Verification of Various Numerical Methods Using Hardware Implementation

Prof. Sandip Mehta
Assistant Professor
Department of Electrical Engineering
Instrumentation & Control Engineering
Institute of Technology, Nirma University
Ahmedabad, India
Email: sandip.mehta@nirmauni.ac.in

Tej Trivedi
Research Scholar
Department of Electrical Engineering
Instrumentation & Control Engineering
Institute of Technology, Nirma University
Ahmedabad, India
Email: 13micc24@nirmauni.ac.in

***Abstract*—Numerical analysis is the study of algorithms that use numerical approximation for the problems of mathematical analysis and control system. These Numerical Algorithms are dedicated to applying its unique expertise in numerical software engineering to delivering high quality computational software and high performance computing services. In this paper different numerical techniques for ordinary differential equations are simulated for different system models. The system will be modeled using differential equations. The differential equations will be solved using available numerical methods. The comparative analysis will be carried out for different numerical methods using advanced software tools. The hardware implementation of the numerical methods will be done using real time embedded system/hardware.**

***Index Terms*—Numerical Methods, LabVIEW, C-RIO.**

I. Introduction

In many engineering problems there is a constant need for solving systems with differential equations. Numerous scientific and technical problems can be described by means of single equation with one variable or systems of **n** equations with **n** variables. These equations that are dependent on number of independent variables and corresponding number of derivatives can be divided into ordinary differential equations and partial differential equations [1].

Ordinary differential equations frequently occur in mathematical models that arise in many branches of science and engineering. Unfortunately it is seldom that these equations have solutions which can be expressed in closed form, so it is common to seek approximate solutions by means of numerical methods. However most real world problems do not have closed form analytical solutions. Instead one has to use computer algorithms to calculate numerical solutions. With the combination of math knowledge and programming ability numerical algorithms can be designed and selection of such algorithms can be made applicable to wide range of real world problems [1], [2].

These ordinary differential equations defined only for one value of the independent variable are called initial value problems satisfying the given differential equation with an initial condition y(t_0) = y_0 . These concept was proposed by Cauchy in Cauchy's Theorem [1], [2]. The different methods for solving differential equations are programmed in LabVIEW [3]–[6] and real-time implementation of it is done on C-RIO [7]–[9]. Later, for verification of these methods, the system models are implemented using real hardware.

978-1-4799-8433-6/15/$31.00

II. Different Methods for Solving Ordinary Differential Equations

This section is devoted to solving ordinary differential equations of the form

$$\frac{dy}{dt} = f(t, y) \tag{1}$$

Here various numerical methods are used to solve such equations; the method in general form is as follows:

New value = old value + slope step size

or, in mathematical terms,

$$y_{i+1} = y_i + \phi h$$

This formula can be applied step by step to compute out into the future and, hence, trace out the trajectory of the solution [10].

Initial Value Problems can be solved using one-step numerical methods such as Euler's Method, Heun's Method, RK Method etc. and multi-step numerical methods such as Adams-Bashforth-Moulton Method and Milne-Simpson Method. All these self-starting & non self-starting methods are discussed in these paper. Runge-Kutta (RK) method is single-step self-starting method which is used to find the initial values for non self-starting; Predictor-Corrector Methods (ABM, Milne's etc.); is described first with its algorithm [11].

Runge-Kutta (RK) Method:

The Runge-Kutta (RK) method of order N = 4 is most popular. The fourth-order RK method gives better accuracy, stability and is also easy to program. This method is based on computing y_{k+1} and starts with the initial point (t_0,y_0) and generates the sequence of approximation using following formula:

$$y_{k+1} = y_k + \frac{h(k_1 + 2k_2 + 2k_3 + k_4)}{6} \tag{2}$$

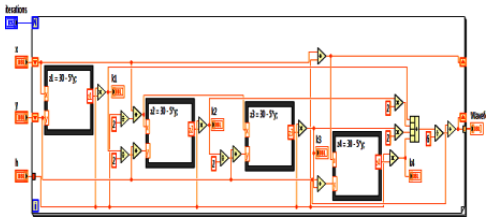

Fig. 1. FourthOrder RK Method LabVIEW code

where,

$k_1 = f(t_k, y_k)$
$k_2 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f_1)$
$k_3 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f_2)$
$k_4 = f(t_k + h, y_k + h f_3)$

Algorithm for RK Method: [11]

Algorithm steps for Fourth Order Runge-Kutta method are shown below:

1) Define f(t,y) [= RHS function of the differential equation y' = f(t,y)]
2) Read t,y,h[the initial values of t, y and the step size h of t]
3) For i = 1(1) n, do till (12)
4) $k_1 \leftarrow$ h.f(t,y)
5) $k_2 \leftarrow$ h.f(t + h/2,y + k_1/2)
6) $k_3 \leftarrow$ h.f(t + h/2,y + k_2/2)
7) t $\leftarrow$ t + h
8) $k_4 \leftarrow$ h.f(t,y + k_3)
9) y $\leftarrow$ y + h(f_1 + 2f_2 + 2f_3 + f_4)/6
10) $y_m \leftarrow$ "the mathematical solution of the equation to be supplied"
11) Write t, y, y_m
12) Next i
13) End

Note: Initial values are assumed as t, y. Compute the value of y at the points t_0 + h, t_0 + 2h, ...

A. Single Step (Self Starting) Methods

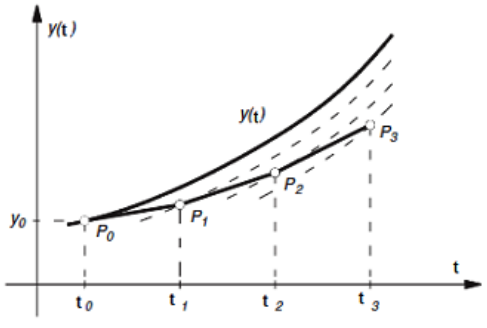

Fig. 2. Euler's Approximation

1) Euler's Method: Euler's Method is one of the simplest representative of the one-step methods, discussed on the basis

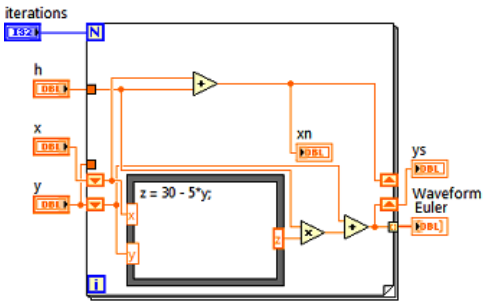

Fig. 3. Euler Method LabVIEW code

of Fig. 2. Process of finding the consecutive values y_n = y(t_n) = y(x_0 + n*h) for n = 1, 2, 3, . . . , begins from the starting point P_0 = (t_0, y_0). The first computed value of the desired function is:

$$y_1 = y(t_0 + h) = y_0 + hf[t_0, y(t_0)] \tag{3}$$

Thus point P_1 = (t_1, y_1) is obtained, which can be treated as the starting point in the process of finding next points. Repeating this procedure several times, the set of discrete values y_n of the function approximating desired solution y(t) is evaluated. [1], [10], [12]

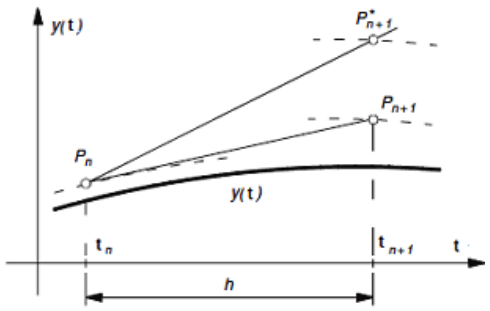

Fig. 4. Heun's Approximation

2) Heun's Method: In case of the Heun's method, an auxiliary coordinate y^*_{n+1} = y_n + hf (t_n, y_n) is calculated first and used next to determine the quantity

$$f(t_{n+1}, y^*_{n+1}) \tag{4}$$

expressing the slope coefficient of the tangent to the curve

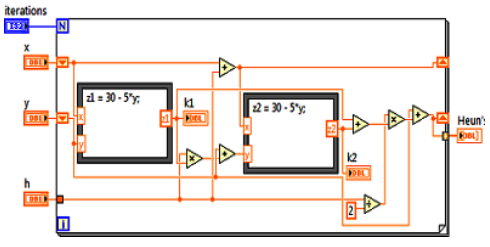

Fig. 5. Heun's Method LabVIEW code

described by equation (5) and passing through the point

$P^*_{n+1} = (t_{n+1}, y^*_{n+1})$ being first approximation of the desired solution. The point, which gives a much better approximation, $P_{n+1} = (t_{n+1}, y_{n+1})$, has the coordinate y_{n+1} calculated from the formula: [1], [10], [12]

$$y_{n+1} = y_n + \frac{1}{2}h[f(t_n, y_n) + f(t_{n+1}, y^*_{n+1})] \tag{5}$$

B. Predictor-Corrector Methods [12]

In case of all one-step methods discussed in the previous section, the value $y_{n+1} = y(t_{n+1})$ of the determined function is calculated on the basis of only one value $y_n = y(t_n)$, computed using the previous iteration information. In the multi-step methods not only the value $y_n = y(t_n)$ but also several other values $y_{nk+1} = y(t_{nk+1})$, $y_{nk+2} = y(t_{nk+2})$, $y_{nk+3} = y(t_{nk+3})$, . . . , $y_n = y(t_n)$ are required for calculation, where the number of steps k = 1, 2, 3, . . . also determines the order of method. [1], [12]

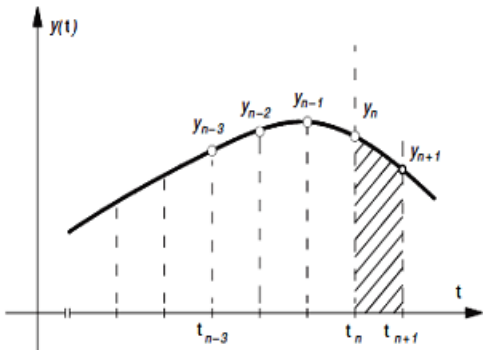

Fig. 6. Discrete Values of Predictor-Corrector Methods

1) ABM's Method: The Adams-Bashforth-Moulton (ABM) Method is a multi-step Predictor-Corrector Method derived from the fundamental theorem of calculus:

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))dt \tag{6}$$

The predictor uses the Lagrange Polynomial approximation for f(t, y(t)) based on the points (t_{k-3}, f_{k-3}), (t_{k-2}, f_{k-2}), (t_{k-1}, f_{k-1}) and (t_k, f_k). It is integrated over the interval $[t_k, t_{k+1}]$ in equation (6). This process produces Adam-Bashforth Predictor:

$$p_{k+1} = y_k + \frac{h}{24}(-9f_{k-3} + 37f_{k-2} - 59f_{k-1} + 55f_k) \tag{7}$$

The corrector is developed similarly. The value p_{k+1} just computed can now be used. A second Lagrange polynomial for f(t,y(t)) is constructed, which is based on the points (t_{k-2}, f_{k-2}), (t_{k-1}, f_{k-1}), (t_k, f_k) and the new point $(t_{k+1}, f_{k+1}) = (t_{k+1}, f(t_{k+1}, p_{k+1}))$. This polynomial is then integrated over $[t_k, t_{k+1}]$, producing the Adams-Moulton corrector: [12]

$$y_{k+1} = y_k + \frac{h}{24}(f_{k-2} - 5f_{k-1} + 19f_k + 9f_{k+1}) \tag{8}$$

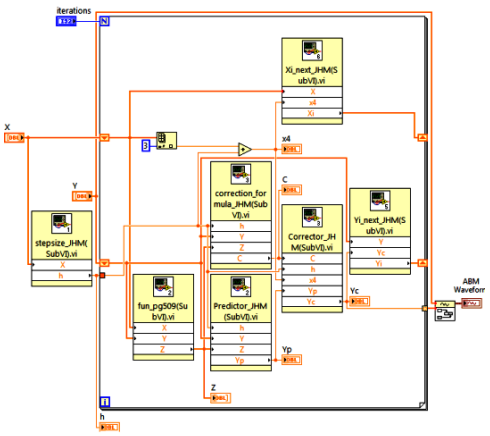

Fig. 7. ABM Method LabVIEW code

2) Milne's Method: Another popular Predictor-Corrector method is Milne-Simpson Method. Its predictor is based on integration of f(t, y(t)) over the interval $[t_{k-3}, t_{k+1}]$:

$$y(t_{k+1}) = y(t_{k-3}) + \int_{t_{k-3}}^{t_{k+1}} f(t, y(t))dt \tag{9}$$

The predictor uses the Lagrange Polynomial approximation for f(t, y(t)) based on the points (t_{k-3}, f_{k-3}), (t_{k-2}, f_{k-2}), (t_{k-1}, f_{k-1}) and (t_k, f_k). It is integrated over the interval $[t_{k-3}, t_{k+1}]$ in equation (9). This produces Milne Predictor:

$$p_{k+1} = y_{k-3} + \frac{4h}{3}(2f_{k-2} - f_{k-1} + 2f_k) \tag{10}$$

The corrector is developed similarly. The value p_{k+1} just computed can now be used. A second Lagrange polynomial for f(t,y(t)) is constructed, which is based on the points (t_{k-2}, f_{k-2}), (t_{k-1}, f_{k-1}), (t_k, f_k) and the new point $(t_{k+1}, f_{k+1}) = (t_{k+1}, f(t_{k+1}, p_{k+1}))$. The polynomial is then integrated over $[t_{k-1}, t_{k+1}]$, and the result is familiar Simpson's rule: [1], [12]

$$y_{k+1} = y_{k-1} + \frac{4h}{3}(f_{k-1} + 4f_k + f_{k+1}) \tag{11}$$

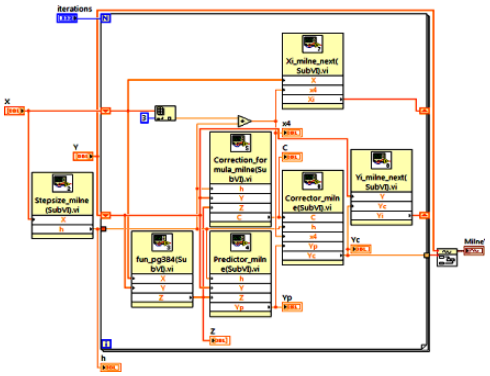

Fig. 8. Milne's Method LabVIEW code

3) Euler's Predictor Method: Euler's Predictor-Corrector (EPC) Method uses the simple Euler's Formula as the predictor and the improved Euler's Formula as corrector. The simple Euler's formula is given by:

$$y(t+h) = y(t) + hf(t,y) \tag{12}$$

and the improved Euler's Formula is given by:

$$y(t+h) = y(t) + \frac{h}{2}[f(t,y) + f(t+h, y+hf(t,y))] \tag{13}$$

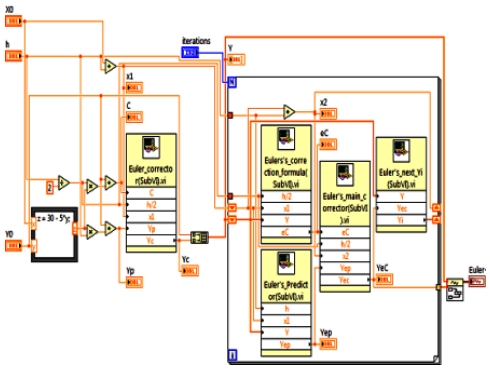

Fig. 9. EPC Method LabVIEW code

III. Simulation Results

The Simulated Results of following equation (14) using different numerical methods is discussed next.

$$y' = 30 - 5y \tag{14}$$

The computation of the above equation (14) is done in LabVIEW with a fix step-size and initial condition y(0) = 1. [12]

The following Table 1 shows the simulation time in LabVIEW and its real-time implementation in C-RIO. The equation (14) is taken as an example to solve, using different numerical methods for comparison and verification of the algorithms by obtaining various results. In the Table 1 for

TABLE I
Comparison of Simulated Response Time against C-RIO Implementation

Methods	**Approx. Simulation Time in LabVIEW**	**Real-Time Implementation (using C-RIO)**
Euler	4ms	50ms(approx.)
Heun	2ms	10ms
ABM	5s	20ms
Milne	Unstable	Unstable
Euler Predictor-Corrector	4ms	20ms

Milne's Method the real-time solution is unstable and that is due to 3 point method. Dahlquist Theorem proposed that if the modulus of characteristic roots of the equation are greater than one then the equation does not obey the concept of zero-stability. Hence, it may happen that this method does not give proper solution in some cases [2], [12].

A. Simulation of Real Systems

The above methods can be used to solve the practical systems in the real-world. Any practical system can be modeled using the concept of mathematical modeling [1], [13], [14] and simulation is done in LabVIEW. Two different real systems are modeled and its real hardware is also prepared for verification. The simulation results in LabVIEW and its Real-Time Implementation in C-RIO is discussed in following tables.

One of the two system is Full-wave Rectifier Integrated

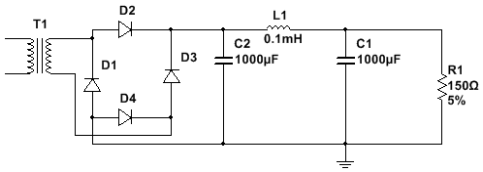

Fig. 10. Multisim Layout of Full-wave Rectifier Integrated With Three-Element Filter

With Three-Element Filter [1], [15], [16]. The time-domain analysis of this circuit (shown in Fig. 10) consists in determining the functions of current and voltage, which are considered as the state variables. The values of currents and voltages are related with state variables mentioned in the following set of differential equations:

$$\frac{dx(t)}{dt} = \frac{1}{C_1} * y(t) - \frac{1}{C_1 R} * x(t) \tag{15}$$

$$\frac{dy(t)}{dt} = \frac{1}{L} * z(t) - \frac{1}{L} * x(t) \tag{16}$$

$$\frac{dz(t)}{dt} = \frac{1}{C_2} * I_s[exp[\frac{|u_s(t)| - z(t)}{2 * V_T}] - 1] - \frac{1}{C_2} * y(t) \tag{17}$$

where, u_s(t) = V_m sin(2π*50*t) Volts.

These equations (15, 16 & 17) are nonlinear in nature

TABLE II
Comparison of Simulated Response Time against C-RIO & Hardware Implementation

Predictor-Corrector Methods	ABM	Milne	Euler
Approx. Simulation Time in LabVIEW	130ms	150ms	120ms
Real-Time Implementation (using C-RIO)	625ms	800ms	300ms
Real Hardware Response Time	250ms		

hence single point numerical methods fails to solve such systems. Predictor-Corrector methods are used for solving these differential equations and comparative analysis is done for the verification of numerical methods. This analysis is also compared with the output response of the real hardware and the conclusion can be made for the suitable numerical method which replicates the similar type of output response.

Comparative analysis for the same is shown in Table II above. The simulation time shown in Table II is approximate time taken after performing few simulations in LabVIEW.

Another system is Unsymmetrical Voltage Doubler [1], [16].

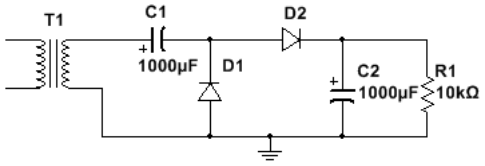

Fig. 11. Multisim Layout of Unsymmetrical Voltage Doubler

Similar to previous system this electronic circuit (shown in Fig. 11) is also analyzed using node voltages. These nodes are used as state variables. Here also, applying the concept of mathematical modeling for different state variables the first order differential equations are obtained.

$$\frac{dx(t)}{dt} = \frac{du_s(t)}{dt} + \frac{I_s}{C_1}[exp(\frac{-x(t)}{V_T}) - 1] - \frac{I_s}{C_1}[exp(\frac{x(t)y(t)}{V_T}) - 1] \quad (18)$$

$$\frac{dy(t)}{dt} = \frac{I_s}{C_2}[exp(\frac{x(t)y(t)}{V_T}) - 1] - \frac{1}{RC_2}y(t) \quad (19)$$

where, $u_s(t)$ = V_m sin(2π*50*t) Volts.

These equations (18 & 19) are solved using various predictor-corrector numerical methods and comparative analysis for this system is also carried out like it was done for previous system. Table III below shows comparison values.

The simulation time shown in Table III is approximate time taken after performing few simulations in LabVIEW.

IV. Conclusion

The various systems are simulated in LabVIEW to verify the results of each method. Basis on this simulation of different methods different solutions are obtained which are compared with the Real-Time Implementation on Hardware. Moreover, mathematical model of such hardware design is dumped into the C-RIO along with various numerical methods as solver. All these together gives the various results and from that, the verification of various numerical methods is done using Real-Time Implementation.

TABLE III
Comparison of Simulated Response Time against C-RIO & Hardware Implementation

Predictor-Corrector Methods	ABM	Milne	Euler
Approx. Simulation Time in LabVIEW	110ms	110ms	107ms
Real-Time Implementation (using C-RIO)	1.875s	2s	900ms
Real Hardware Response Time	800ms		

Acknowledgment

We would like to thanks Dr. Dipak M. Adhyaru, (Section Head IC), Institute of Technology, Nirma University for giving his valuable guidance for preparation of the topic and suggestions towards entire work for writing this paper.

References

[1] Stanislaw Rosloniec, "Fundamental Numerical Methods for Electrical Engineering", Springer.

[2] Endre Suli and David F. Mayers, "An Introduction to Numerical Analysis", University of Oxford, CAMBRIDGE UNIVERSITY PRESS

[3] Jovitha Jerome, "Virtual Instrumentation Using LabVIEW", PHI.

[4] Jeffrey Travis, Jim Kring, "LabVIEW For Everyone Graphical Programming Made Easy and Fun", Third Edition, Pearson Education.

[5] S. Sumathi, P. Surekha, "LabVIEW based Advanced Instrumentation Systems", Springer.

[6] LabVIEW, "Learning & Solving errors", [Online] Available: http://forums.ni.com/*topic name/error*.

[7] Maciej Rosol, Adam Pilat, Andrzej Turnau, "Real-time controller design based on NI Compact-RIO", Proceedings of the International Multiconference on Computer Science and Information Technology pp. 825830.

[8] National Instruments, "About LabVIEW & NI CRIO", [Online] Available: http://www.ni.com/getting-started/install-software/compactrio.

[9] Downloading "Add-ons/Toolkits" for LabVIEW [Online] Available: http://sine.ni.com/*add-on/toolkit*.

[10] Steven C. Chapra, Raymond P. Canale, "Numerical Methods for Engineers", Sixth Edition, McGraw-Hill (Higher Education).

[11] T Veerarajan, T Ramachandran, "Numerical Methods With Programs in C", Tata McGraw-Hill.

[12] John H. Mathews, Kurtis D. Fink, "NUMERICAL METHODS USING MATLAB", Fourth Edition, PHI.

[13] Katsuhiko Ogata, "Modern Control Engineering", Fourth Edition, PHI.

[14] I. J. Nagrath, M. Gopal, "Control System Engineering", Fifth Edition, New Age International Publishers.

[15] R. S. Sedha, "A TEXTBOOK OF APPLIED ELECTRONICS", Multicolour Edition, S. Chand.

[16] Robert Boylestad, Louis Nashelsky, "ELECTRONIC DEVICES AND CIRCUIT THEORY" Ninth Edition, PHI.