Modeling, Analysis and Speed Control Design Methods of a DC Motor

Dr. Jamal A. Mohammed*

Received on: 23/12/2009
Accepted on:5 /1/2011

Abstract

Modern manufacturing systems are automated machines that perform the required tasks. The electric motors are perhaps the most widely used energy converters in the modern machine-tools and robots. These motors require automatic control of their main parameters (position, speed, acceleration, currents).

With the help of an example, a DC motor system, the use of MATLAB/Simulink for comprehensive study of modeling, analysis and speed control design methods has been demonstrated.

Keywords: DC motor, open loop, closed loop, system, speed control.

نمذجة وتحليل وطرق تصميم السيطرة على سرعة محرك التيار المستمر

الخلاصة:

أنظمة التصنيع الحديثة هي مكائن آلية تنجز المهمات المطلوبة. المحركات الكهربائية ربما تمثل محولات الطاقة الأكثر استخداماً في معدات المكائن الحديثة والإنسان الآلي.
تتطلب هذه المحركات سيطرة آلية لمعاملاتها الأساسية (كالموقع والسرعة والتعجيل والتيارات).
في البحث الحالي تمّ استعراض دراسة شاملة لنمذجة وتحليل وطرق تصميم السيطرة على السرعة لمنظومة المحرك ذو التيار المستمر باستخدام برنامج الـ MATLAB/Simulink.

1-Introduction:

The mechatronic systems, robots and low to medium power machine-tools often use DC motors to drive their work loads. These motors are commonly used to provide rotary (or linear) motion to a variety of electromechanical devices and servo systems.

There are several well known methods to control DC motors such as: Proportional-Integral PI, Proportional-Integral-Derivative PID or bipositional [1]. Despite a lot of researches and the huge number of different solutions proposed, most industrial control systems are based on conventional PID controllers.

The purpose of developing a control system is to enable stable and reliable control. Once the control system has been specified and the type of control has been decided, then the design and analysis are done.

There are three major objectives of system analysis and design: producing the desired transient response, reducing steady-state error, and achieving stability [2].

A simple technique for designing a DC motor speed controller with continuous time based on system theory concepts was presented in [1]. In [3], Rusu, et al successfully designed and implemented an open loop control system to control the speed of a DC motor. In [4], Ayasun

* Electromechanical Engineering Department ,University of Technology/Baghdad

and Nwankpa described the MATLAB/Simulink realization of the DC motor speed control methods (field resistance, armature voltage and resistance) and feedback control system for DC motor drives. In [5], Othman successfully designed a speed controller for closed loop operation of the BLDC motor. In [6], Allaoua, et al presented a new design method to determine optimal PID controller parameters using the Particle Swarm Optimization method.

The current paper intends presenting comprehensive study for all the techniques of modeling, analyzing, and designing a DC motor speed controller based on system theory concepts.

The speed control design methods used are studied under MATLAB and Simulink in the following sections.

2- MATLAB Control Design Method for Speed Modeling:

i- System Equations:

To perform simulations of a system, an appropriate model needs to be established. This paper was presented the system contains a DC motor based on the motor specifications needs to be obtained. This is achieved by developing the open loop transfer function of a DC motor with the use of the system equations of the motor as given by Rashid [7].

The design method uses the concepts of the system theory, such as signals and systems, transfer functions, direct and inverse Laplace transforms. This requires building the appropriate Laplace model for each component of the whole control system.

In order to build the DC motor's transfer function, its simplified mathematical model has been used. This model consists of differential equations for the electrical part, mechanical part and the interconnection between them. The electric circuit of the armature and the free body diagram of the rotor are shown in the Fig. 1. All the values for the physical parameters are listed in Table A in the Appendix.

The motor torque, T_m, is related to the armature current, i, by a constant factor K_t. The back *emf*, e_m, is related to the rotational speed, $\dot{\theta}$ by the following equations [7]:

$$T_m = K_t\, i \qquad \text{..... (1)}$$

$$e_m = K_e\, \dot{\theta} \qquad \text{..... (2)}$$

Assuming that, K_t (torque constant) = K_e (electromotive force constant) = K_m (motor constant).

From Fig. 1 and Table A, the following equations can be written based on Newton's law combined with Kirchhoff's law:

$$J\,\ddot{\theta} + b\,\dot{\theta} = K_m\, i - T_L \qquad \text{....(3)}$$

$$L_m \frac{di}{dt} + R_m\, i = V - K_m \dot{\theta} \qquad \text{..... (4)}$$

a- Transfer Function

Using Laplace Transforms, the above equations can be expressed in terms of *s*-domain.

$$s\left(J\,s + b\right)\dot{\theta}(s) = K_m I(s) - T_L(s) \qquad \text{..(5)}$$

$$\left(L_m s + R_m\right) I(s) = V(s) - K_m s\,\dot{\theta}(s) \qquad \text{....(6)}$$

By eliminating $I(s)$, the following open-loop transfer function can be obtained, where the rotational speed $\dot{\theta}$ is the output and the voltage V is the input.

When the motor is used as a component in a system, it is desired to describe it by the appropriate transfer function between the motor voltage and its speed. For this purpose assuming (load torque) T_L=0 and (friction torque) T_f=0, since neither affects the transfer function. Therefore

$$\left.\frac{\dot{\theta}(s)}{V(s)}\right|_{T_L(s)=0} = \frac{K_m}{(Js+b)(L_m s+R_m)+K_m^2}$$

[1,8,9] (7)

b. State-Space Representation:

In the state-space form, Eqs. 3 and 4 can be expressed by choosing $\dot{\theta}$ and i as the state variable and V as an input. The output is chosen to be $\dot{\theta}$.

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & \frac{K_m}{J}\\ -\frac{K_m}{L_m} & -\frac{R_m}{L_m}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L_m}\end{bmatrix}V$$

$$\dot{\theta} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} \quad \ldots.(8)$$

ii- Design Requirements:

Since the most basic requirements of a motor are that it should rotate at the desired speed, the steady-state error e_{ss} of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed $\dot{\theta}_{ss}$ as soon as it turns on. In this case, it is desirable to have a settling time T_s in less than 2sec. Since speed faster than the reference may damage the equipment, a peak overshoot *Mp* of less than 5% is wanted.

The definition of these parameters can be demonstrated as in [10].

iii- MATLAB Representation and Open-loop Response:

The transfer function in Eq. 7 can be represented into MATLAB by defining the numerator (*num*) and denominator (*den*) matrices as follows: $num=K_m$ and $den=(Js+b)(L_m s+R_m)+K_m^2$, respectively, and the corresponding plot is shown as in Fig. 2.

The motor is assumed to be exposed at steady-state speed to step change in speed with 0.1 p.u. (10% of its steady-state speed).

From the plot, it can be seen that when a step input voltage is applied to the system at steady-state speed, the motor can only achieve a maximum speed of 0.11 p.u.; ten times smaller than our desired speed (1 p.u.). The motor parameters were determined on the basis of its step response about a steady-state operating point, namely input voltage V= 6 volts and speed $\dot{\theta}$ = 600 r/s. Also, it makes the motor takes 3sec to reach its steady-state speed; this does not satisfy the 2sec settling time criterion.

The system can also be represented using the state-space Eqs. 8 to get the same output plot obtained in Fig. 2.

3- Continuous PID Control Design Method:

PID controllers are commonly used to regulate the time-domain behavior of many different types of dynamic plants [8].

Considering the unity feedback system architecture as shown in Fig. 3, where it can be assumed that the plant is a DC motor whose speed must be controlled and the Controller provides the excitation for the plant; designed to control the overall system behavior.

Different characteristics of the motor response (steady-state error, peak overshoot, rise time, etc.) are controlled by selection of the three gains that modify the PID controller dynamics.

First, taking a look at how the PID controller works in the closed-loop system shown in Fig. 3. The variable *e* represents the tracking error; the difference between the desired input value R and the actual output Y. This error signal will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal.

Therefore, the PID controller is defined by the relationship between the controller input e and the controller output u that is applied to the motor armature:

$$u = K_P\, e + K_I \int e\, dt + K_D \frac{de}{dt} \qquad (9)$$

, where K_P = Proportional gain, K_I = Integral gain, and K_D=Derivative gain.

The signal u will be sent to the plant, and the new output Y will be obtained and sent back to the sensor again to find the new error signal e. The controller takes e and computes its derivative and it's integral again. This process goes on and on. By adjusting the weighting constants K_P, K_I, and K_D, the PID controller can be set to give the desired performance. Effects of each of these parameters on a closed-loop system are summarized as in [8].

Taking the Laplace transform of Eq. 9 gives the following transfer function:

$$K(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D\, s$$

$$= \frac{K_D\, s^2 + K_P\, s + K_I}{s} \quad [8] \qquad \ldots.(10)$$

This transfer function clearly illustrates the proportional, integral, and derivative gains that make up the PID compensation.

With adding 0.1 p.u. step input to the steady-state speed and the design criteria previously mentioned, the PID controller can be designed and added into the system, recalling the transfer function of a PID controller in Eq. 10.

i. Design Steps of PID Controller:

To designing a PID controller for a given system, the following steps must be followed to obtain a desired response.

1. Obtaining an open-loop response and determining what needs to be improved.
2. Adding a K_P to improve the rise time.
3. Adding a K_D to improve the overshoot (increase damping).
4. Adding a K_I to eliminate the steady-state error.
5. Adjusting each of K_P, K_I, and K_D until obtaining a desired response.

The MATLAB is used to obtain a closed-loop transfer function directly from the open-loop transfer function.

Using K_P with a gain of 100, the step response can be shown as in Fig. 4.

From the plot in Fig. 4, it can be seen that, both the steady-state error and the overshoot are too large. Adding an integral term will eliminate the steady-state error and a derivative term will reduce the overshoot. Trying a PID controller with small K_I and K_D, the resulting response will be plotted as in Fig. 5.

ii. Tuning the Gains:

By tuning the gains of the PID controller and producing the optimum response using trial and error method (it provides simplest way to achieve a good compensator [11]), the simulation start with the best initial gains.

The settling time shown in Fig. 5 is too long. So it is necessary to increase K_I to reduce the settling time. Changing K_I to 200 makes the response looks as in Fig. 6.

The response in the Fig. 6 seems much faster than before, but the large K_I has worsened the transient response (big overshoot). Therefore, K_D must increase to reduce the overshoot.

Changing K_D to 10 can get the plot seen in Fig. 7.

So, it must be known that using (after several trial and error runs) a PID

controller with K_P=100, K_I=200, and K_D=10, all of the design requirements will be satisfied, providing the desired response.

4- Root Locus Control Design Method:

For designing a controller using the root locus method, it should follow the following steps:

i. Open-loop Root Locus:

The main idea of root locus design is to find the closed-loop response from the open-loop root locus plot. Then by adding zeros and/or poles to the original plant, the closed-loop response can be modified. So, first, it is necessary viewing in the root locus for the plant.

Three arguments must be founded are the damping ratio (*zeta*) term (ξ > 0.8 corresponds to an overshoot *Mp* < 5%), the natural frequency (ω_n) term (= 0 corresponds to no rise time criterion), and the *sigma* term (4.6/2 = 2.3 sec). Using these criteria results a root locus plot shown in Fig. 8.

The following three equations are used in continuous system designs:

$$\xi\,\omega_n \geq 4.6 / T_s \quad \text{(Sigma term)} \qquad (11)$$

$$\omega_n \geq 1.8 / T_r \qquad \ldots.(12)$$

$$\xi \geq [((\ln Mp/\pi)^2/(1+(\ln Mp/\pi)^2)]^{0.5} \qquad \ldots.(13)$$

, where ξ: Damping ratio, ω_n: Natural frequency, T_s: Settling time, T_r: Rise time and *Mp*: Peak overshoot [11].

ii. Finding the Gain:

Recalling that, the settling time and the overshoot must be as small as possible. Large damping corresponds to points on the root locus near the real axis. A fast response corresponds to points on the root locus far to the left of the imaginary axis. Matlab can find the gain corresponding to a point on the root locus. The gain can be found and the step response plotted using this gain all at once.

When a point is selected on the root locus plot in Fig. 8, half-way between the real axis and the damping requirement, say at -6+2.05*i*, the MATLAB should return the output similar to the following: Selected point=-6.0204+2.0401*i*, gain; K=10.0708, poles=-6.0000±2.0400*i*, *Mp* %=0.0101 and the ξ = 0.947.

Noticing that, the values returned in MATLAB results may not be exactly the same, but should at least have the same order of magnitude. With the above results shown in Fig. 9, it can be seen that, there is no overshoot and the settling time is about 1sec, so the overshoot and settling time requirements are satisfied. The only problem can be seen from this plot is that; the steady-state error is about 50%. If the gain is increased to reduce the steady-state error, the overshoot becomes too large. So it is necessary adding a Lag Controller (in transfer function form) to reduce the steady-state error.

iii. Adding a Lag Controller:

The plot in Fig. 8 shows a very simple root locus. The damping and settling time criteria were met with the proportional controller. The steady-state error is the only criterion not met with the proportional controller. A Lag Compensator [11] can reduce the steady-state error. By doing this, it might however increase the settling time. By using the following lag controller first:

$$(s + 1) / (s + 0.01) \qquad \ldots.(14)$$

, the root locus will be shown as in Fig. 10, which looks very similar to the original one.

iv. Closed-loop Response:

By closing the loop and seeing the closed-loop step response, the

following Matlab results will be obtained: selected point = -5.6122+ 4.4444*i*, gain; K=19.9563, poles=-5.6138±4.4445*i*, and -0.7824.

When prompted to select a point, pick one that is near the damping requirement (diagonal dot line in Fig. 10). The resultant plot will be shown as in Fig. 11.

The gain should be about 20. It can be seen from the figure that the response is not quite satisfactory. It may also note that even though the gain was selected to correlate with a position close to the damping criterion, the overshoot is not even close to 5%. This is due to the effect of the lag controller; its pole is much slower. What this means is that it can go beyond the dotted lines that represent the limit, and getting the higher gains without worrying about the overshoot.

Keep trying until getting a satisfactory response. It should look similar to the following (using a gain of around 50): Selected point= -5.5714+8.8889*i*, gain; K = 49.8876, poles = -5.5498 ± 8.8887*i* and -0.9104.

Fig. 12 shows the corresponding response with gain =50.

It can be seen from the figure that the steady-state error is smaller than 1%, and the settling time and overshoot requirements have been met. As result, it concluded that, the design process for root locus is very much a trial and error process. That is why it is nice to plot the root locus, picking the gain, and plotting the response all in one step. If it has not been able to get a satisfactory response by choosing the gains, it can have tried a different lag controller, or even added a Lead Controller.

5- Frequency Control Design Method:

i. Original Bode Plot:

The main idea of frequency-based design is to use the Bode plot of the open-loop transfer function to estimate the closed-loop response.

Adding a controller to the system changes the open-loop Bode plot, therefore changing the closed-loop response. First, the Bode plot for the original open-loop transfer function is drawn as shown in Fig. 13.

ii. Adding Proportional Gain:

From the bode plot in Fig. 13, it can be seen that the phase margin Φ_m [11] can be greater than about (180^0-120^0=60^0) if ω is less than 10 r/s. Adding gain to the system so the bandwidth frequency [11] is 10 r/s, which will give Φ_m of about 60^0. The gain at 10 r/s can be found by reading off the Bode plot in Fig. 13 (it looks to be slightly more than -40dB, or 0.01 in magnitude) and the exact magnitude: mag = 0.0139. To have an open-loop gain of 1 (0dB) at 10 r/s, it may be needed to add a control gain of 1/0.0139≈72 (≈37dB) to get the Bode plot shown as in the Fig. 14.

From this plot: (mag=0.0139, phase = -123.6835, and ω=10 r/s).

iii. Closed-loop Response:

From the plot in Fig. 14, it can be seen that the phase margin is now quite large and the corresponding closed-loop response looks like as in Fig. 15. The figure shows that the settling time is fast enough, but the overshoot and the steady-state error are too high.

iv. Adding a Lag Controller:

Reducing the gain to 50, and trying the lag controller in Eq. 14, which should reduce the steady-state error by a factor of 1/0.01=100 (but could increase the settling time) gives the bode plot shown in Fig. 16 with

(K=50≈34dB) with good phase margin (≈70^0).
Closing the loop gives the step response shown in Fig. 17, which meets the design requirements.
It can see that adding a lag controller can reduce the steady-state error and at the same time, the overshoot can be reduced by reducing the gain.

6- A State-Space Control Design Method:

For the main problem, the dynamic equations in state-space form are given as in the Eqs. 8.

i. Designing the Full-state Feedback Controller:

The schematic for a full-state feedback system is shown as in the Fig. 18.
The characteristic polynomial for this closed-loop system is the determinant of (*s*I-(A-BK)) where *s* is the Laplace variable. Since the matrices A and BK are both 2x2 matrices, there should be 2 poles for the system. By designing a full-state feedback controller, these two poles can move anywhere wanted them. The first try is to place them at:
-5±i ;(poles= $-\xi\omega_n \pm j\omega_n (1-\xi^2)^{0.5}$) [11] corresponds to ξ = 0.98 which gives 0.1% overshoot and a sigma=$\xi\omega_n$=5 which leads to≈1sec settling time by using Eqs. (11-13).
Once coming up with the wanted poles, MATLAB will find the controller matrix, K.
After adding the K matrix into the system in Fig. 18, the state-space equations become:

$$\dot{X} = (A - BK)X + Bu$$
$$Y = CX \qquad [11]$$
$$\ldots.(15)$$

, and the plot of the closed-loop response can be shown as in Fig. 19.

ii. Adding a Reference Input:

From the plot in Fig. 19, it can be seen that the steady-state error is too large. Computing the input can be done in one step by adding a constant gain *Nbar* after the reference R. The Matlab can find the scale factor *N* which will eliminate the steady-state error to a step reference for a continuous-time, single-input system with full-state feedback using the schematic shown in Fig. 20. The corresponding step respond plot is shown in the Fig. 21 and the MATLAB results are: *Nbar*=10.0048 and poles = -4.3600±1.0000i.
Now, the steady-state error is much less than 1% and all the other design criteria have been met as well.

7- Discrete PID Control Design Method:

A digital DC motor model can be obtained from conversion of the analog model. The controller for this example will be designed by a PID method.
To discretise the controller, the integral and derivative terms of the controller have to be approximated. The integral becomes a sum and the derivative becomes a difference. The continuous time signal *e* is sampled at a sample period *T* [8].

i. Continuous to Discrete Conversion:

The first step in designing a discrete control system is to convert the continuous transfer function in Eq. 7 to a discrete one [12] as follows:

$$\frac{\dot{\theta}(z)}{V(z)} = \frac{0.0092z + 0.0057}{z^2 - 1.0877z + 0.2369} \qquad \ldots.(16)$$

The *numz* of Eq. 16 shown above, has one extra zero in the front, so, must rid of it before closing the loop.

By generating the vector of discrete output signals and connecting them, the closed-loop response without any control will be shown as in Fig. 22.

From the design requirement, the sampling time $T = 0.1*\tau = 0.1*0.6T_s = 0.12$sec, (which is 1/10 the time constant τ of the system with a settling time T_s of 2sec) [8].

ii. PID Controller:

For mapping the continuous-time transfer function for a PID controller in Eq. 10 from the *s*-plane to *z*-plane, the bilinear transformation [10] will be used. According to the PID design method for the DC motor in Sec. 4, K_P=100, K_I=200 and K_D=10 are satisfied the design requirement. All of these gains will be used in this example. Finally, the close-loop stair step response will be plotted as shown in Fig. 23.

It can be seen from this figure that, the closed-loop response of the system is unstable. Therefore, there must be something wrong with compensated system. So, it should take a look at root locus of the compensated system shown in Fig. 24.

From this root-locus plot, it can be seen that the PID controller has a pole at -1 in the *z*-plane, outside the unit circle and the system will be unstable. The pole location can be changed by changing the compensator design. It is chosen to cancel the zero at -0.62. This will make the system stable for at least some gains.

Furthermore, an appropriate gain can be chosen from the root locus plot to satisfy the design requirements.

The new *z*-plane will have a pole at -0.625 instead of -1, which almost cancels the zero of uncompensated system as shown on the root-locus plot in Fig. 25. The MATALB results are: ξ =1 and *Mp*=0.

Then MATALB will return the appropriate gain and the corresponding compensated poles, and plot the closed-loop compensated response as shown in Fig. 26, where: selected point=-0.6265-0.0*i*, K=68.3287, poles=-0.6265, 0.7344±0.1294*i*, and 0.2275. The plot shows that the settling time is less than 2 sec and the overshoot is around 3%. In addition, the steady-state error is zero. Therefore this response satisfies all of the design requirements.

8- Simulink Design Method for Speed Modeling:

A basic feedback control system shown in Fig. 3 represents a very common block diagram form.

For the proposed system, the *Plant* will be a DC motor.

i. Building the Model:

The motor system shown in Fig. 1 will be modeled by summing the torques acting on the rotor inertia and integrating the acceleration to give the velocity. Also, Kirchoff's laws will be applied to the armature circuit. First, the integrals of the rotational acceleration and of the rate of change of armature current will be modeled:

$$\int \frac{d^2\theta}{dt^2} = \frac{d\theta}{dt}, \text{ and } \quad \int \frac{di}{dt} = i \qquad \ldots(17)$$

Next, both Newton's law and Kirchoff's law will be started to model. These laws applied to the motor system to give the following equations:

$$J\frac{d^2\theta}{dt^2} = T_m - b\frac{d\theta}{dt}$$

$$\frac{d^2\theta}{dt^2} = \frac{1}{J}\left(K_t\, i - b\frac{d\theta}{dt}\right) \qquad \ldots.(18)$$

$$L_m\frac{di}{dt} = -R_m\, i + V - e_m$$

$$\frac{di}{dt} = \frac{1}{L_m}\left(-R_m\, i + V - K_m \frac{d\theta}{dt}\right). \quad (19)$$

ii. Open-loop Response:

To simulate the motor system, first, an appropriate simulation time must be set. By selecting parameters from the Simulation menu and entering "3" in the Stop Time field, the open-loop response can be viewed. The physical parameters must be set as J=0.01; b=0.1; K_m=0.01; R_m=1; and L_m=0.5. Running the simulation model shown in the Fig. 27 will give the output on the Scope shown in the Fig. 28.

iii. Extracting a Linear Model into MATLAB:

A linear model of the system (in state space or transfer function form) can be extracted from a Simulink model into MATLAB.

To verify the model extraction, an open-loop step response of the extracted transfer function will be generating in MATLAB. The corresponding plot will be exactly similar to that shown in Fig. 2 which is equivalent to the Scope's output.

iv. Implementing Lag Compensator Control:

In the motor speed control root locus example in Sec. 5, a Lag Compensator was designed with the following transfer function: $50(s+1)/(s+0.01)$.

To implement this in Simulink, the open-loop system model in Fig. 27 will be contained in a Subsystem block, and inserting a Lag Compensator into a closed-loop around the plant model by feeding back the plant output as shown in Fig. 29 . The output of the Sum block will provide the error signal, which will be feed into a Lag Compensator. Finally, a step input is applying for viewing the output on the scope.

v. Closed-loop Response:

To simulate the proposed system, first, an appropriate simulation time must be set, then selecting parameters from the Simulation menu and entering "3" in the Stop Time field.
3sec is suitable for simulation, since it is more than the desired settling time. The physical parameters must be set. As a result, the Scope output will be shown as in Fig. 30.

9-Conclusions:

There are many motor control system design methods that may be more or less appropriate to a specific type of application. The designer engineer must choose the best one for his work. Therefore, the current work makes comprehensive study of the analysis, modeling and speed control design techniques of a DC motor.

An important advantage of the root-locus method is that the roots of the characteristic equation of the system can be obtained directly, which results in a complete and accurate solution of the transient and steady-state response of the controlled variable.

The frequency-response approach yields enough information to indicate whether the system needs to be adjusted or compensated and how the system should be compensated.

PID controller enables the motor to reach the speed smoothly and within an acceptable period of time. It found that using (after several trial and error runs) a PID controller with K_P=100, K_I=200, and K_D=10, all of the design requirements will be satisfied, providing the desired response

It has observed that both the transfer function and the PID control could have a large influence upon the response of the system.

It is found the design process for root locus is very much a trial and error process.

Future suggestion is to replace the conventional speed controller by adaptive self tuning of PID controller.

10- References:

[1] M. S. RUSU, and L. Grama, The Design of a DC Motor Speed Controller, Fascicle of Management and Tech. Eng., Vol. VII (XVII), 2008, pp. 1055-1060.

[2] J. J. Jordan, A DC Motor Drive for a Dyno-Microcontroller and Power Electronics, University of Queensland, Australia, BSc. thesis, Oct. 2001.

[3] J. Santana, J. L. Naredo, F. Sandoval, I. Grout, and O. J. Argueta, "Simulation and Construction of a Speed Control for a DC Series Motor," Mechatronics, Vol. 12, issues 9-10, Nov.-Dec. 2002, pp. 1145-1156.

[4] S. Ayasun and C. O. Nwankpa, DC Motor Speed Control Methods Using MATLAB/Simulink and Their Integration into Undergraduate Electric Machinery Courses, IEEE Trans Educ, March, (2007), 347-354.

[5] A. S. Othman, "Proportional Integral and Derivative Control of Brushless DC Motor," European Journal of Scientific Research, Vol. 35 No. 2 (2009), pp. 198-203.

[6] B. Allaoua, B. Gasbaoui and B. Mebarki, "Setting Up PID DC Motor Speed Control Alteration Parameters Using Particle Swarm Optimization Strategy", Leonardo Electronic Journal of Practices and Technologies, Issue 14, Jan.-June 2009, pp. 19-32.

[7] M. H. Rashid, Power Electronics, Circuits Devices and Applications, 2nd Edition, Prentice- Hall International, Inc., New Jersey, 1993.

[8] B. Shah, Field Oriented Control of Step Motors, MSc. Thesis, SVMIT-Bharuch, India, Dec. 2004.

[9] W. P. Aung," Analysis on Modeling and Simulink of DC Motor and its Driving System Used for Wheeled Mobile Robot", World Academy of Science, Engineering and Technology 32, 2007, pp.299-306.

[10] B. Kuo, Automatic Control Systems, Prentice-Hall, Englewood, Cliffs. NJ, 1995.

[11] N. S. Nise, Control Systems Engineering (3rd Edition), John Wiley and Sons Inc., New York, 2000.

[12] A. Klee, Development of a Motor Speed Control System Using MATLAB and Simulink, Implemented with a Digital Signal Processor, MSc. Thesis, Orlando, Florida, 2005.

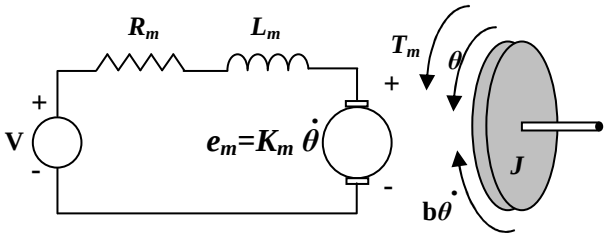

Fig. 1: The electric circuit of the armature and the free body diagram of the rotor for a DC motor

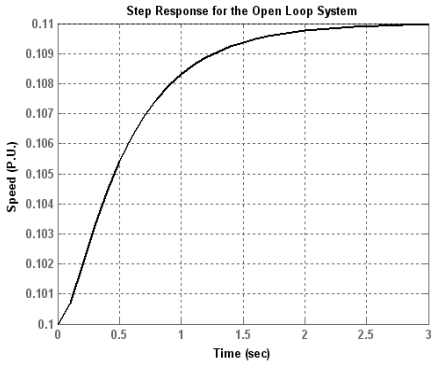

Fig. 2: The Open-loop step response of the motor system using T. F. or state-space; with initial speed=0.1 p.u. at steady state

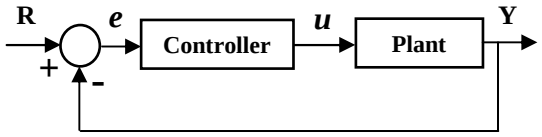

Fig. 3: Feed back system schematic

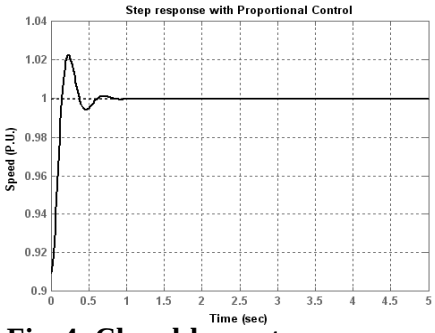

Fig. 4: Closed-loop, step response; K_P = 100

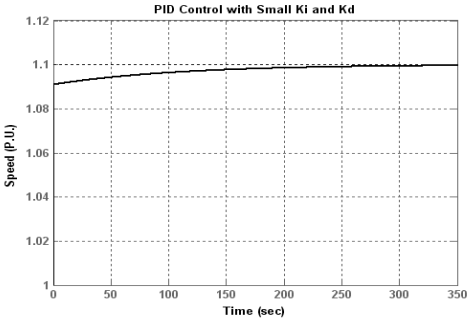

Fig. 5: Step response using a PID controller with small K_I and K_D

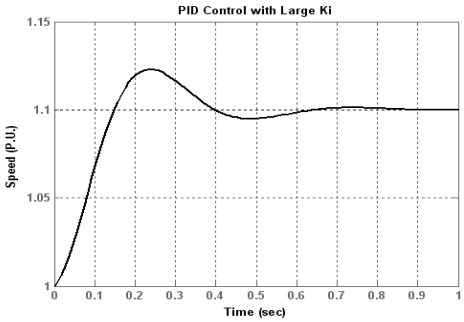

Fig. 6: Step response using a PID controller with large K_I

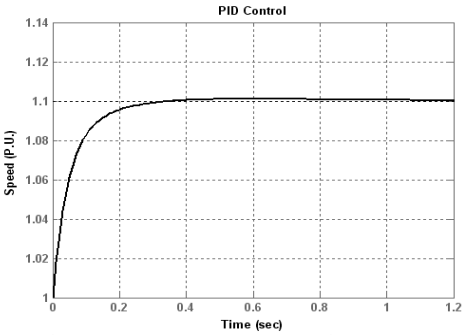

Fig. 7: Step response using a PID controller: K_P=100, K_I=200, K_D=10

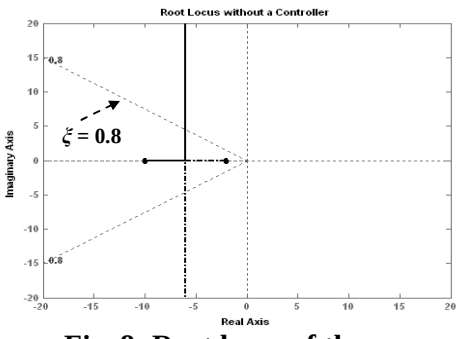

Fig. 8: Root locus of the open-loop system

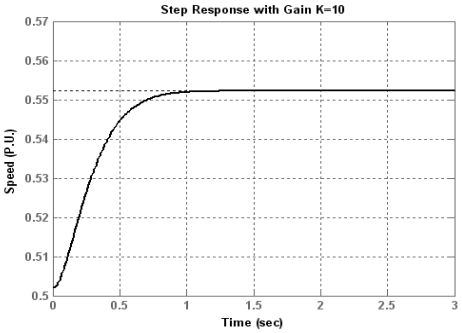

Fig. 9: Open-loop system, step - response with gain; K=10

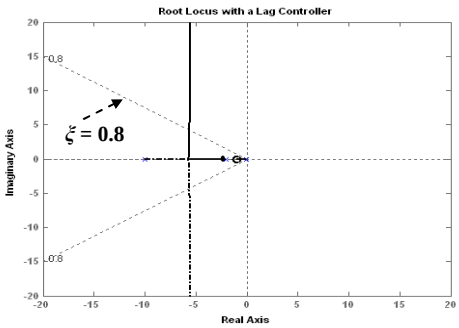

Fig. 10: Root locus with a lag controller

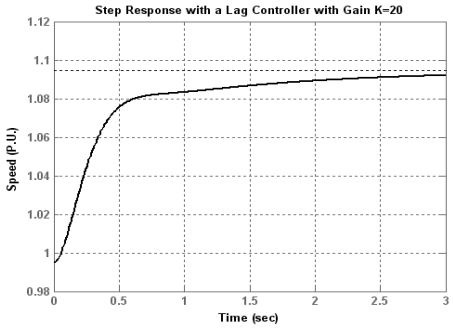

Fig. 11: Closed-loop step-response with a lag controller and gain=20

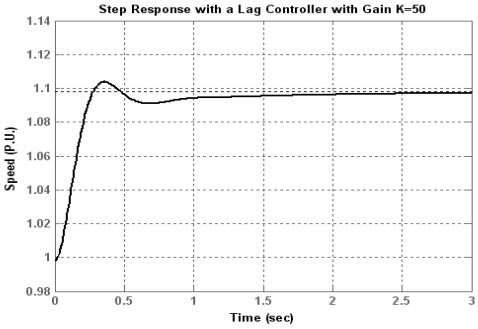

Fig. 12: Closed-loop, step-response with a lag controller and gain=50

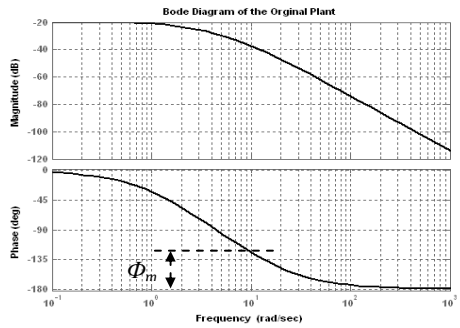

Fig. 13: Bode plot of the original open - loop transfer function

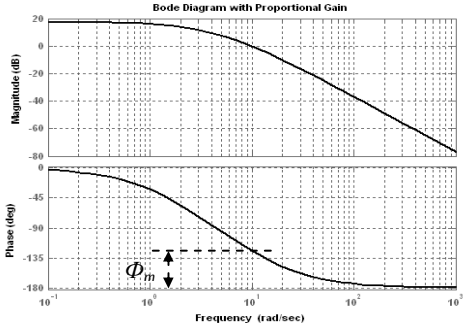

Fig. 14: Bode plot of the plant with proportional gain=72

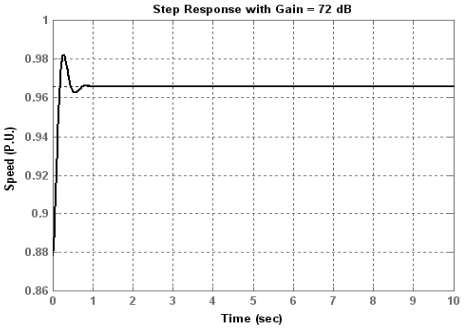

Fig. 15: Step response of the open - loop system with gain=72

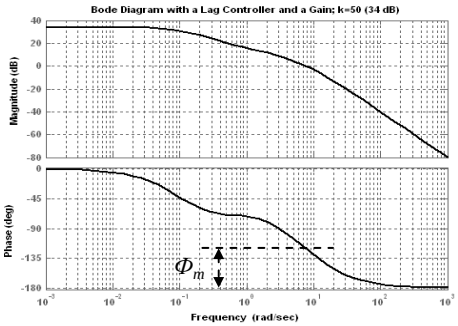

Fig. 16: Bode plot of the plant with a lag controller and gain=50

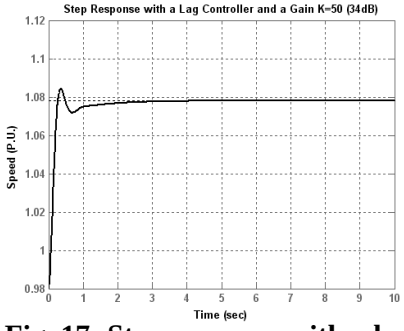

Fig. 17: Step response with a lag controller and gain=50

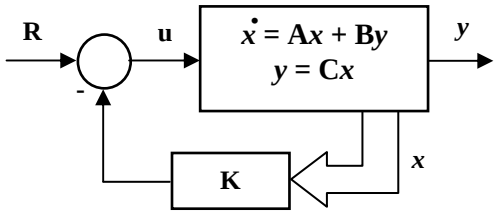

Fig. 18: The schematic for a full - state feedback system

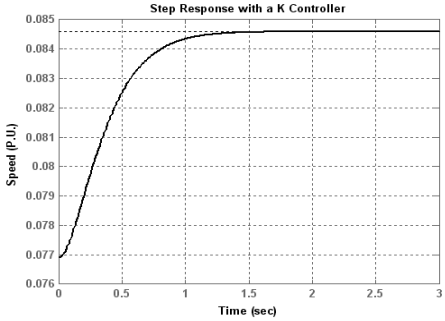

Fig. 19: Closed-loop step response with a K controller

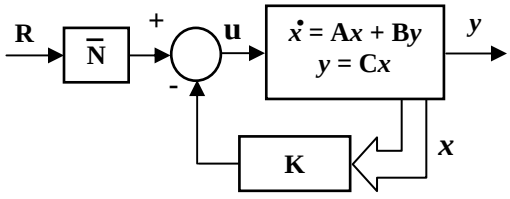

Fig. 20: Schematic for a full-state feed -back system with adding *Nbar* gain=10

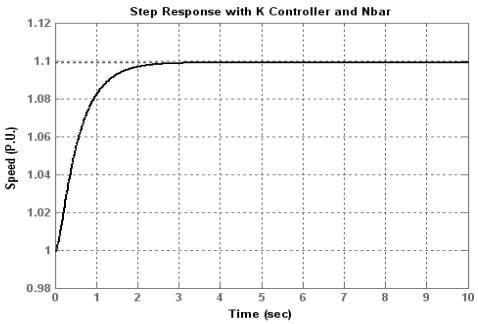

Fig. 21: Closed-loop, step-response with adding *Nbar* =10

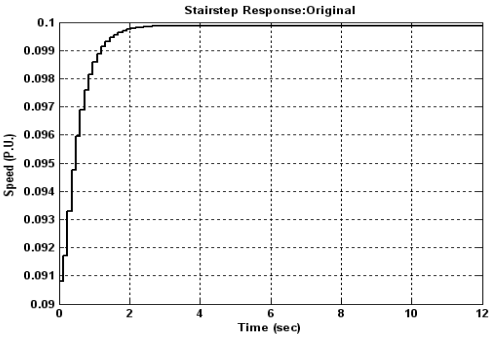

Fig. 22: Closed-loop, stair-step response with no control

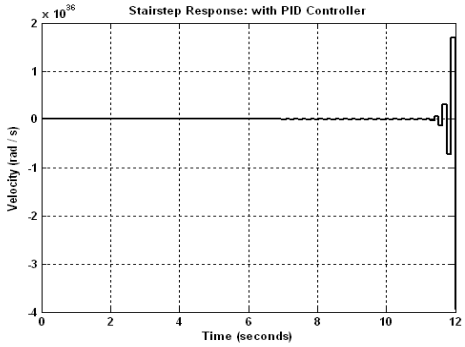

Fig. 23: Close-loop stair-step response

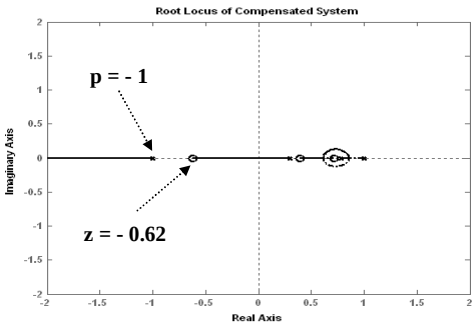

Fig. 24: Root locus of the compensated system with discrete PID controller

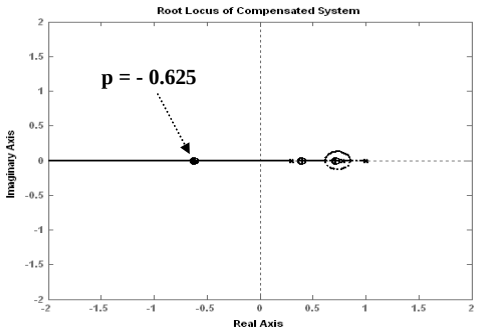

Fig. 25: Root locus of the stable system

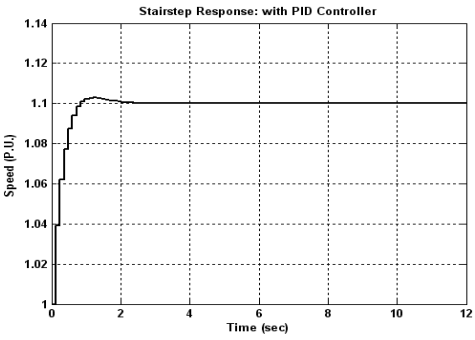

Fig. 26: Stair-step stable response with PID controller

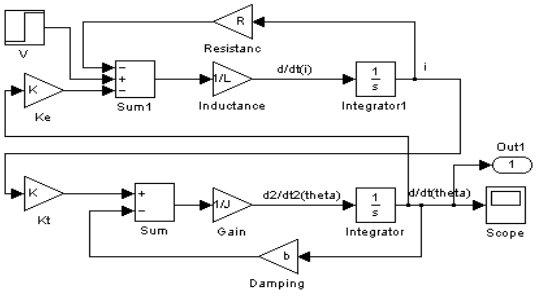

Fig. 27: Simulation of the open-loop motor system model

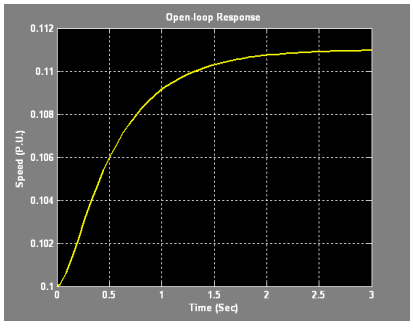

Fig. 28: Open-loop step response; scope output

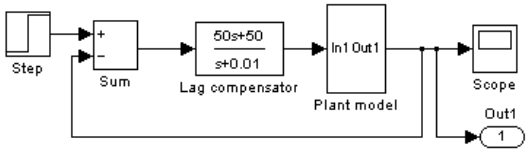

Fig. 29: Closed-loop system using lag compensator control

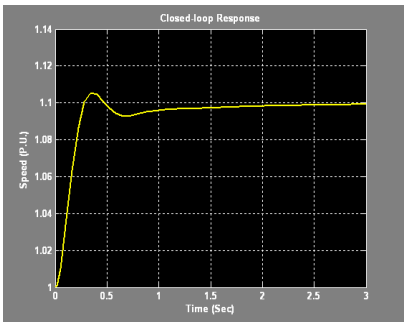

Fig. 30: Closed-loop step response; scope output

Appendix

Table A: Physical parameters of the DC motor

J	Moment of Inertia of the Rotor (kg.m^2)	0.01
b	Damping ratio of the Mechanical System (Nms)	0.1
K_m	Motor Constant (Nm/A)	0.01
R_m	Motor Electric Resistance (Ω)	1
L_m	Motor Electric Inductance (H)	0.5
V	Input Voltage (Volt)	6
$\dot{\theta}$	Rotating Speed (r/s)	600

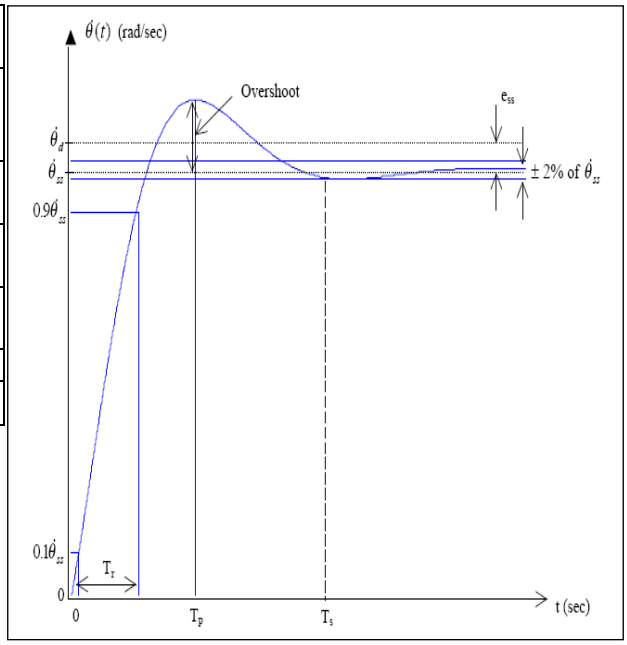

Fig. A: Transient response (closed loop) of a DC motor