Coding-Centered Dynamic Modeling of Induction Motors: An Educational Approach

N. FÜSUN OYMAN SERTELLER[1], DURSUN USTUNDAG[2], MEHMET CEVRİ[3]
[1] Electric-Electronic Engineering Department, Faculty of Engineering,
Marmara University, Istanbul,
TURKEY

[2]Mathematics Department, Faculty of Science,
Marmara University, Istanbul,
TURKEY

[3]Mathematics Department, Faculty of Science,
Istanbul University, Istanbul,
TURKEY

Abstract: - The analysis and synthesis of electric machines represent pivotal areas of inquiry within the broader field of electric machine study, particularly in the context of the induction motor (IM), which finds extensive application across a range of industrial settings. This study presents a comprehensive dynamic analysis of the 3-phase IM using Mathematica 13.0 software, with the objective of enhancing understanding by providing educationally oriented codes that are ready for further development. While similar analyses have been conducted previously, this study is distinctive in its use of Mathematica in this form. The accuracy of the results has been validated through a comparable MATLAB study. By leveraging the full potential of Mathematica's symbolic computing capabilities, this research addresses existing academic and research gaps and proposes a robust coding structure for IMs. The study employs Mathematica's specialized tools and animation capabilities to elucidate the characteristics of IMs over time, rendering it particularly useful for the design and control of electric machines. Electrical engineering students, who frequently excel in steady-state circuit problems, may encounter difficulties in dealing with time-varying electric machine quantities. Mathematica is a vital tool for electrical machine dynamics research and instruction because it makes it easier to handle intricate symbolic calculations, visualize data, and solve differential equations both symbolically and numerically.

Key-Words: - Induction Motor, Dynamic Response, Symbolic Tool, Coding, Education, Matlab, Mathematica.

Received: June 9, 2025. Revised: July 11, 2025. Accepted: August 13, 2025. Published: December 31, 2025.

1 Introduction

Induction motors (IMs) are preferred in industrial and research contexts due to several advantageous characteristics, including durability, ease of maintenance, low cost, straightforward mathematical modelling, and a simpler start-up procedure compared to synchronous and brushless direct current (DC) motors [1]. However, during the start-up phase, IMs draw high currents, which can result in voltage and torque oscillations and the generation of harmonics that have a detrimental impact on power systems. To mitigate these issues, several analysis models and equivalent circuits have been developed with the objective of enabling accurate predictions [2], [3], [4]. In their detailed analyses, researchers frequently employ reference frame transformations to model IMs using the dq axis, a crucial approach for students specializing in electrical machine design and control. Computer-based tools significantly enhance the understanding of these models, enabling students to better grasp operational features. Commonly used software packages for modelling and simulation include those based on FORTRAN and MATLAB/Simulink [5], [6], [7], [8]. In this study, motor considerations are implemented in the Scilab environment, and the induction motor is applied with a power P = 75 kW and input parameters for the 5AM280S6 electric motor model [9]. To demonstrate the operation of machines, a virtual example has been prepared to illustrate electromagnetic rotation in motors using computer programs [10]. The creation of a successful simulation tool requires a significant investment of time, energy, and expertise in computing languages and a comprehensive understanding of the operational characteristics of the

electrical machine and its performance. Although MATLAB/Simulink is an effective tool, it has limited symbolic capabilities in comparison to Mathematica [11]. Mathematica is particularly proficient in symbolic calculations, data visualization, and solving equations, making it an invaluable tool for modelling and analyzing electromechanical systems [12]. This combination of advanced computational capabilities and user-friendly visualization tools positions Mathematica as a superior choice for educational purposes in the study of IMs and their dynamics.

The objective of this paper is to propose the utilization of Mathematica's tools for the development of a comprehensive coding structure for IMs, with a particular focus on the analysis of motor characteristics over time. A summary of the development scripts is provided below:

- Specifying IM parameter values.
- Defining system equations for dynamic analysis across different cases.
- Setting initial conditions and time intervals.
- Solving differential equations under both load and no-load conditions
- Providing special examples for further analysis.

The proposed program can calculate and plot a range of characteristics associated with the IM, including fundamental parameters such as voltage, current, and speed, as well as *dq-axis* voltages, currents, inductances, power, torque, and speed. This integration of computational and visual elements facilitates enhanced analysis and comprehension.

2 IM Parameter Specifying, Identification, and Dynamic Equation Modeling

The code defines the voltage differential equations in the standard and the dq system, including the current, inductance, reactance, and ohmic resistance. The mechanical differential equations and their corresponding representations are also included. Solving these differential equations numerically provides the desired results, which can be plotted. This offers a comprehensive understanding of motor behavior under various operating conditions. This study demonstrates the use of reduced order models, explores IM parameter features, and encourages solving electric machines through coding. Understanding the interplay between electrical and mechanical dynamics is crucial for effectively designing and controlling IMs. Mathematica is an ideal tool for students and researchers to better understand these dynamics and their implications for motor performance. IM characteristics from all models are plotted on equivalent axes to highlight key differences. This study examines three progressively detailed models of IM characteristics as follows:

I. The "first order model" assumes an electrical steady state for the stator and rotor, with rotor speed being the sole dynamic variable. This is stated in Case 1.
II. The "third order model" includes rotor variables and assumes that stator variables are either negligible or in a steady state. It also excludes stator resistance from calculations. This is stated in Case 2.
III. The "fifth order model" incorporates stator and rotor variables, excluding deep bar effects. While there are potential limitations in realism, this model provides insightful simulations. This is stated in Case 3.

In this study, a large IM intended to drive a fan is characterized by the following parameter values:

- Stator Resistance (r_1): 0.460 Ω
- Rotor Resistance (r_2): 0.433 Ω
- Stator Leakage Reactance (x_1): 3.51 Ω
- Rotor Leakage Reactance (x_2): 5.05 Ω
- Magnetizing Reactance (x_m): 95.6 Ω

Here, the coded motor undergoes across the line-to-line start, with a 3-phase configuration. The motor load inertia(J) is *80 kg·m²*, and it is intended to drive a fan load. It is assumed that the power drawn by the fan is a cubic function of speed, with load torque proportional to speed squared. The fan load is 800 *kW*, operated by a *60 Hz*, $V_t = 6000$ *volts* rms line-to-line source, with a total of 8 poles (*P*) and a synchronous speed of 900 *rpm*. Deep bar effects and saturation are neglected in the coding. Optimization and control are also possible with the help of coding.

For the first case, the governing equation is written as follows:

$$s = 1 - \frac{P\omega_m}{\omega_0} \tag{1}$$

where s is the slip for IM, while P is the number of total poles on the motor and ω_m is the rotor speed in rad/s. Terminal impedance through the formulas is given as follows:

$$\left.\begin{aligned} Z_r &= j\,x_2 + \frac{r_2}{s} \\ Z_{ag} &= Z_r \,\square\, j\,x_m \\ Z_t &= r_1 + j\,x_1 + Z_{ag} \end{aligned}\right\}, \tag{2}$$

where $r_1, r_2, x_1, x_2,$ and x_m are defined as stator ohmic resistance, rotor ohmic resistance, stator reactance, rotor reactance, and mutual reactance respectively. Consequently, Z_r, Z_{ag} *and* Z_t represent the rotor impedance, the result of the parallel connection of the rotor impedance and mutual reactance, and the total impedance of the motor, respectively. The terminal voltage(V_t) , terminal current(I_a), rotor current(I_2), and induced torque(T_e) are given in equation (3):

$$\left.\begin{aligned} I_a &= \frac{V_t}{Z_t} \\ I_2 &= -\frac{j\,x_m}{j\,x_m + Z_r} I_a \\ T_e &= \frac{P\,P_{ag}}{\omega_0} = \frac{3}{2}\frac{P\,r_2}{\omega_0\,s}\left|I_2\right|^2 \end{aligned}\right\}, \tag{3}$$

where P is the total number and ω_0 denotes the rotor actual speed. The model load torque (T_l) and dynamic equation of the rotor speed are given in (4)[13]:

$$\left.\begin{aligned} T_l &= T_0\,(\frac{P\,\omega_m}{2\,\omega_0})^2 \\ \frac{d\,\omega_m}{dt} &= \frac{1}{J}\,(T_e - T_l) \end{aligned}\right\}. \tag{4}$$

The subsequent procedures, guidelines, and formulas are applied to Case 2 (third-order model) and Case 3 (fifth-order model). Equations (5) through (9), which establish a synchronously rotating reference frame, are the governing equations. In this reference frame, the dq axes are observed to rotate at synchronous speed, which is referred to as the speed term. The flux linkages, as seen in the last terms of equation (8), are associated with this speed. The dq index represents the current, flux, and voltage (i, λ *and* V) in this axis. [13], [14].

$$\left.\begin{aligned} \omega &= \omega_s \\ \frac{d\theta}{dt} &= \omega_s \end{aligned}\right\} \tag{5}$$

When considering the d-axis of the stator, the magnetizing flux, and the leakage flux, it is essential to recognize the interconnection of these elements with the winding due to its own current, i_{sd}. However, because of the current present in the other winding on the same axis, i_{rd} only the magnetizing flux links the stator winding. Based on this logic and scientific reasoning, the fluxes for the stator and rotor parts can be expressed by the following equations:

$$\left.\begin{aligned} \lambda_{sd} &= L_s\,i_{sd} + L_m\,i_{rd} \\ \lambda_{sq} &= L_s\,i_{sq} + L_m\,i_{rq} \\ \lambda_{rd} &= L_r\,i_{rd} + L_m\,i_{sd} \\ \lambda_{rq} &= L_r\,i_{rq} + L_m\,i_{sq} \end{aligned}\right\}, \tag{6}$$

and the L_s , L_r can be written as following:

$$\left.\begin{aligned} L_s &= L_{ls} + L_m \\ L_r &= L_{lr} + L_m \end{aligned}\right\}, \tag{7}$$

where L_s and L_r represent the self-inductances of the stator and rotor, respectively. L_{ls} and L_{lr} represent the leakage inductances of the stator and rotor, respectively. Meanwhile, L_m signifies the mutual inductance between the stator and rotor circuit.

The standard dq voltage equations are derived from the following vector-matrix form voltage equations, which are applied separately to the stator and rotor:

$$\frac{d\mathbf{x}}{dt} = \begin{bmatrix} -R_s/L_s & \omega_s & R_s L_m & 0 \\ -\omega_s & -R_s/L_s & 0 & R_s L_m \\ R_r L_m & 0 & -R_r/L_r & (\omega_s - \omega_m) \\ 0 & R_r L_m & -(\omega_s - \omega_m) & -R_r/L_r \end{bmatrix} \mathbf{x} + \mathbf{u} \tag{8}$$

where $\mathbf{x}=[\lambda_{ds}, \lambda_{qs}, \lambda_{dr}, \lambda_{qs}, \omega_m]^T, \mathbf{u}=[V_{ds}, V_{qs}]^T$, R_s and R_r: Stator and rotor resistances. ω_s: Synchronous speed (angular frequency). L_m , L_s , L_r : Coefficients derived from mutual and self-inductances.

The IM to be modelled [15] was considered a squirrel cage IM, the bar density in the rotor is uniform, therefore short-circuited as its voltages are $V_{rd} = V_{rq} = 0$ [13]. $V_{qs} = \sqrt{\frac{2}{3}}\,6000$ *and* $V_{ds} = 0$, since it is a common choice in the synchronously rotating reference frame.

The first row of (9) contains the electromagnetic torque equation, whereas the second row has the mechanical side equation:

$$\left.\begin{aligned} T_e &= \frac{3P}{4}\,L_m(i_{qs}i_{dr} - i_{ds}i_{qr}) \\ p\,\omega_m &= \frac{P}{2\,J}(T_e - T_l) \end{aligned}\right\} \tag{9}$$

The acceleration of IM is determined by the difference between the electromagnetic torque generated and the load torque

3 Code Implementation and Visualization of Results

Although MATLAB is effective for exploring electric machines, Mathematica excels, particularly in solving differantial equations, which are crucial for understanding the behavior of electric machines. Therefore, the Mathematica implementation has proven effective in this paper as an educational tool. A steady-state speed model with dynamic rotor speed is solved using the "First Order" equation, based on the classical 3-phase IM dynamic circuit equation in Figure 1(a). The "Third Order" model, utilizing *dq* analysis illustrates the transient behavior of rotor speed in Figure 1(b), emphasizing the transient effects before reaching steady state. The "Fifth Order" model further investigates *dq* analysis and its characteristics in Figure 1(c), focusing on the transient behavior prior to achieving steady state. Key differences between the models are highlighted by overlaying plots from all models on equivalent axes.

(a)

(b)

(c)

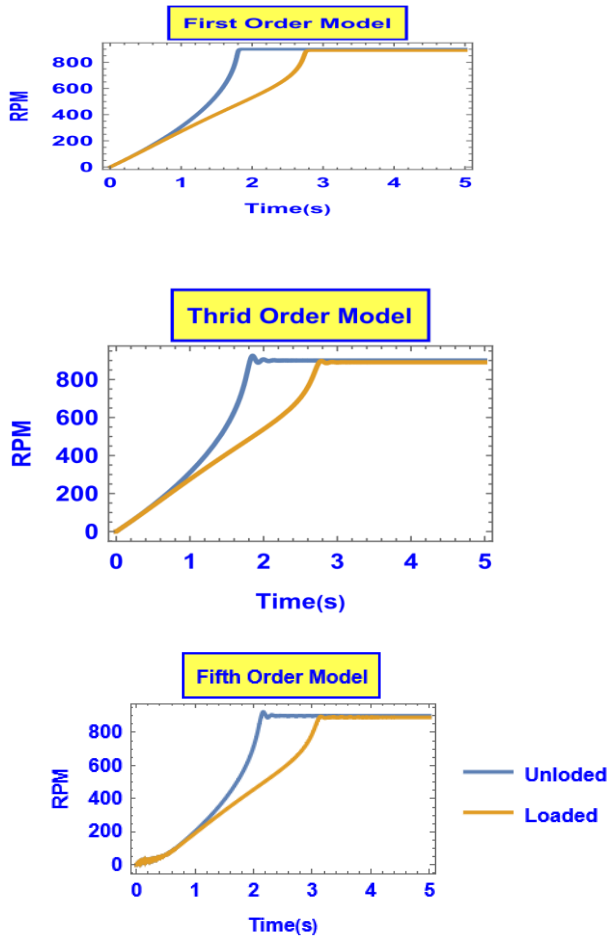


Fig. 1: Rotor speed Characteristics (rad/s) of 3 Phase IM for three model.
Source:Created by the authors.

The active power changes (Watt-P) for three models have been plotted by developing scrips for the three models in Figure 2(a), Figure 2(b), and Figure 2(c).

(a)

(b)

(c)

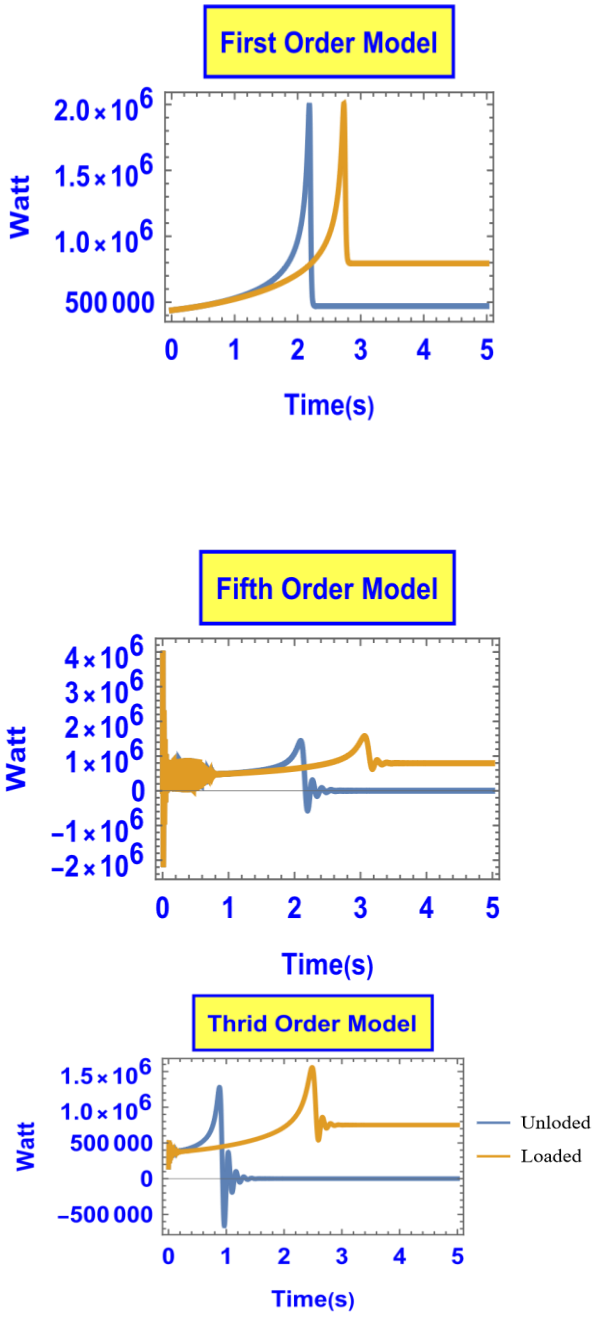
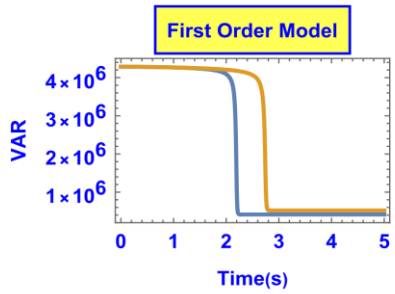
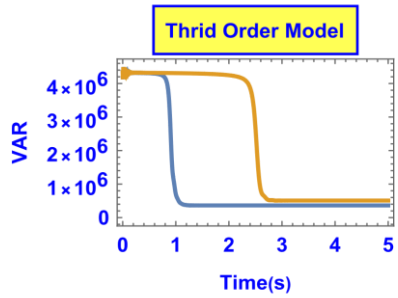


Fig. 2: Active Power (Watt-P) Characteristics of Motor for three models
Source:Created by the authors.

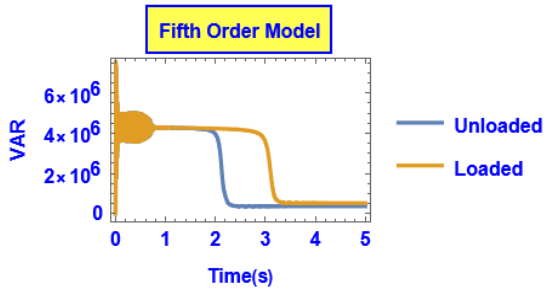
The reactive power characteristics (Q-VAR) are presented in Figure 3(a), Figure 3 (b) and Figure 3 (c) for three models.



(a)



(b)



(c)

Fig. 3: Reactive Power (VAR) Characteristics of Motorfor three models.
Source:Created by the authors.

Following the determination and derivation of the IM parameters, we offer the formulation and Mathematica scripts for putting the first, third, and fifth-order models—Cases 1, 2, and 3. The first step in the investigation is to define the values of the IM parameters and create generic formulas based on them. We then derive variables in the form of modules and the formulas that go with them. The DSolve command is used to solve the differential equations, and the Plot command is used to visualize the results. The torque and speed characteristics for various loads are shown in the appendix section (Appendix B and Appendix D). Furthermore, the Manipulate or Animate commands improve the mathematical process by enabling a deeper comprehension of electrical machinery. The analysis of IM operation at various levels, from basic to advanced, is covered in the parts that follow:

A. Case 1:

The "First Order" model assumes a steady state for the stator and rotor, with rotor speed being the only dynamic variable. Once the IM parameters are provided with the “NDSolveValue” command the problem is solved and results are given with plots. These steps are provided with the developed “module” given in Appendix A, which assists in constructing the circuit equation. This module includes both the definition of the parameters and the module itself.

B. Case 2:

Appendix B contains the script for the "Third Order Model." From (5) to (9) are utilized to investigate how non-linearities affect an IM's dynamic behavior; (9) explains the motor's mechanical behavior. Here, the rotor and stator currents must be determined. Since the rotor is "round," The quadrature axis functions just like the stator. This model assumes that the resistance is modest enough and that the stator runs in a steady condition with” modules”. The flux-current related other motor parameters are outlined in the script in Appendix B in here torque characteristics for loaded and unloaded is given.

C. Case 3:

The running script in Appendix C contains the 'fifth model,' while Appendix D addresses the manipulation of the function in Case 3. This model, which uses equations (5) through (9) and assumes rotor and stator parameters under unsteady conditions, is employed to investigate how nonlinearities affect the dynamic behavior of the IM. It is simpler still, as it directly utilizes the previously defined simulation model. The results are presented in Mathematica, which incorporates built-in numerical algorithms such as the Runge-Kutta and explicit methods to simulate motor performance in both no-load and loaded states.

4 Student Feedback

The questionnaire was administered in person to a cohort of graduate students. Totally, five questions were asked to the Electric-Electronic Engineering 12 master’s students in the 2024-2025 summer term after they had performed the script models for IM using Mathematica programming:

- Is it easy to use?
- Is it understandable and applicable?
- The input/output variable behavior in the dq analysis was clearly observable.

- Being able to observe simulation results graphically was very helpful
- Does it help you understand the subject better?

Feedback from students after completing the case models was satisfactory, positive, and encouraging.

5 Conclusions

The standard and dq-axis (two-axis) models of three-phase IMs, using Mathematica, provide an effective educational tool through symbolic computation and numerical simulation. This study aids in understanding motor operation principles and prepares students and researchers for advanced work in electric machines and control systems. The detailed models essential command, as "modules," show Mathematica's versatility in education and research. However, dq modeling may introduce harmonics and imbalances when transitioning between dq and three-phase systems, requiring further analysis to improve accuracy. Additionally, integrating Mathematica scripts enhances the learning experience, allowing students to explore and visualize motor characteristics effectively.

References

[1] A. Diaz, R. Saltares, C. Rodriguez, R.F. Nunez, E.I. Ortiz-Rivera and L. Gonzalez-Llorente, Induction Motor Equivalent Circuit for Dynamic Simulation, *IEEE International Electric Machines and Drives Conference*, Miami, FL, USA, 2009, pp. 858-863, doi:10.1109/IEMDC.2009.5075304.

[2] A. W. Leedy, Simulink/Matlab Dynamic Induction Motor Model for Use in Undergraduate Electric Machines and Power Electronics Courses, *Proceedings of IEEE Southeastcon*, 2013, pp.1–6, doi: 10.1109/SECON.2013.6567399

[3] A, Sangeetha, B.Parthiban, Review of Modeling and Dynamic Analysis of Three Phase Induction Motor Using MATLAB Simulink, *International Journal of Science, Engineering and Technology Research (IJSETR),* 3(3), 2014. pp. 374-378, doi: 10.9734/jerr/2024/v26i111331

[4] J. D. Lavers, and R. W. Y. Cheung, A Software Package for The Steady State and Dynamic Simulation of Induction Motor Drives, *IEEE Trans Power Syst. Vol. PWRS-1*, Miami, FL, USA, May. 1986, pp.167-173,

[5] N. Mohan, Advanced Electric Drives- Analysis, Control and Modeling Using MATLAB/Simulink, Wiley & Sons, Inc., Hoboken, New Jersey. 2014. doi: 10.1002/9781118910962.fmatter
L. Maraaba, Z. Al-Hamouz, M. Abido . An Efficient Stator Inter-Turn Fault Diagnosis Tool for Induction Motors. *Energies* 2018, 11, 653, doi:10.3390/en11030653,
doi: 10.46300/9109.2023.17.2

[6] V.Sarac, G. Stefanov, Numerical Analysis of Transient and Steady-State Operation of Three-Phase Induction Motor, *13th International Conference On Applied Electromagnetics*, Niš, Serbia, 2017.

[7] M.Rafał, Simulation of Three-Phase Induction Motor in Scilab, *Advanced Research in Scientific Areas*, 2012.

[8] S. Mann, D.E. Garcia, P.V. Do, D. Lam, P. Scourboutakos, Moveillance: Visualizing Electric Machines, *22nd Symp. on Virt. and Aug. Rea. (SVR)*, Porto de Galinhas, Brazil, 2020, pp. 420-424, doi: 10.1109/SVR51698.2020.00069

[9] N.F. Oyman Serteller, An Educational Study of Electromagnetic Phenomena on Ferromagnetic Structure Using A Software Environment, *Proc. Natl. Acad. Sci., India, Sect. A Phys. Sci*., 2024, pp. 94: 37–45, doi: 10.1007/s40010-023-00864-6

[10] T.C. O'Connell, P.T. Krein, and J.T. Mossoba, Symbolic Analysis with Mathematica in Power Electronics and Electromechanical Systems *IEEE Worksh.on Comp. in Power Elect,* 2004, Urbana, IL, USA, *pp.* 133-139,

[11] P.C. Patel, Modeling of Induction Motor, *International Journal Of Engineering Development and Research*, 2(2), 2014, pp.2077-2082,

[12] MIT Open Course Ware, Department Of Electrical Engineering And Computer Science Lecture Notes, *Electric Machines*. 2013.

[13] R.C.Panaitescu, N. Mohan, W. Robbins, P. Jose, T. Begalke, C. Henze, T. Undeland, E. Persson, An Instructional Laboratory for The Revival of Electric Machines and Drives Courses, *IEEE 33rd Annual IEEE Power Electronics Specialists Conf. Proceedings,* 2002, *Cairns, Queensland, Australia, pp.* 455-460,

[14] M. Konuhova, Induction Motor Dynamics Regimes: A Comprehensive Study of Mathematical Models and Validation, *Applied Sciences*.,2025,15(3):1527, doi: 10.3390/app15031527

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy

In the present study, N. Füsun Oyman Serteller was responsible for the conceptualization, methodology, design, and literature review; Dursun Ustundag contributed to computer programming, evaluation and analysis of results; and Mehmet Cevri contributed to proofreading, checking mathematical equations, and reviewing and editing the article in terms of content.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

It is not necessary for the article to be subject to ethics committee approval. In addition, the proposed article is devoid of any conflict of interest with any person or institution.

Appendix A

```
(* Clear all values and defaults associated with symbols. *)
ClearAll["Global`*"];
Off[General::spell, General::spell1]
$DefaultFont = {"Times", 10};
$TextStyle = {FontFamily → "Times", FontSize → 10,
   FontSlant → "Italic"};
(* Constant parameters of motor*)
p = 4; omz = 120 * Pi;
vt = Sqrt[2 / 3] * 6000;
r1 = 0.460; r2 = 0.433; x1 = 3.51;
x2 = 5.05; xm = 95.6; J = 80;
T0 = (p / omz) * 8 * 10^5;
(*im1 function*)
im1[omm_, Tl_] := Module[{s, zr, zm, zag, zt, ia,
    ir, P, Q, pag},
   s = 1 - p * omm / omz;
   zr = r2 / s + I * x2;
   zm = I * xm;
   zag = zm * zr / (zm + zr);
   zt = zag + I * x1 + r1;
   ia = vt / zt;
   ir = ia * zm / (zm + zr);
   P = 1.5 * Re[vt * Conjugate[ia]];
   Q = 1.5 * Im[vt * Conjugate[ia]];
   pag = 1.5 * Abs[ir]^2 * r2 / s;
   {{P, Q}, ((pag / omz) p - Tl (p omm / (120 Pi))^2) / J}];
(*Initial Conditions and Time Interval*)
x0 = 0;
tspan = Range[0, 5, 0.001];
(*Simulation without load*)
Tl = 0.1;
solUnloaded = NDSolveValue[{x'[t] == im1[x[t],
       Tl][[2]], x[0] == x0}, x, {t, 0, 5}];
(*Under Load Simulation*)
Tl = T0;
solLoaded = NDSolveValue[{x'[t] == im1[x[t],
       Tl][[2]], x[0] == x0}, x, {t, 0, 5}];
(*RPM Values unloaded*)
rpmUnloaded = (60 / (2 Pi)) solUnloaded[tspan];
(*RPM Values loaded*)
rpmLoaded = (60 / (2 Pi)) solLoaded[tspan];
(*Visualisation:RPM Graph*)
```

```
g11 = ListLinePlot[{Transpose[{tspan, rpmUnloaded}],
     Transpose[{tspan, rpmLoaded}]},
   PlotRange → All, Frame → True,
   FrameLabel → {"Time(s)", "RPM"},
   PlotLabel → Style[Framed["First Order Model"],
     8, Blue, Background → Lighter[Yellow]],
   PlotLegends → {"Unloded", "Loaded"},
   LabelStyle → Directive[Bold, Blue, 8],
   ImageSize → {150, 150}];
Te11[t_] := Module[{sl, zrl, ztl, ial, irl, zagl, zm, pagl},
   sl = 1 - p * solLoaded[t] / omz;
   zrl = r2 / sl + I * x2;
   zm = I * xm;
   zagl = zm * zrl / (zm + zrl);
   ztl = zagl + I * x1 + r1;
   ial = vt / ztl;
   irl = ial * zm / (zm + zrl);
   pagl = 1.5 * Abs[irl]^2 * r2 / (sl);
   pagl];
(*Time Dependent Variation of Torque Graph*)
Punloaded[t_] := im1[solUnloaded[t], Tl][[1, 1]];
Ploaded[t_] := im1[solLoaded[t], T0][[1, 1]];
Teunloaded[t_] := Punloaded[t] / omz;
Teloaded[t_] := Ploaded[t] / omz;
(*Visualisation:Active Power Graph*)
g12 = Plot[{Punloaded[t], Ploaded[t]}, {t, 0, 5}, PlotRange → All,
   AxesLabel → {"Time(s)", "Watt"}, Frame → True,
   FrameLabel → {"Time(s)", "Watt"},
   PlotLabel → Style[Framed["First Order Model"], 8, Blue,
     Background → Lighter[Yellow]], PlotLegends → {"Unloded", "Loaded"},
   LabelStyle → Directive[Bold, Blue, 8], ImageSize → {150, 150}];
GraphicsGrid[{{g11, g12}}, Frame → All]
```

Appendix B

```
(*Reactive Power Fuction unloaded*)
Qunloaded[t_] := im1[solUnloaded[t], Tl][[1, 2]];
(* Reactive Power Fuction loaded*)
Qlloaded[t_] := im1[solLoaded[t], T0][[1, 2]];
(*Visualisation:Reactive Power Graph*)
g1 = Plot[{Teunloaded[t] / omz, Teloaded[t] / omz}, {t, 0, 5}, PlotRange → All, Frame → Tr
   FrameLabel → {"Time(s)", "Nm"}, PlotLegends → {"Unloaded", "Loaded"},
   PlotLabel → Style[Framed["First Order Model"], Blue, Background → Lighter[Yellow]]
   LabelStyle → Directive[Bold, Blue, 7], ImageSize → {150, 150}];
g2 = Plot[{Qunloaded[t], Qlloaded[t]}, {t, 0, 5}, PlotRange → All, Frame → True,
   FrameLabel → {"Time(s)", "VAR"},
   PlotLabel → Style[Framed["First Order Model"], Blue, Background → Lighter[Yellow]]
   PlotLegends → {"Unloded", "Loaded"}, LabelStyle → Directive[Bold, Blue, 7],
   ImageSize → {150, 150}];
Te[t_] := (im1[solUnloaded[t], Tl][[1, 1]]) / omz;
Tel[t_] := (im1[solLoaded[t], Tl][[1, 1]]) / omz;
g3 = Plot[{Te[t] / omz}, {t, 0, 5}, PlotRange → All, Frame → True,
   FrameLabel → {"Time(s)", "Nm"}, PlotRange → All,
   PlotLabel → Style[Framed["First Order Model"], Blue, Background → Lighter[Yellow]]
   PlotLegends → {"Unloded", "Loaded"}, LabelStyle → Directive[Bold, Blue, 7],
   ImageSize → {150, 150}];
g4 = Plot[{Te[t], Tel[t]}, {t, 0, 5}, PlotRange → All, Frame → True,
   FrameLabel → {"Time(s)", "Watt"},
   PlotLabel → Style[Framed["First Order Model"], Blue, Background → Lighter[Yellow]]
   PlotLegends → {"Unloded", "Loaded"}, LabelStyle → Directive[Bold, Blue, 7],
   ImageSize → {150, 150}];
```

```
(*Constants*)
Ls = (x1 + xm) / omz;
Lr = (x2 + xm) / omz;
M = xm / omz;
y11 = Lr / (Ls * Lr - M^2);
y22 = Ls / (Ls * Lr - M^2);
y12 = -M / (Ls * Lr - M^2);

Δdqdr[lamdad_, lamdadr_, lamdaq_, lamdaqr_] := Module[{ids, idr, iqs, iqr},
   ids = y11 * lamdad + y12 * lamdadr;
   iqs = y11 * lamdaq + y12 * lamdaqr;
   idr = y12 * lamdad + y22 * lamdadr;
   iqr = y12 * lamdaq + y22 * lamdaqr;
   Return[{ids, iqs, idr, iqr}];
  ];
(*Definition of system equation*)
im3[{lamdadr_, lamdaqr_, omm_}, Tl_] :=
  Module[{lamdad, lamdaq, omme, oms, ids, iqs, idr, iqr, dlamdadr, dlamdaqr, domm}
   lamdad = vt / omz;
   lamdaq = 0.00;
   omme = p * omm;
   oms = omz - omme;
   {ids, iqs, idr, iqr} = Δdqdr[lamdad, lamdadr, lamdaq, lamdaqr];
   dlamdadr = oms * lamdaqr - r2 * idr;
   dlamdaqr = -oms * lamdadr - r2 * iqr;
   domm = (1.5 * p / J) * (lamdad * iqs - lamdaqr * ids) - Tl * (omme / (120 Pi)) ^2 / J;
   {dlamdadr, dlamdaqr, domm}];
im33[{lamdadr_, lamdaqr_, omm_}, Tl_] :=
  Module[{lamdad, lamdaq, omme, oms, ids, iqs, idr, iqr, dlamdadr, dlamdaqr, domm}
   lamdad = vt / omz;
   lamdaq = 0.00;
   omme = p * omm;
   oms = omz - omme;
   {ids, iqs, idr, iqr} = Δdqdr[lamdad, lamdadr, lamdaq, lamdaqr];
   dlamdadr = oms * lamdaqr - r2 * idr;
   dlamdaqr = -oms * lamdadr - r2 * iqr;
   domm = (1.5 * p / J) * (lamdadr * iqs - lamdaqr * ids) - Tl * (omme / (120 Pi)) ^2 / J;
   {dlamdadr, dlamdaqr, domm}];
```

Speed characteristics for each loaded condition

```
(*Initial conditions and time interval*)
tspan = Range[0, 3, 0.001];
T0 = (p / omz) * 8 * 10^5;
Tl = 0.01;
n = 3;
X = Table[Subscript[x, i][t], {i, n}];
inCond = {Subscript[x, 1][0] == 0.00001, Subscript[x, 2][0] == 0.00001, Subscript[x, 3][0] == 0.00001};
(* System of Differential equations *)
difeqs = Thread[D[X, t] == im3[X, Tl]];
```

Appendix C

```
(* Solve the system of differential equations *)
solUnloaded = NDSolve[Transpose[{difeqs, inCond}], X, {t, 0, 5}];
omme[t_] := X[3] /. solUnloaded[1];
rpm[t_] := (30 / Pi) omme[t];
iq[t_] := y12 * X[2] /. solUnloaded[1];
id[t_] := y11 * vt / omz + y12 * X[1] /. solUnloaded[1];
Pin[t_] := 1.5 vt * iq[t] + r1 * (id[t]^2 + iq[t]^2);
Qin[t_] := 1.5 * vt * id[t];
(*Loaded initial state *)
Tl = T0;
difeqsl = Thread[D[X, t] == im33[X, Tl]];
solLoaded = NDSolve[Transpose[{difeqsl, inCond}], X, {t, 0, 5}];

(*Functions*)
ommel[t_] := X[3] /. solLoaded[1];
rpml[t_] := (30 / Pi) ommel[t];
iql[t_] := y12 * X[2] /. solLoaded[1];
idl[t_] := y11 * vt / omz + y12 * X[1] /. solLoaded[1];
Pinl[t_] := 1.5 vt * iql[t] + r1 (idl[t]^2 + iql[t]^2);
Qinl[t_] := 1.5 * vt * idl[t];
Teout[{lamdadr_, lamdaqr_, omm_}] :=
  Module[{lamdaq, lamdad, omme, oms, ids, iqs, idr, iqr, dlamdadr, dlamdaqr, domm},
   lamdaq = 0.00;
   lamdad = vt / omz;
   omme = p * omm;
   oms = omz - omme;
   {ids, iqs, idr, iqr} = Δdqdr[lamdad, lamdadr, lamdaq, lamdaqr];
   dlamdadr = oms * lamdaqr - r2 * idr;
   dlamdaqr = -oms * lamdadr - r2 * iqr;
   domm = (1.5) * (lamdadr * iqs - lamdaqr * ids);
   domm];
tork[t_] := Teout[{Subscript[x, 1][t], Subscript[x, 2][t],
     Subscript[x, 3][t]}] /. solUnloaded[1];
```

```
g11 = Plot[{rpm[t], rpml[t]}, {t, 0, 5}, PlotRange → All, Frame → True,
   FrameLabel → {"Time(s)", "RPM"}, PlotLabel → Style[Framed["Thrid Order Model"],
     Blue, Background → Lighter[Yellow]], PlotLegends → {"Unloded", "Loaded"},
   LabelStyle → Directive[Bold, Blue, 7], ImageSize → {150, 150}];
(*Active Power Graphics*)
g12 = Plot[{Pin[t], Pinl[t]}, {t, 0, 5}, PlotRange → All, Frame → True,
   FrameLabel → {"Time(s)", "Watt"}, PlotLabel → Style[Framed["Thrid Order Model"],
     Blue, Background → Lighter[Yellow]], LabelStyle → Directive[Bold, Blue, 7],
   ImageSize → {150, 150}];
(*Reactive Power Graphics*)
g13 = Plot[{Qin[t], Qinl[t]}, {t, 0, 5}, Frame → True, FrameLabel → {"Time(s)", "VAR"},
   PlotLabel → Style[Framed["Thrid Order Model"], Blue, Background → Lighter[Yellow]],
   LabelStyle → Directive[Bold, Blue, 7], AxesLabel → {"Time, Second", "VAR"},
   PlotLegends → {"Unloaded", "Loaded"}, ImageSize → {150, 150}];
g14 = Plot[{tork[t] / omz, torkl[t] / omz}, {t, 0, 5}, PlotRange → All, ImageSize → {150, 150},
   PlotLabel → Style[Framed["Thrid Order Model"], Blue, Background → Lighter[Yellow]],
```

```
(*Define functions*)
omm[t_] := x[3][t] /. solUnloaded[1];
rpm[t_] := (30 / Pi) omm[t];
omml[t_] := x[3][t] /. solLoaded[1];
rpml[t_] := (30 / Pi) omml[t];

(*Plot RPM*)
Plot[{rpm[t], rpml[t]}, {t, 0, 10}, PlotRange → All, PlotLabel → "Third Order Model", Frame → True,
 FrameLabel → {"Time, (s)", "RPM"}, PlotLegends → {"Unloaded", "Loaded"}, LabelStyle → Directive[Bold, Blue, 7],
 ImageSize → {150, 150}]
```

```
(*Constants*)
Ls = (x1 + xm) / omz;
Lr = (x2 + xm) / omz;
M = xm / omz;
y11 = Lr / (Ls * Lr - M^2);
y22 = Ls / (Ls * Lr - M^2);
y12 = -M / (Ls * Lr - M^2);
im5[{lamdad_, lamdaq_, lamdadr_, lamdaqr_, omm_}, Tl_] :=
  Module[{ids, iqs, idr, iqr, dlamdad, dlamdaq, dlamdadr, dlamdaqr, domm, omme, oms}, omme = p omm;
    oms = omz - omme;
    {ids, iqs, idr, iqr} = Δdqdr[lamdad, lamdadr, lamdaq, lamdaqr];
    dlamdad = omz lamdaq - r1 ids;
    dlamdaq = vt - omz lamdad - r1 iqs;
    dlamdadr = oms lamdaqr - r2 idr;
    dlamdaqr = - oms lamdadr - r2 iqr;
    domm = (1.5 p / J) (lamdad iqs - lamdaq ids) - Tl (omm / (120 Pi)) ^2 / J;
    {dlamdad, dlamdaq, dlamdadr, dlamdaqr, domm}];
im55[{lamdad_, lamdaq_, lamdadr_, lamdaqr_, omm_}, Tl_] :=
  Module[{ids, iqs, idr, iqr, dlamdad, dlamdaq, dlamdadr, dlamdaqr, domm, omme, oms}, omme = p omm;
    omme = p omm;
    oms = omz - omme;
    {ids, iqs, idr, iqr} = Δdqdr[lamdad, lamdadr, lamdaq, lamdaqr];
    dlamdad = omz lamdaq - r1 ids;
    dlamdaq = vt - omz lamdad - r1 iqs;
    dlamdadr = oms lamdaqr - r2 idr;
    dlamdaqr = - oms lamdadr - r2 iqr;
    domm = (1.5 p / J) (lamdad iqs - lamdaq ids) - Tl (omme / (120 Pi)) ^2 / J;
    {dlamdad, dlamdaq, dlamdadr, dlamdaqr, domm}];
(*Initial conditions and time span*)
tspan = Range[0, 10, 0.001];
n = 5;
X = Table[Subscript[x, i][t], {i, n}];
inCond = {Subscript[x, 1][0] == 0, Subscript[x, 2][0] == 0, Subscript[x, 3][0] == 0, Subscript[x, 4][0] == 0,
   Subscript[x, 5][0] == 0};
(*Unloaded case*)
TlUnloaded[t_] := If[t ≤ 5, 0.0000001, (p / omz) * 8 * 10^5];
difeqs = Thread[D[X, t] == im5[X, TlUnloaded[t]]];
solUnloaded = NDSolve[Join[difeqs, inCond], X, {t, 0, 10}];
omm[t_] := X[5] /. solUnloaded[1];
rpm[t_] := (30 / Pi) omm[t];
iq[t_] := y11 X[2] + y12 X[4] /. solUnloaded[1];
id[t_] := y11 X[1] + y12 X[3] /. solUnloaded[1];
Pin[t_] := 1.5 vt iq[t];
Qin[t_] := 1.5 vt id[t];

(*Loaded case*)
TlLoaded[t_] := If[t ≤ 5, (p / omz) * 8 * 10^5, 1.5 * (p / omz) * 8 * 10^5];
difeqsl = Thread[D[X, t] == im55[X, TlLoaded[t]]];
solLoaded = NDSolve[Join[difeqsl, inCond], X, {t, 0, 10}];

ommel[t_] := X[5] /. solLoaded[1];
rpml[t_] := (30 / Pi) ommel[t];
iql[t_] := y11 X[2] + y12 X[4] /. solLoaded[1];
idl[t_] := y11 X[1] + y12 X[3] /. solLoaded[1];
Pinl[t_] := 1.5 vt iql[t];
Qinl[t_] := 1.5 vt idl[t];
```

APPENDIX D

Manipulation script of the speed characteristic

```
Tl = 0.0001;
tspan = Range[0, 5, 0.001];
(*Boşta başlangıç durumu*)
n = 5;
X = Table[Subscript[x, i][t], {i, n}];
inCond = {Subscript[x, 1][0] == 0, Subscript[x, 2][0] == 0, Subscript[x, 3][0] == 0, Subscript[x, 4][0] == 0,
   Subscript[x, 5][0] == 0};
difeqsUnloaded = Thread[D[X, t] == im5[X, Tl]];
solUnloaded = NDSolve[Join[difeqsUnloaded, inCond], X, {t, 0, 5}];
(*Yüklü başlangıç durumu*)
Tl = Tl0;
difeqsLoaded = Thread[D[X, t] == im55[X, Tl]];
solLoaded = NDSolve[Join[difeqsLoaded, inCond], X, {t, 0, 5}];
(*rpm fonksiyonları*)
rpmUnloaded[t_] := (30 / π) (X /. solUnloaded)[1, 5];
rpmLoaded[t_] := (30 / π) (X /. solLoaded)[1, 5];
(*Manipulate*)
Manipulate[ListLinePlot[{Table[{t, rpmUnloaded[t]}, {t, tmin, tmax, Δt}], Table[{t, rpmLoaded[t]}, {t, tmin, tmax
  PlotRange → All, Frame → True,
```