Chapter 6

Designing a DC Motor Simulator Based on Virtual Instrumentation

Nicolae Patrascoiu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79202

Abstract

This chapter proposes a state-space model for the DC motor built for separately excited voltage and considering two inputs: supply voltage and load torque. The three states of the resulted model are represented by angular speed, angular displacement, and current supply, and either of these states can be an output variable for the simulation model. Consequently, the system's model has two inputs and three outputs. Using this model, LabVIEW functions and programming structure of a simulator based on the virtual instrument is built, through which it is possible to observe the dynamic characteristics of the DC motor in different operating conditions. In this way, students can verify, by simulation, the operation of the DC motor as a dynamic system observing and measuring its reaction in different operating conditions.

Keywords: modeling, simulation, human-computer interface, front panel, diagram block virtual instrument

1. Introduction

One of the most used actuators in control systems is direct current (DC) motor. It is the means by which electrical energy is converted to mechanical energy. DC motors have a high ratio of starting torque to inertia and therefore they have a faster dynamic response. DC motors are constructed using rare earth permanent magnets, which have high residual flux density. As no field winding is used, the field copper losses are zero and hence, the overall efficiency of the motor is high. The speed-torque characteristic of this motor is flat over a wide range, as the armature reaction is negligible. Moreover, speed is directly proportional to the armature voltage for a given torque. The armature of a DC motor is specially designed to have low inertia [1].

The speed of DC motors can be controlled by applying variable armature voltage. These are called armature voltage controlled DC motors. Wound field DC motors can be controlled by either controlling the armature voltage or by controlling the field current [2]. In this chapter, we consider modeling and simulation of an armature controlled DC motor. There are many types of DC motor and their detailed construction is quite complex, but it is possible to derive the equations for a satisfactory dynamic model from basic electromagnetic relationships [3].

The general output variable of this actuator can be angular speed or angular displacement motion, but coupling the motor axle with wheels or drums and cables, translational motion can be obtained.

2. Building the simulation model

The physical model of an armature controlled DC motor [4] is given in **Figure 1**, where:

e_a, i_a—armature supply voltage and current; e_f, i_f—field voltage and current; R_a, L_a—armature winding resistance and inductance; e—back electromotive force (e.m.f.); $\omega(t)$—angular speed; $T_m(t)$—electromagnetic torque; $T_L(t)$—load torque.

When the armature is supplied with a DC voltage of e_a volts, the armature rotates and produces a back e.m.f e_b. The armature current i_a depends on the difference of e_a and e_b, so applying Kirchhoff's law it can obtain:

$$e_a(t) - e_b(t) = R_a \cdot i_a(t) + L_a \cdot \frac{di_a(t)}{dt} \tag{1}$$

The mathematical model of the mechanical system is:

$$T_m(t) = J \cdot \frac{d\omega(t)}{dt} + F \cdot \omega(t) + T_L(t) \tag{2}$$

where $T_m = k_m \cdot i_a$ is the torque T_m produced by the motor, J is the moment of inertia, F is the frictional coefficient and T_L is the load torque applied to the shaft of the motor considered to be time dependent. Based on these relationships considering that the back e.m.f. is proportional to

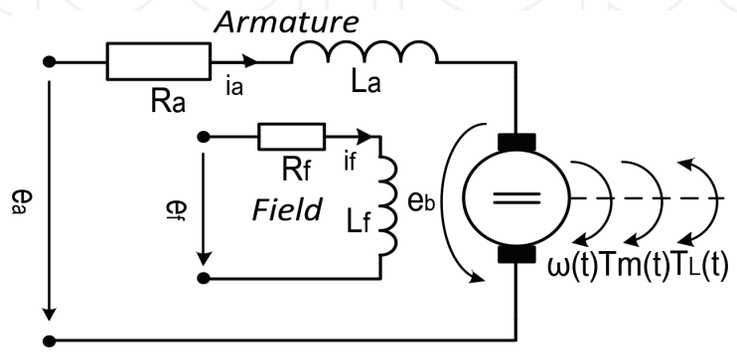

Figure 1. Armature controlled DC motor.

the speed of the motor, i.e., $e_b(t) = k_e \cdot \omega(t)$ and also considering that in SI (commonly used) $k_e \approx k_m = k$, is obtained:

$$\begin{aligned} \frac{di_a(t)}{dt} &= \frac{1}{L_a} \cdot e_a(t) - \frac{R_a}{L_a} \cdot i_a(t) - \frac{k}{L} \cdot \omega(t) \\ \frac{d\omega(t)}{dt} &= -\frac{F}{J} \cdot \omega(t) + \frac{k}{J} \cdot i_a(t) - \frac{1}{J} \cdot T_L(t) \end{aligned} \tag{3}$$

Considering the angular displacement $\alpha(t)$ instead of angular speed $\omega(t)$ as output variable, it is necessary to include the relationship between these:

$$\omega(t) = \frac{d\alpha(t)}{dt} \tag{4}$$

Generally, the mathematical model used is the transfer function [4, 5], but in this chapter it is proposed a Multiple Input Multiple Output (MIMO) mathematical model in a state-space form. Based on Eqs. (3) and (4), this model can build for use into a program simulation.

To build the state-space MIMO model, the input, state, and output vectors are defined [6]:

1. input vector $u(t)$, whose two components are represented by armature voltage $e_a(t)$ and load torque $T_L(t)$, that is,

$$u(t) = \begin{bmatrix} e_a(t) \\ T_L(t) \end{bmatrix} \tag{5}$$

2. state vector $x(t)$, whose three components are represented by armature current $i_a(t)$, angular displacement $\alpha(t)$ and angular speed $\omega(t)$, that is,

$$u(t) = \begin{bmatrix} i_a(t) \\ \alpha(t) \\ \omega(t) \end{bmatrix} \tag{6}$$

3. output vector $y(t)$, whose three components we consider to be the same with state vector components (Relation (8)) so it is possible to simulate these three physical variables.

Having these vectors, it is possible to write Eqs. (3) and (4) in the matrix form:

$$\begin{vmatrix} \dot{i}_S(t) \\ \dot{\alpha}(t) \\ \dot{\omega}(t) \end{vmatrix} = \begin{vmatrix} -\frac{R_a}{L_a} & 0 & -\frac{k}{L_a} \\ 0 & 0 & 1 \\ \frac{k}{J} & 0 & -\frac{F}{J} \end{vmatrix} \cdot \begin{vmatrix} i_S(t) \\ \alpha(t) \\ \omega(t) \end{vmatrix} + \begin{vmatrix} \frac{1}{L_a} & 0 \\ 0 & 0 \\ 0 & -\frac{1}{J} \end{vmatrix} \cdot \begin{vmatrix} e_a(t) \\ T_L(t) \end{vmatrix} \tag{7}$$

Act on systems general equations, this matrix equation can be written in the compact form:

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) \tag{8}$$

Bring on output vector definition, the input-output equation can be written in the matrix form:

$$y(t) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} i_S(t) \\ \alpha(t) \\ \omega(t) \end{vmatrix} + \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix} \cdot \begin{vmatrix} e_a(t) \\ T_L(t) \end{vmatrix} \tag{9}$$

or in the compact form:

$$y(t) = C \cdot x(t) + D \cdot u(t) \tag{10}$$

Having now the mathematical model, described by matrices A, B, C, D:

$$A = \begin{vmatrix} -\frac{R_a}{L_a} & 0 & -\frac{k}{L_a} \\ 0 & 0 & 1 \\ \frac{k}{J} & 0 & -\frac{F}{J} \end{vmatrix}; \quad B = \begin{vmatrix} \frac{1}{L_a} & 0 \\ 0 & 0 \\ 0 & -\frac{1}{J} \end{vmatrix}; \quad C = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}; \quad D = \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix} \tag{11}$$

It is possible to simulate the DC motor. Several studies, such as [7] [8], show that computer simulation enhances learning effectiveness especially in technical disciplines such as studying the behavior of dynamic systems.

3. Designing of virtual simulation tool

A program developed in LabVIEW is called a virtual instrument (VI), and it has two components: the block diagram that represents program itself and the front panel that is the user interface [9].

3.1. Front panel of the virtual instrument

The front panel of the virtual instrument and is built with controls and indicators, which are the interactive input and output terminals. The Controls simulate the instrument input devices and supply data to the block diagram of the VI and the Indicators simulate the instrument output devices and display data that are acquired or generated [10, 11].

The front panel has two components, one is used to set the parameters of the simulation and another is used to collect the results obtained from simulation. The user via the Tab Control button accomplishes switching between the two components.

The first component, called Simulation Process, contains elements (Control type) through which the user can:

- set the motor parameters (DC Motor Parameters): R_a [Ω], L_a [H], k [$N \cdot m/A$], J [$kg \cdot m^2$], F [$(N \cdot m)/(rad \cdot s)$];

http://dx.doi.org/10.5772/intechopen.79202

- set simulation time:
- choose, from the list, the shapes for the input variables and set their parameters (amplitude and duration);
- choose simulation previews for current and/or displacement and/or speed

An overview of the entire first component of the front panel is shown in **Figure 2**.

This part of the front panel also contains elements of graph indicator type through which a preview of the input and output variables can be displayed.

The second component, called Simulation Results, allows the user to the acquisition of information about the dynamic behavior of DC motor.

This contains Controls through which the user:

- can select the display state-space equations form and select the transfer function, both models are represented by coefficients calculated based on the DC motor parameters sets on the front panel, **Figure 3**; and
- can select and display the graphical results of the simulation, that is, the graphical responses for the imposed outputs shown in **Figure 4**.

For each selected output variable, a window opens, which displays parameters that are relevant to this dynamic system, such as the peak value (maximum value) and time of its occurrence, the steady-state value and its determining time, rise time, falling time, overshoot. For displacement, in case the load torque is greater than the active torque occurs motor reversing, a flashing message (OVER TORQUE) and a proper light indicator, also displaying the time of its occurrence, signal this situation like as shown in **Figure 4**.

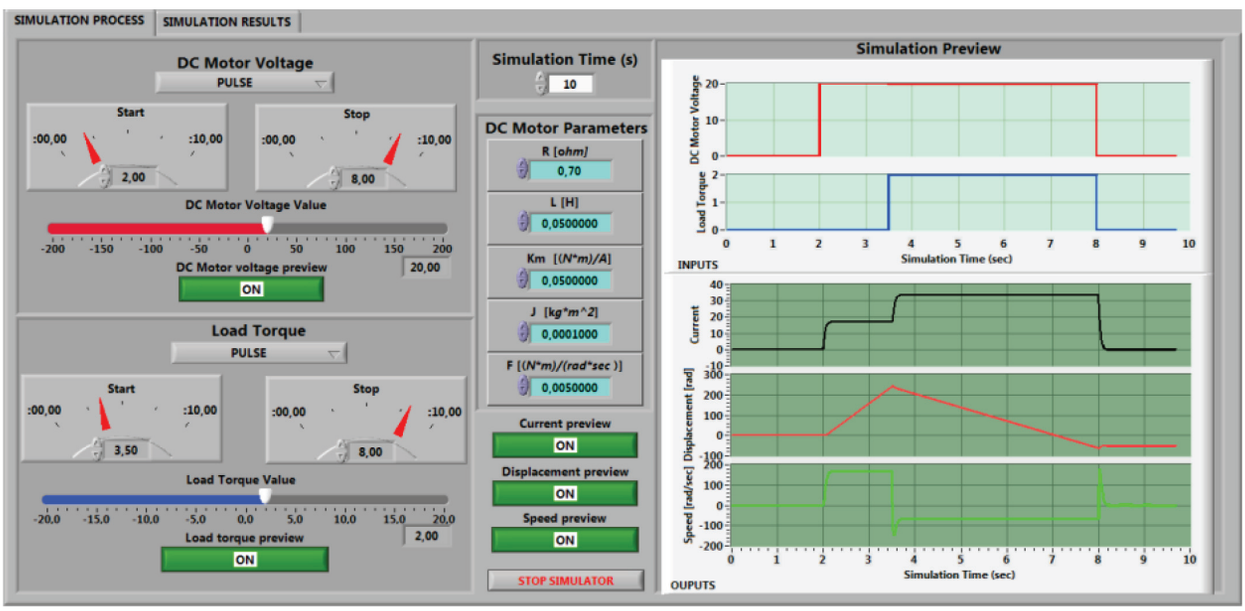

Figure 2. (a) Controls used to set the simulation parameters; (b) indicators used to preview of the simulation.

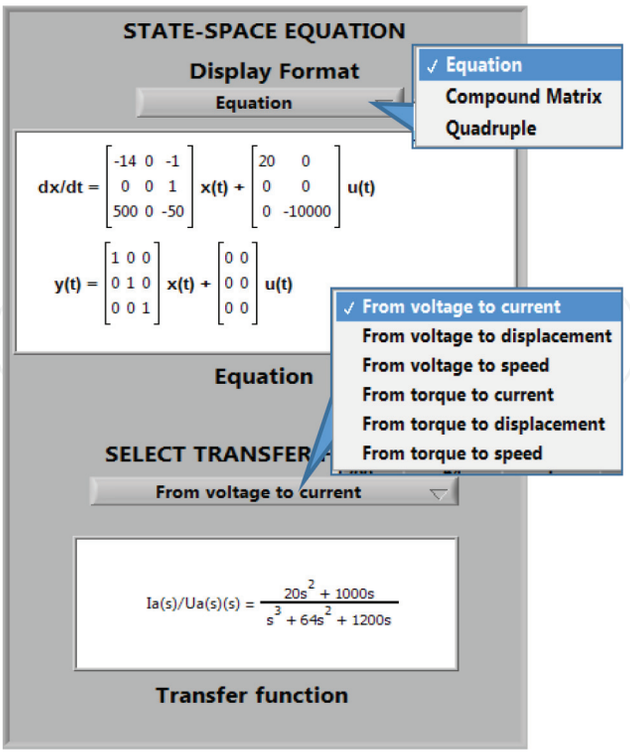

Figure 3. DC motor models.

Figure 4. DC motor simulation results.

http://dx.doi.org/10.5772/intechopen.79202

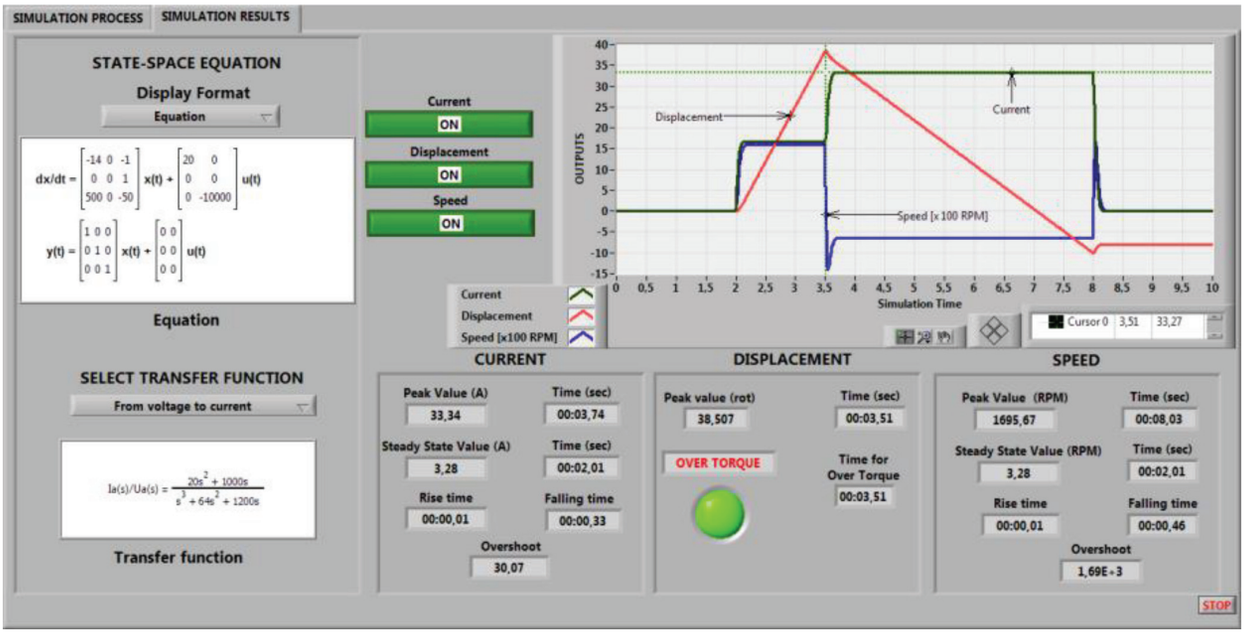

Figure 5. Second component of the front panel of the simulator.

The entire component SIMULATION RESULTS of the front panel is shown in **Figure 5**.

3.2. Block diagram of the virtual instrument

After the front panel is built, the code using graphical representations of functions to control the front panel objects is added. The block diagram represents the program under which the simulator runs and contains the graphical source code. Front panel objects appear as terminals on the block diagram. Block diagram objects include terminals, subVIs, functions, constants, structures, and wires, which transfer data among other block diagram objects.

The basic structure in constructing the virtual instrument is a While Loop that continues to run the program until the user presses the STOP SIMULATOR button on the front panel.

The program is running in two sequences according to the main algorithm shown in **Figure 6**. The user selects one of the shapes (step, pulse or ramp) and the values for the two variables input u_a and T_L and depending on constructive parameters of the DC motor, whose all values are set also by the user on the front panel, the state-space model is generated. By numerical simulation, based on this model, output variables are determined, and also, state-space and transfer functions models of the DC motor are displayed. The chosen output variables that represent the dynamic parameters of the system (maximum value, steady-state value, rise time, falling time, overshoot) are calculated.

The first sequence of the program and the MathScript Node are used to introduce the DC motor model into the program, and it is shown in **Figure 7**. The main element of the first sequence is the Control and Simulation Loop in conjunction with MathScript Node. Control and Simulation Loop is one of the components of the LabVIEW Control Design and Simulation Module. With this module, the plant and control models using transfer function, state-space

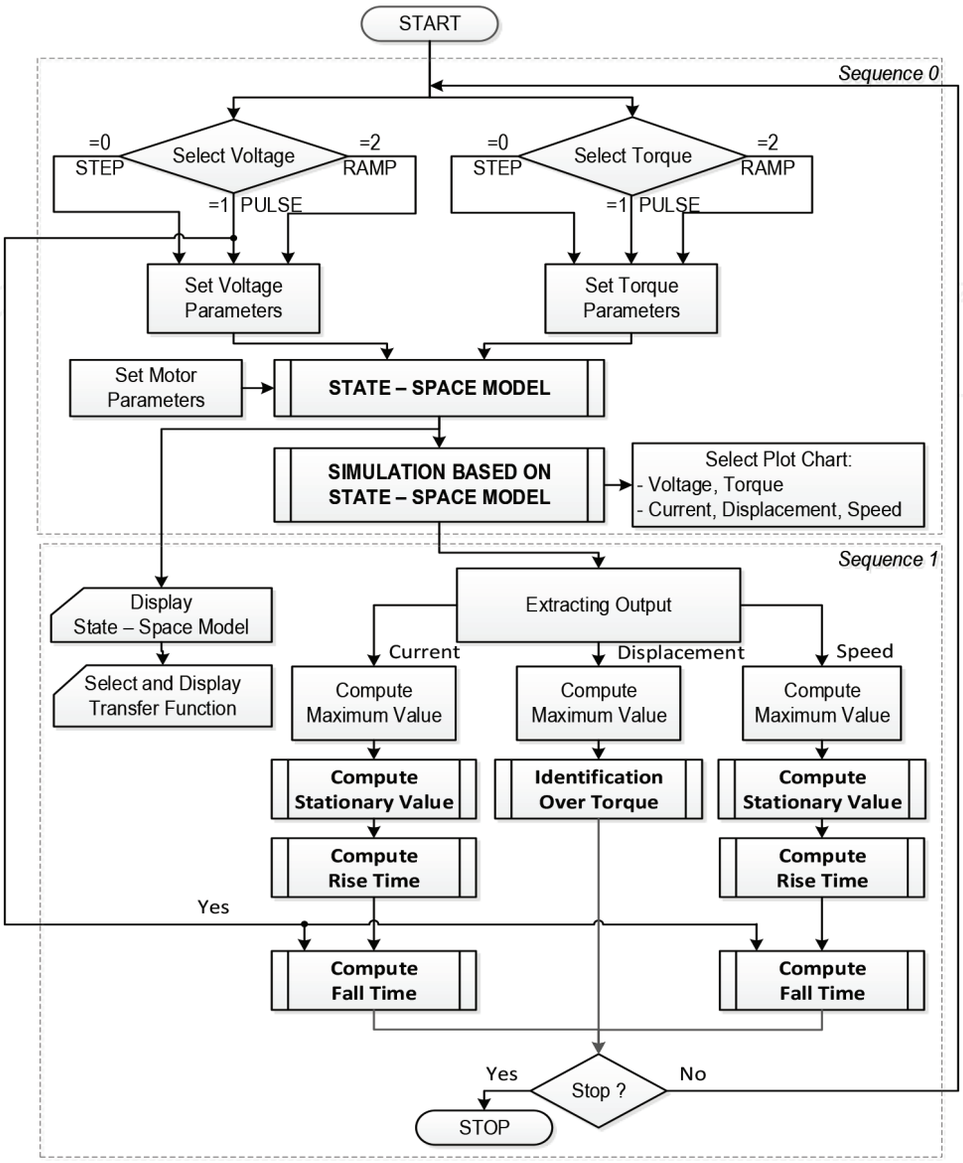

Figure 6. Main program algorithm.

model, or zero-pole-gain model can be constructed, and also system performance with tools such as step response, pole-zero maps, and Bode plots can be analyzed through simulation techniques. The simulation time is set from corresponding control on the front panel, and this is the only parameter of the loop.

The state-space model of the DC motor is placed into Control and Simulation Loop through a MathScript. MathScript is a high-level, text-based programming language. MathScript includes more than 800 built-in functions, and the syntax is similar to MATLAB and allows creating custom-made m-file, like in MATLAB. To process scripts in LabVIEW, LabVIEW MathScript Window or a MathScript Node can be used. If it is necessary to integrate MathScript functions (built-in or custom-made m-files) as part of a LabVIEW application and combine graphical and textual programming, as found in this application, MathScript Node is used [12, 13].

http://dx.doi.org/10.5772/intechopen.79202

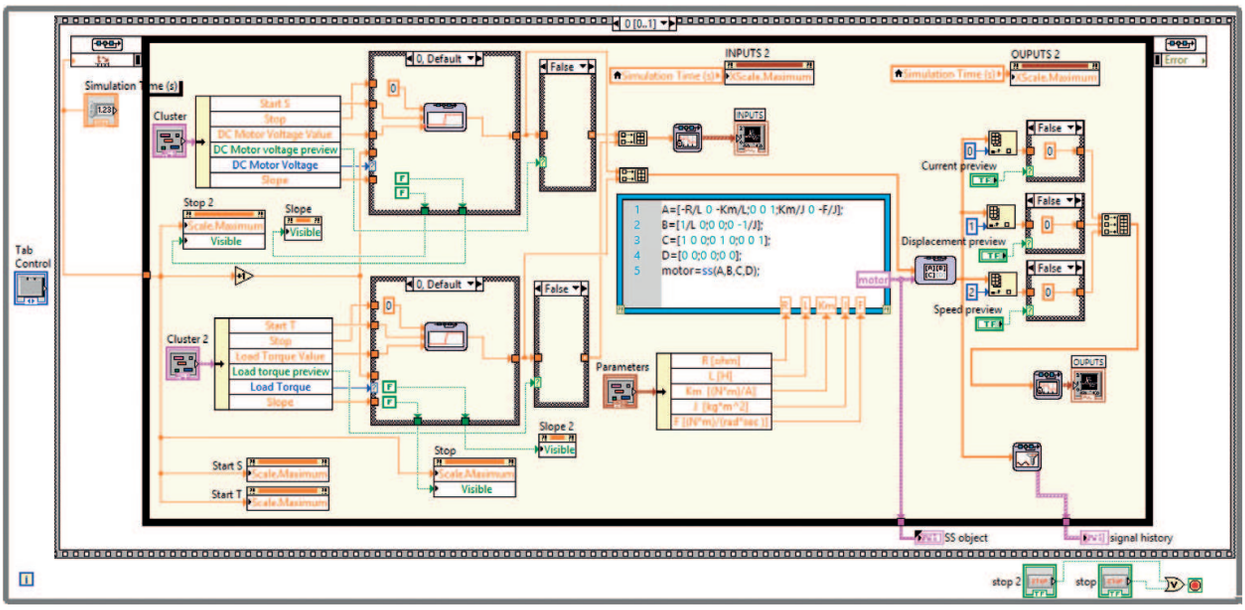

Figure 7. The first sequence of the simulator.

DC motor parameters are introduced into the model, and these can be changed interactively via the corresponding buttons on the front panel. The output from the MathScript Node is SS (state-space) object called motor and is a 2D matrix, which is a representation of the state-space system dynamic equations corresponding to the general forms (8) and (10). From this output, the matrices A, B, C, and D can be extracted, but also other information, such as properties, state names, transport delay. This object, together with the input vector, is used to generate the output vector $y^T = [ia, \alpha, \omega]$ through the LabVIEW simulation function State-Space. Each of the three components of the output vector can be identified using the Index Array function that returns the element or subarray of the n-dimensional array corresponding to the index value. After identifying any of these three components, these can be visualized on the front panel, through a graphic indicator. The obtained vector is also processed through the Collector function. This function collects a signal at each time step of the simulation and returns a history of the signal value and the time at which this function recorded each value in the history. Based on the signal history, the values of the dynamic regime, in the next sequence, can be identified.

For each of the two components of the input vector u_a respectively T_L standard signals are established, used to time domain analysis, namely: step, pulse, and ramp. The user can choose at any time and for each of the two components either of these signals through corresponding controls on the front panel. This choice is made by using a case structure and for the input selection, is used a control placed on the front panel button like is shown in **Figure 8**.

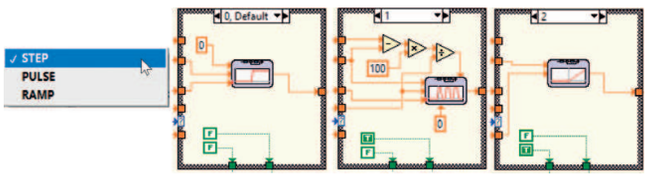

Figure 8. Selection of the input signals.

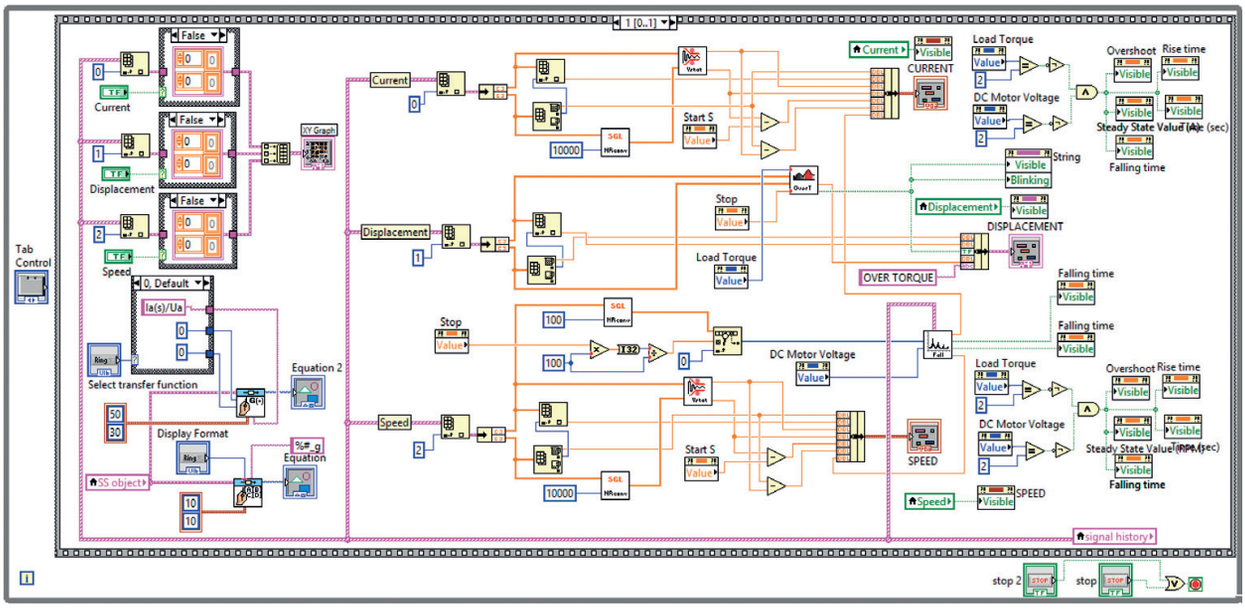

Figure 9. The second sequence of the program.

In the second sequence, shown in **Figure 9**, an identification of values is performed that characterize the operation in dynamic regime of the DC motor.

As mentioned earlier, the second sequence is used for extracting the information, obtained through simulation. Display of the equations of these models is done through *CD* Draw State-Space Function respectively CD Draw Transfer Function Equation, in Control Design and Simulation Toolkit.

By considering the DC motor like a MIMO system, having two inputs and three outputs are obtained six transfer functions that are displayed in a matrix form. To give to user the possibility to choose any of the six transfer functions, a control is provided on the front panel (Select transfer function) through which is generated the line and column index where the transfer function can be found (**Figure 10**).

As already mentioned, extraction of values that characterize the dynamic regime is made from the history signal. The history signal is a vector obtained as output from MathScript Node and represents the result of simulation. Just as in the first sequence, extraction of components from this vector is done through the Index Array function that returns the subarrays for the three outputs. Each of the three subarrays is composed of the values of output variables and values of the time variable on the simulation interval so that it is also possible to determine the corresponding time for each value that characterizes the dynamic regime.

3.3. Determination of dynamic regime values

The peak value that is the maximum absolute value of each of the output variables is computed using Array Max&Min function that returns the maximum and minimum values found in subarrays, along with the indexes for each value [7]. Since the peak value can be positive or

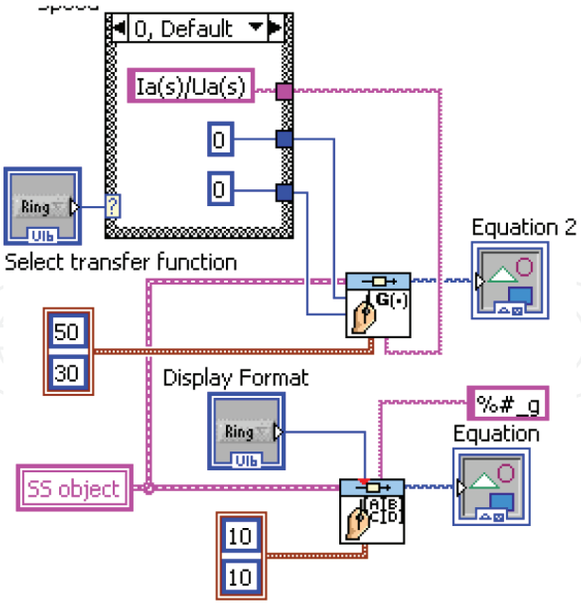

Figure 10. Functions used to display the equations of the model.

negative, depending on the direction of rotation, the maximum value is displayed for the maximum positive value and the minimum value for the maximum negative value. Selection is made through a CASE structure based on the comparison of absolute values corresponding to the maximum and the minimum values of displacement. Once determined the maximum value, based on its index, the moment of its appearance is searched.

The steady-state value *Sv* and the moment of its appearance *TSv* are computed using a SubVI named *Vstat* whose algorithm is shown in **Figure 11**.

This value is determined for each of the three components of the output vector. It considers the steady state to be reached if the difference between two consecutive absolute values for outputs variable is less than the constant $c = 10^{-6}$.

The algorithm shown above applies to current, to displacement and/or to speed, and each of them can be represented by the subarray *S(n)*. The moment of appearance of the steady-state value *TSv* is determined by reading the index *i* of the *Sv* in subarray *S(n)*, and with this index, the value with the same index *i* is searched in subarray *T(n)*, which is an array of the time values. Implementing of this algorithm in the virtual instrument structure is made by the SubVI *Vstat*, and this is shown in **Figure 12**.

For extracting the values *S(i)* of the subarray *S(n)* a while loop is used, whose condition terminal *i* is incremented until it fulfills the condition:

$$|S(i)|\text{-}|S(i\text{-}1)| \leq c \tag{12}$$

The rise time is computed through the difference between time *TSv*, when is identified the steady-state value, and the time when is generated the proper input variable. The time when is generated the input variable is taken from controls which determine the moment of start through a local variable or Property Node [9] StartS, as shown in **Figure 13**.

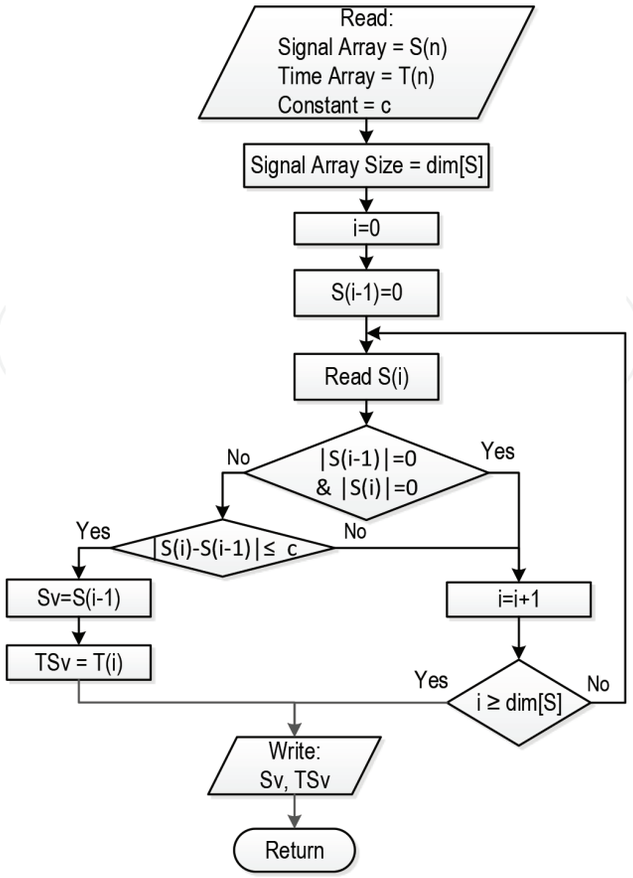

Figure 11. Algorithm used to compute steady-state value.

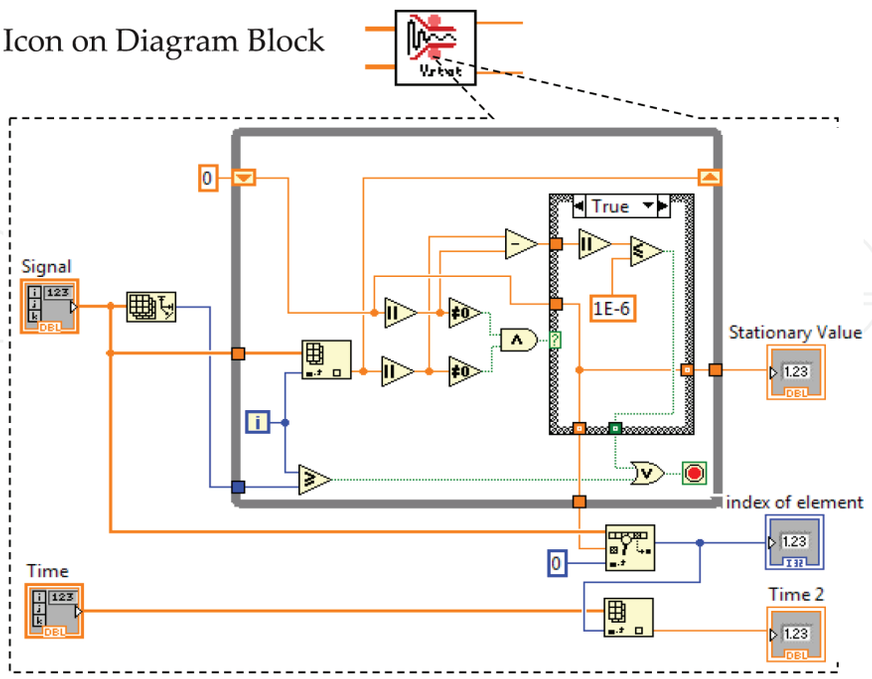

Figure 12. Implementation of the algorithm *Vstat*.

http://dx.doi.org/10.5772/intechopen.79202

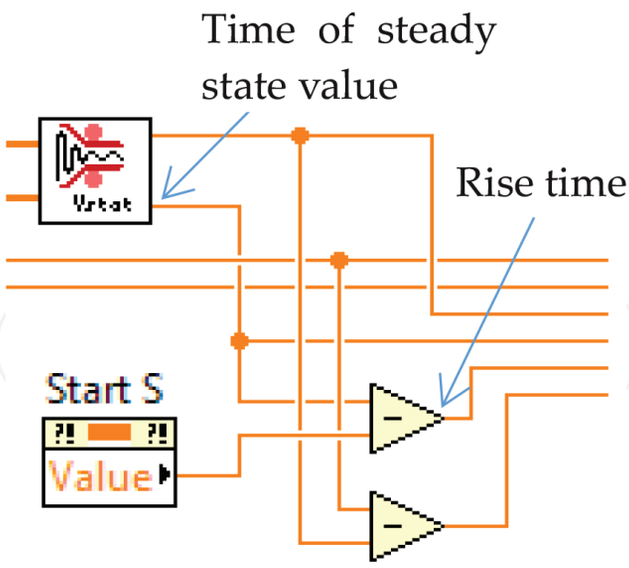

Figure 13. Computation of the rise time.

It is known that, if the load torque exceeds the active torque of the electrical machine, the "overturn" occurs. Therefore, it is necessary that the simulator be able to determine the overturning moment.

Identifying the phenomenon of "overturn" and the overturning moment is achieved using a SubVI named SENS, whose algorithm is shown in **Figure 14**.

Identifying the phenomenon of overturning is done by analyzing the sequence of values *Sd(n)* for angular displacement and the determination of the overturning moment is done by comparing the successive values *Sd(k-1)* and *Sd(k)* in the sequence considered. Thus, the phenomenon is considered to occur if:

$$\mathrm{Sd}(k)\text{-}\mathrm{Sd}(k\text{-}1) \le c_1 \tag{13}$$

where $c_1 = -0.01$ is a negative constant used to make the comparison.

If the torque shape is a pulse, the stop moment *TTs* and its index are identified in the sequence of time values and based on this index are removed all values with the index higher than this, from sequence *Sd(n)*. Obviously, the comparison is done only if two successive values from the sequence *Sd(n)* are different from zero.

By fulfilling the condition (13), the value *Sot* (=*Sd(k-1)*) is identified, corresponding to the overturn, in the sequence of values for angular displacement *Sd(n)*, starting with the appearance of the inversion of rotation. To signal the state of overturn, the value "true" is assigned to the boolean variable *Ovt*, otherwise the variable receives the value "false."

Indication of the overturning state is made on the front panel by a red color of a LED indicator (that otherwise is green) and by blinking message "OVERTURN," that becomes visible only when this phenomenon occurs.

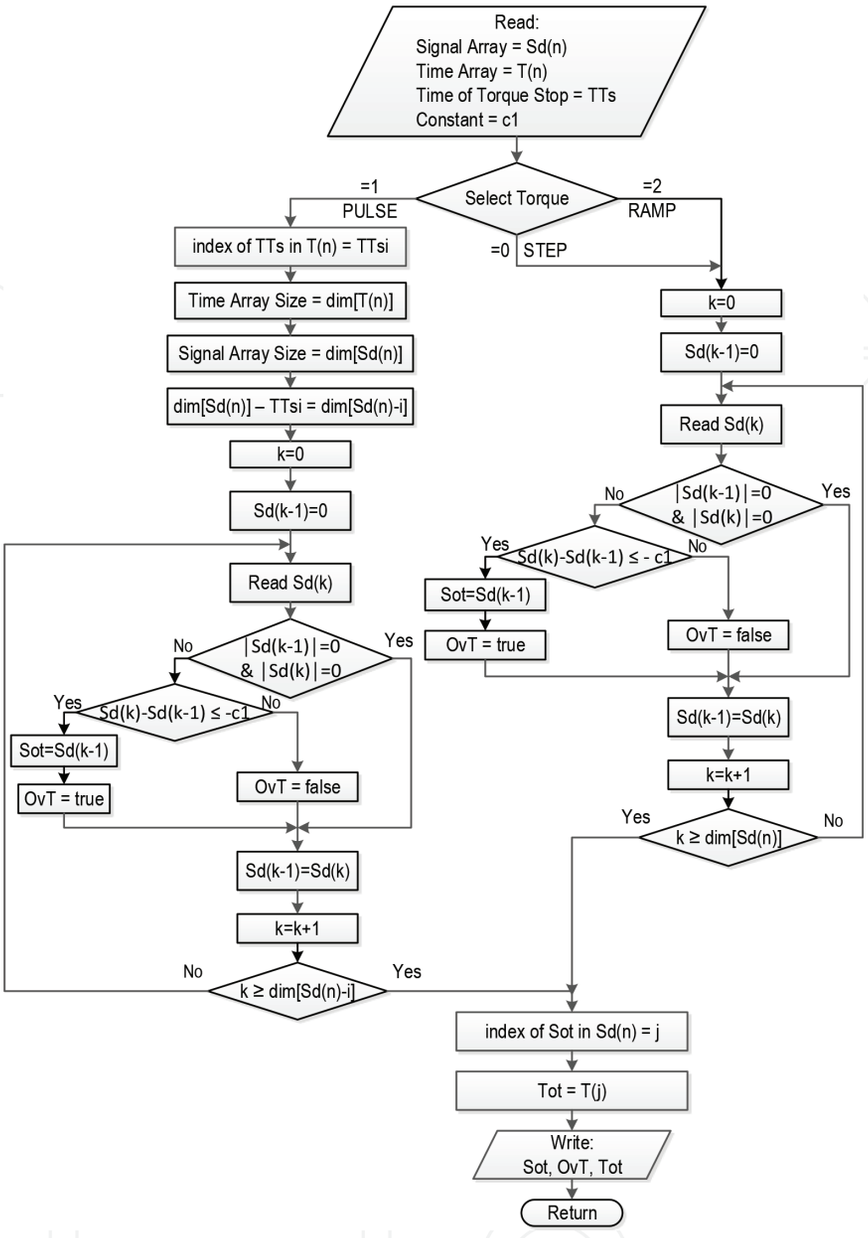

Figure 14. Algorithm used to identify the phenomenon of "overturn" and overturning moment.

In cases where the torque is step or ramp shape, considering the fact that these shapes did not have the stop time, it is not necessary to identify this moment in the times sequence values. In these cases, the overturn moment *Sot* is searched by identifying the condition given by (15) for successive values in sequence of values for angular displacement *Sd(n)*. Completion of the algorithm is carried on, as in the case of the torque having pulse shape.

After identifying the value *Sot*, corresponding to the overturning moment, its index is searched in the sequence of values for angular displacement *Sd(n)*. Based on the index *j* thus determined, the moment *Tot*, the moment when overturn occurs in the time array sequence *T(n)* is searched.

Implementation of the algorithm in the virtual instrument structure is made by the SubVI Over Torque, **Figure 15**, and this is realized through a While Loop that runs until the condition (13) is reached or until all values from the sequence of values for the angular displacement *Sd(n)* are tested. Depending on the shape of the input signal (Signal Type), one of the three cases is selected, noting that cases 0 and 2 (step and ramp) are the same.

In the execution of the search operation in the array sequences, 1D Array Search Function is used, that searches for the element *(Sd(k-1))* in a 1D array *(Sd(n))* starting at start index for that is set with 0 value. Because the search is linear, it is no need to sort the array before calling this function and LabVIEW stops searching as soon as the element is found. To make the index increments after each test for two consecutive values from the sequence array, the iteration (*i*) terminal of the While Loop is used that provides the current loop iteration count, which is zero for the first iteration.

Another value of the dynamic regime of DC motor that can be determined by the simulator is the fall time of the DC motor response. This value can be calculated, and it will be displayed as well, only for a pulse shape input. To display the value of this variable, proper indicators are used, which become visible on the front panel only if the input signal type is pulse shape.

To determine this value, the algorithm is used as shown in **Figure 16**.

Proper indicators became visible on the front panel only for a pulse shape input, when choosing the value true for the boolean variables *B1* and *B*, otherwise their values will be false.

In the sequence of the time values, the index *j* is identified, corresponding to the stop moment *TTs*, so from the sequence of the current and speed values, all values that have the index greater than *j* are removed. They are thus obtained for both current and speed variables, two sequences of values each with *S* size, and the values in these sequences are reordered starting at index 0.

Determination of fall time of the DC motor response is accomplished by identifying the moment when an absolute value for current or speed is zero (or less than a constant c_2, close to zero), according to the relations:

$$\begin{aligned} |\mathrm{Sc}(i)| &< c_2 \\ |\mathrm{Ss}(i)| &< c_2 \end{aligned} \tag{14}$$

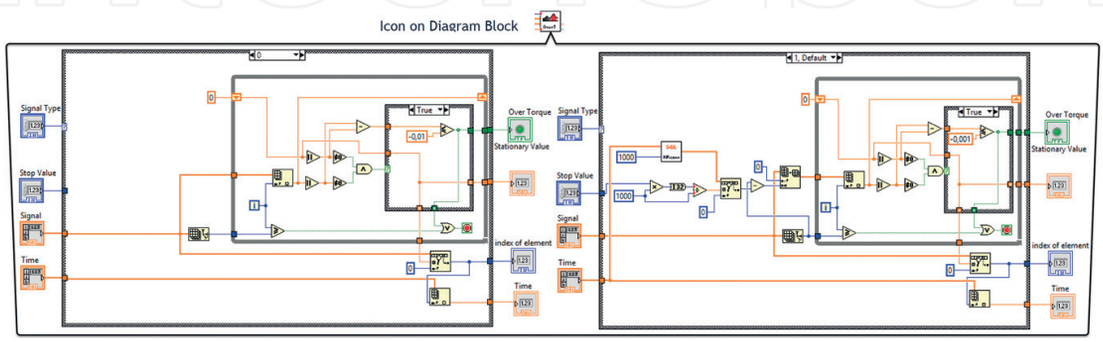

Figure 15. Implementation of the algorithm Over Torque.

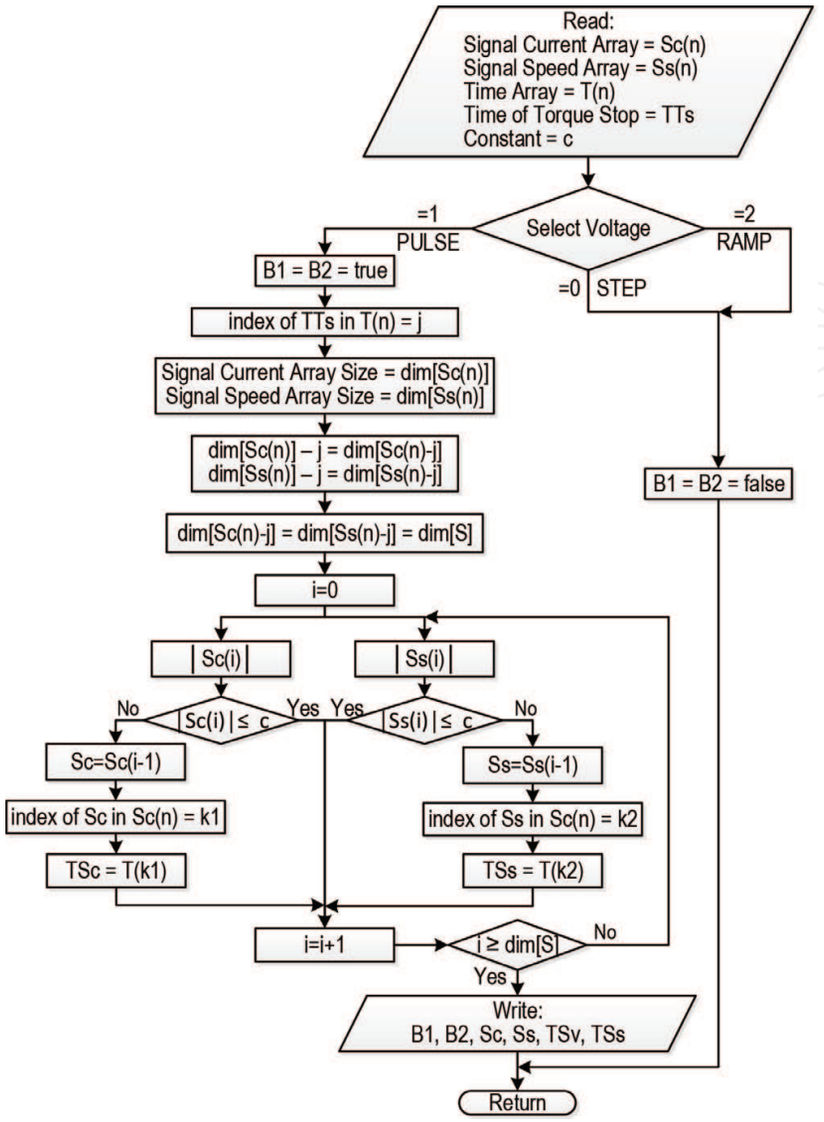

Figure 16. Algorithm used to identify the fall time of the DC motor response.

where $|Sc(i)|$ and $|Ss(i)|$ represent absolute values extracted from the sequences of values for current and speed, respectively.

The corresponding indexes k_1 and k_2 are identified for that the conditions (16) are achieved, and based on these indexes, the moments *TSc* and *TSs* are searched, corresponding to the achievement of these conditions.

Implementing the algorithm in the virtual instrument structure is made by the SubVI *Fall,* and this is shown in **Figure 17**.

The algorithm for determining the fall time of the DC motor response is made through a case structure. The three cases of this structure, corresponding to the three types of signals generated for DC motor command, are selected by controlling the DC Motor voltage from front panel through the value signal slide.

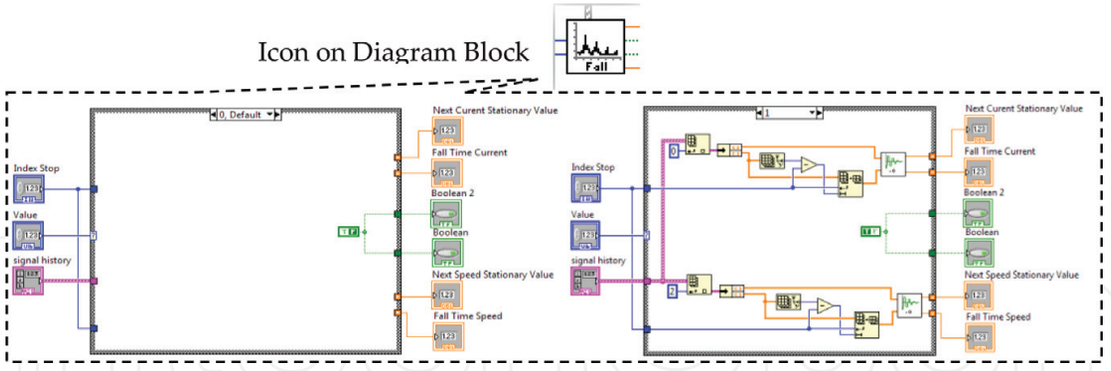

Figure 17. Implementation of the algorithm Fall.

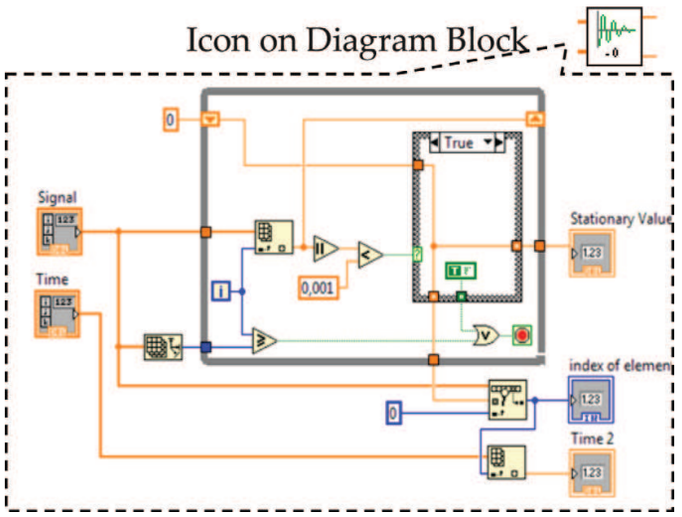

Figure 18. The SubVI Stationary Value Fall.

As seen in **Figure 17**, actual implementation of the algorithm takes place only in the sequence 1, which corresponds to a step shape command type for which is possible the compute this parameter of the dynamic regime. In the same sequence, boolean variables *B1* and *B2* are set to true. In the other two sequences, only the boolean variables B1 and B2 are set to false values.

In the structure of this SubVI, there is another SubVI (**Figure 18**) called Stationary Value Fall that determines just the moments in which the absolute values of current and displacement are 0 (or smaller than c_2, constant very close to zero). Its operation is similar to determine the stationary value in the SubVI Over Torque and consists in checking the differences between two successive values for displacement or current and to determine when this difference is zero (or less than constant c_2 whose value is set in this case at 10^{-3}).

4. Some results of simulation

To illustrate the operation of the simulator, several captures for the following elements of the front panel are presented:

- DC motor voltage and torque, in the graphical form, used as inputs variable shapes;
- DC motor parameters;
- simulation results in the graphical form for current, displacement and velocity used as outputs variables;
- numerical results and values for dynamical regime parameters, obtained from the simulation process.

Two types of simulations are used as follows:

- DC motors with different parameters are used and their simulation is done under the same conditions regarding the shape and parameters of input signals, that is, voltage DC motor and respectively, load torque;
- for the same type of DC, motor forms and parameters of the input signals can be modified or not, but in this case, one or more of the DC motor parameters are changed.

The simulation is performed for four DC motors with parameters presented in below table.

	Motor A	**Motor B**	**Motor C**	**Motor D**
R [Ω]	2.0	1	0.699	2.3
L [H]	0.0169	0.5	0.279	0.00845
Km [$N.m/A$]	0.283	0.01	0.215	0.66
J [$kg.m^2$]	0.0112	0.01	0.00279	0.052
F [$(N.m)/(rad.s)$]	0.058	0.1	0.00415	0.002

Considering the first type of simulation, to the four DC motors, a voltage as the control signal and a load torque having the shape and parameters, such as shown in **Figure 19** are applied. The voltage applied to the DC motors has a pulse shape, with an amplitude of 24 volts, from 2 to 8 s and the load torque increases linearly up to 1 N.m also from 2 s on an interval of 3.6 s.

In **Figure 20a–d**, the results of simulations corresponding to the four DC motors are presented.

As observed on the displacement graph, in the cases (a), (b), and (d), due to high load torque, overturning phenomenon occurs, and this phenomenon is indicated by turning on a LED, a blinking message, and identifying its moment of appearance. The peak values for current,

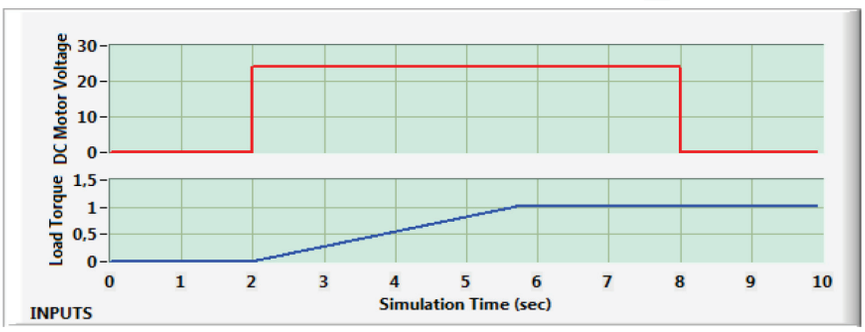

Figure 19. The inputs shapes used for simulation.

http://dx.doi.org/10.5772/intechopen.79202

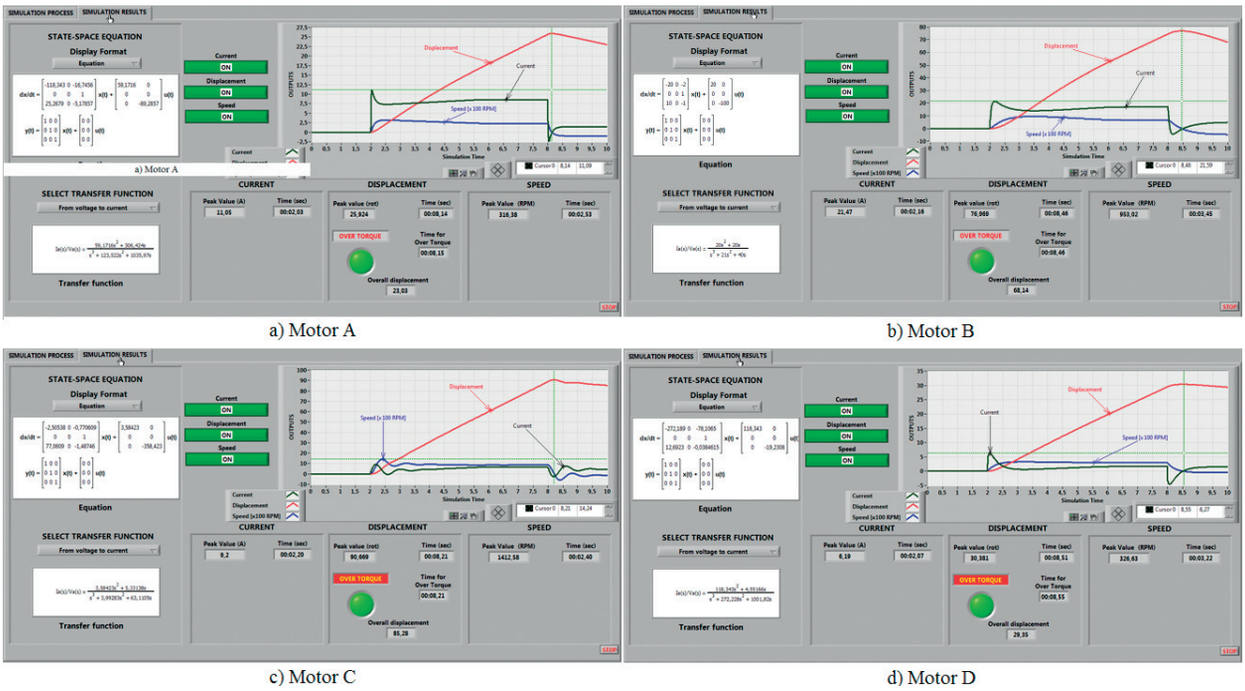

Figure 20. The results of the motors simulation. (a) Motor A. (b) Motor B. (c) Motor C. (d) Motor D.

displacement and speed with its moments of appearance and also, the overall displacement are displayed. Evidently, the user can select, for each DC motor, any of the three output variables or combinations thereof and the type of the transfer function.

For the second type of simulation, the DC motor D is chosen, and that it will be supplied with voltage under the shape of a step, impulse and ramp and also, that will be under the influence of a load torque that has a step shape with a front increased linearly.

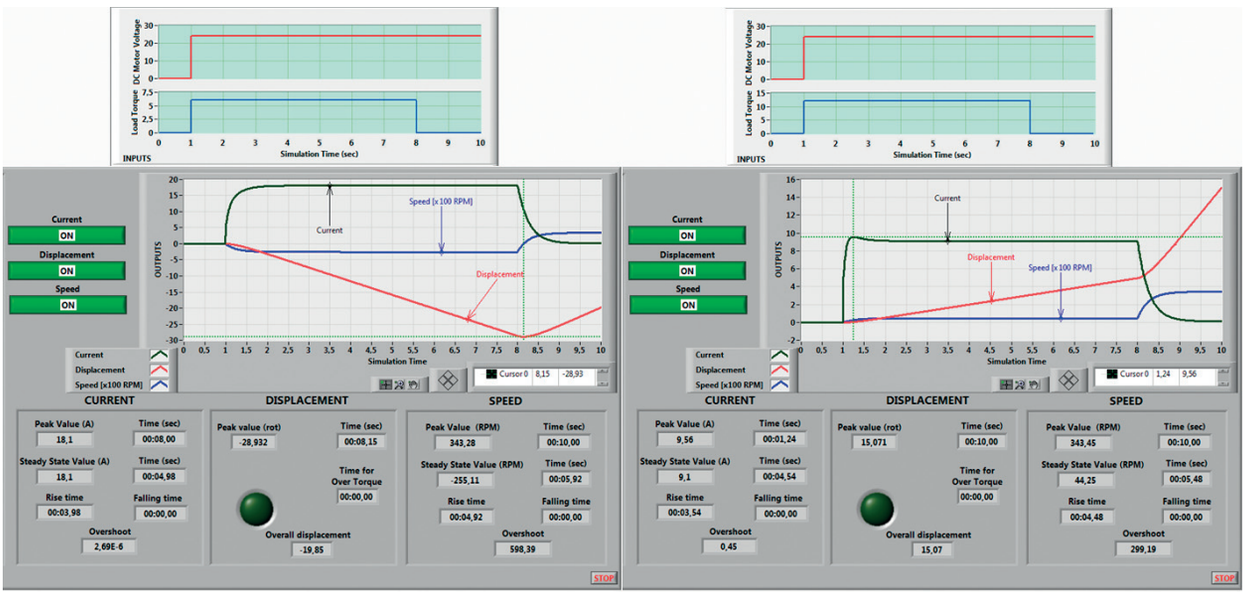

Figure 21. Increasing the load torque amplitude of 6–12 N.m.

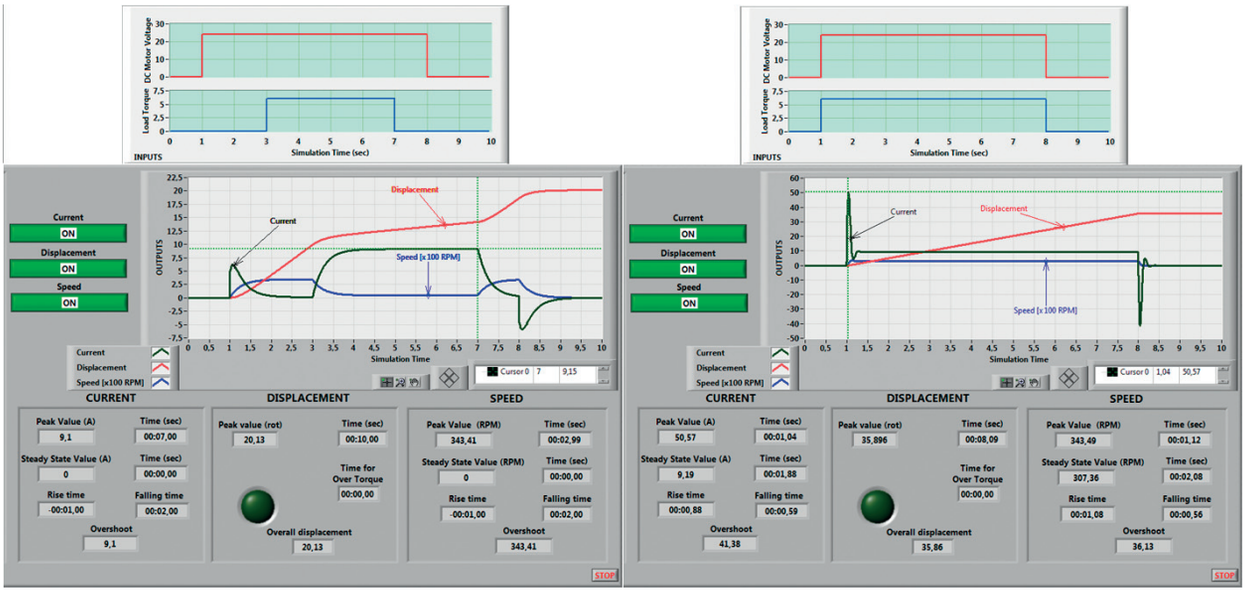

Figure 22. Decreasing the armature resistance of 2.3–0.3 Ω.

In **Figures 21** and **22**, the simulation results are compared for the DC motor powered by 24 V for a load torque increasing from 6 to 12 N.m, respectively for a load torque at 6 N.m and armature resistance decreasing from 2.3 to 0.3 Ω.

5. Conclusions

It is known that computer simulation is an experimental method based on a set of techniques by which operating with virtual things, the functioning of a real system or process is understood. To understand the functioning of systems dynamics, simulation environments such as Matlab or Simulink are used, and these simulations can be made based on corresponding mathematical models. These simulation environments are very powerful in terms of numerical calculation but offer few facilities for interactive modification of model parameters and/or the simulation conditions.

The simulator that we propose combines the computing power of the Matlab-Simulink environment with interfacing facilities provided by the LabVIEW programming environment. In this way, the users can understand the dynamics of a system such as a DC motor, following its chart of response to different stimuli or different operating conditions. On the other hand, by obtaining the mathematical model expressed by a transfer function in various operating conditions of the DC motor, students in modeling complex systems in which the DC motor represents a subsystem can use this one. Getting results in a graphical form allows users to compact large amounts of information and easily recognize these phenomena then by other modes of representation that would be more difficult to understand. They can also understand the effects of changes in the dynamic system parameters of its operation.

In addition, because LabVIEW is mainly a programming environment for data acquisition, it is possible to make a comparison between data acquired from a real system and data obtained by simulation from a mathematical model of this system by creating a hardware-in-the-loop structure.

The simulator can be extended to study other systems with minimal changes, using corresponding mathematical models.

Acknowledgements

An important part of this chapter represents the papers of the author which have been presented at the 5th International Multidisciplinary Scientific Symposium "Universitaria" (SIMPRO 2012) conference, at the 13th International Conference on Engineering of Modern Electric System (EMES 2105) and at the 14th International Conference on Engineering of Modern Electric System (EMES 2107).

Some of these papers have been written by a team of authors, and the author of this chapter has the acceptance of his co-authors for their use in the realization of this chapter.

Author details

Nicolae Patrascoiu

Address all correspondence to: nicolaepatrascoiu@upet.ro

System Control, Computer Engineering and Electrical Engineering Department, University of Petrosani, Romania

References

[1] Jagan NC. Control systems. 2th ed. Hyderabad: BS Publications; 2008

[2] Alexandru F. Machines and Electric Drives (in Romanian). Bucharest: Ed. Tehnica; 1986

[3] Golnaraghi F, Kuo BC. Automatic Control Systems. 9th ed. USA: John Wiley & Sons, Inc; 2010

[4] Dorf RC, Bishop RH. Modern Control Systems. 12th ed. USA: Prentice Hall; 2011

[5] Matko D, Karha R. Simulation and Modelling of Continuous System. New York: Prentice-Hall; 1992

[6] Patrascoiu N. Modeling and simulation of the DC motor using Matlab and LabVIEW. International Journal of Engineering Education. 2005;**21**(1):49-54

[7] Trundle KC, Bell RL. The use of a computer simulation to promote conceptual change: A quasi-experimental study. Computers & Education. 2010;**54**(4):1078-1088

[8] White R. Computational Mathematics. Models, Methods, and Analysis with MATLAB. USA: Chapman & Hall/CRC; 2004

[9] Beyon JY. LabVIEW Programming, Data Acquisition, and Analysis. New York: Prentice Hall; 2001

[10] Faraco G, Gabriele L. Using LabVIEW for applying mathematical models in representing phenomena. Computers & Education. 2007;**49**(3):856-872

[11] Bitter R, Mohiuddin T, Nawrocki M. LabVIEW. Advanced Programming Technique. 2nd ed. New York: CRC Press; 2007

[12] Halvorsen HP. LabVIEW MathScript. Telemark: Tutorial, Telemark University College, Department of Electrical Engineering, Information Technology and Cybernetics; 2011

[13] Marcu MD, Popescu FG, Pana L. Modeling and simulation of power active filter for reducing harmonic pollution using the instantaneous reactive power theory. Environmental Engineering and Management Journal. June 2014;**13**(6):1377-1382