Liang, “Difference Equation of Lorenz System”, *International Journal of Pure and Applied Mathematics*, Vol. 83, No. 1, 2013, pp. 101-110.
[10] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, Wiley, 2016.
[11] I. Kovacic and M. J. Brennan, *The Duffing Equation: Nonlinear Oscillators and Their Behavior*, Wiley, 2011.
[12] R. Zivieri, S. Vergura, and M. Carpentieri, “Analytical and numerical solution to the nonlinear cubic Duffing equation: An application to electrical signal analysis of distribution lines”, *Applied Mathematical Modelling*, Vol. 40, No. 21-22, 2016, pp. 9152-9164.
[13] A. Guy and A. Bowling, “Modification of Nose–Hoover Thermostat to Improve Temperature Response in Molecular Simulations”, *Journal of Computational Nonlinear Dynamics*, Vol. 12, No. 3, 2017, pp. 031019:1-6.
[14] E. Davis and A. Zaher, “Time-dependent generalization of the Nosé-Hoover thermostatting technique for molecular dynamics simulations”, *Journal of Physics*, Vol. 490, No. 1, 2014, pp. 012101:1-4.
[15] N. Wulf, A. D. George, and A. Gordon-Ross, “Optimizing FPGA Performance, Power, and Dependability with Linear Programming”, *ACM Transactions on Reconfigurable Technology and Systems*, Vol. 10, No. 3, 2017, pp. 1-23.
[16] H. Yang, J. Zhang, J. Sun, and Le Yu, “Review of advanced FPGA architectures and technologies”, *Journal of Electronics (China)*, Vol. 31, No. 5, 2014, pp. 371-393.
[17] X. Zou, L. Wang, Y. Tang, Y. Liu, S. Zhan, and F. Tao, “Parallel design of intelligent optimization algorithm based on FPGA”, *The International Journal of Advanced Manufacturing Technology*, Vol. 94, No. 9-12, 2018, pp. 3399-3412.