European Scientific Journal December 2013 edition vol.9, No.36 ISSN: 1857 – 7881 (Print) e - ISSN 1857- 7431

MECHATRONICS MOTION CONTROL DESIGN OF ELECTRIC MACHINES FOR DESIRED DEADBEAT RESPONSE SPECIFICATIONS, SUPPORTED AND VERIFIED BY NEW MATLAB BUILT-IN FUNCTION AND SIMULINK MODEL

Farhan A. Salem, PhD
Department of Mechanical Engineering, Faculty of Engineering,
Mechatronics program, Taif University, Taif, Saudi Arabia
Alpha center for Engineering Studies and Technology Researches,
Amman, Jordan

Abstract

To help in facing the two top challenges in developing Mechatronics motion systems, particularly, early identifying system level problems and ensuring that all design requirements are met, this paper presents Mechatronics design of electric machine and corresponding motion control in terms of desired output position or velocity for desired deadbeat response specifications, the design is facilitated using either or both proposed new MATLAB built-in function named deadbeat( ), and new Simulink model, both oriented on Mechatronics design of both electric machine and motion control systems, based on system parameters, shape and dimensions to meet desired deadbeat response with desired settling time. The introduced new MATLAB built-in function determines the coefficients that yield the optimal deadbeat response for desired output speed or angle control of a given electric machine, also calculates transfer functions (open and closed), of used motor, controller, prefilter and overall system, also, plots overall system deadbeat response and all corresponding response curves, finally, calculates main overall system deadbeat response specification, particularly *Mp, Ess, Ts*. The proposed design, model and function, can be used for various Mechatronics motion control design applications, where the proper selection of actuating machine and design of precise motion control system are of concern.

Keywords: Mechatronics system, motion control, deadbeat response, DC machine, MATLAB built-in function

1. Introduction

The term control system design refers to the process of selecting feedback gains that meet design specifications in a closed-loop control system. Motion control is a sub-field of control engineering, in which the position or velocity of a given machine are controlled using some type of actuating machine. The accurate control of motion is a fundamental concern in mechatronics applications, where placing or moving an object in the exact desired location or with desired speed with the exact possible amount of force and torque at the correct exact time, while consuming minimum electric power, at minim cost, is essential for efficient system operation.

Because of the ease with which they can be controlled, systems of DC machines have been frequently used in many Mechatronics applications requiring a wide range of motor speeds and a precise output motor control ,there are many DC machines that may be more or less appropriate to a specific type of application each has its advantages, limitations and disadvantages; the actuating machines most used in mechatronics motion control systems are PMDC motors.

Often, the goal for a motion control system is to achieve a fast response to a step command with minimal overshoot, there are many motor motion control strategies that may be more or less appropriate to a specific type of application each has its advantages and disadvantages, the designer must select the best DC machines and corresponding motion control strategy for specific application and desired overall response. Different researches on this theme, can be found including (Richard C. Dorf,2001)(Thomas R. Kurfess, 2005)(Mohsen Shahinpoor, 1987)(Bashir M. Y. Nouri, 2005)(Chun Htoo Aung et al, 2008)(Ramjee Prased)(Ahmad A. Mahfouz et al, 2013)(Wai Phyo Aung, 2007)(Farhan A. Salem, 2013)(Farhan A. Salem, 2013)(Farhan A. Salem, 2013), most of these researches, study separate systems and applications design and control issues. To help in facing the two top challenges in developing Mechatronics motion systems, particularly, early identifying system level problems and ensuring that all design requirements are met, this paper presents Mechatronics design of electric machine and corresponding motion control in terms of desired output position or velocity for desired deadbeat response specifications, the proposed design can be used for various Mechatronics motion control design applications, where the proper selection of actuating machine and design of precise motion control system are of concern ,in this paper writer is most concerned with controlling the output angular and/or linear motions and meeting the known finite settling time, of a given DC motor for specific application and satisfying all deadbeat response characteristics.

A deadbeat response is response that proceeds rapidly to the desired reference trajectory level at *minimum* amount of time without error and holds

at that level with minimal overshoot for step input. The ± 2 % *error* band at the desired level could be an acceptable range of variation from the desired response. Then, if the response enters the ±2% band at time T_s, it has satisfied the settling time upon entry to the band. The deadbeat control could be used in systems where the known finite settling time is required. A deadbeat response has the following characteristics, (1) Steady-state error = 0, (2) Fast response; minimum both rise time T_R and settling time T_s, (3) 0.1% < percent overshoot <2%, (4) Percent undershoot <2% (The ± 2 % error band). Characteristics (3) and (4) require that the response remain within the ±2% band so that the entry to the band occurs at the settling time T_s. (Richard C. Dorf,2001). The closed-loop transfer function, *T(s)* of the system is used to determine the coefficients *(controller gains, poles and zeros)*, that yield the optimal deadbeat response, where *T(s)* is compared with standard , corresponding, order and normalized transfer function . Deadbeat response is illustrated in Figure . 1.

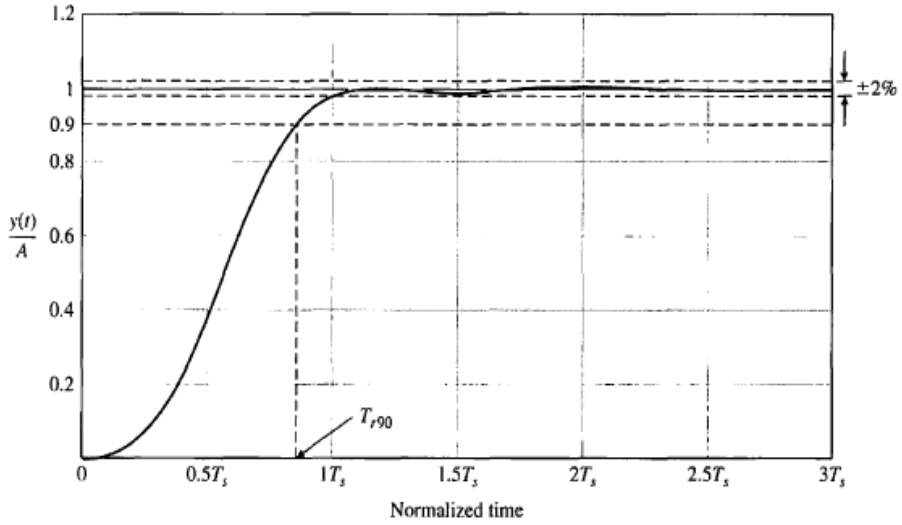

Figure 1 Deadbeat response and performance specifications (Richard C. Dorf,2001).

2. System modeling

The control of mechatronics system's motion is simplified to electric machine motion control that may or not include gear system, electric machine is powered and desired output movements will rely on how the electric motor is commanded, by using a simple controller (e.g. PIC microcontroller), and corresponding feedback element and, interfaces, the output movements (the rotation to a fixed speed or angle) can be controlled easily. Motion control system and components -A negative closed loop feedback control system with forward controller and corresponding simulink model shown in Figure 2(a)(b) are to be used**.**

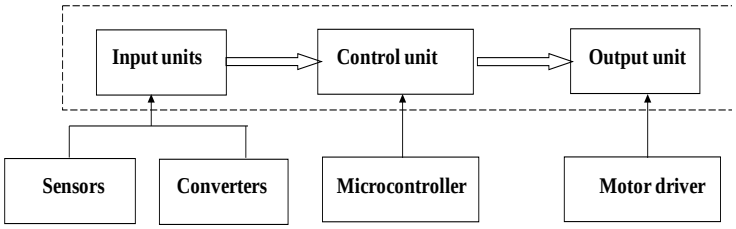

Figure 2 (a) Motion control system and components

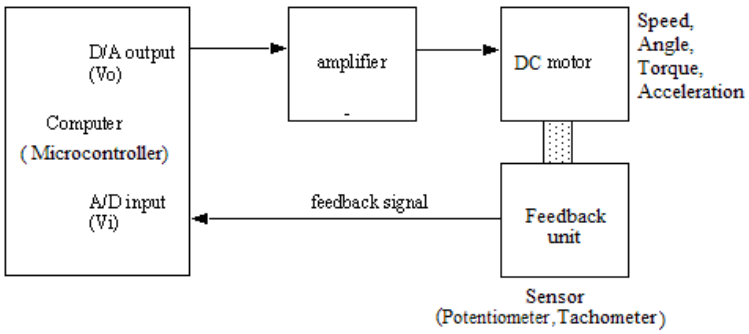

Figure 2 (b) motion control using microcontroller, and corresponding feedback element

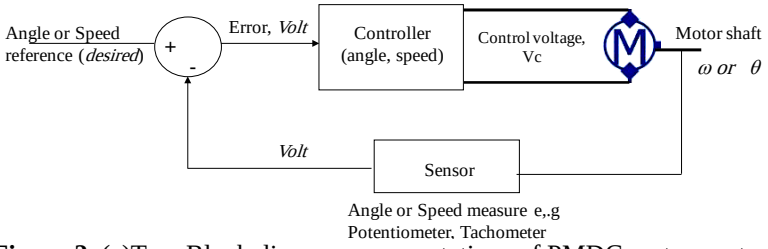

Figure 2 (c)Two Block diagram representations of PMDC motor control

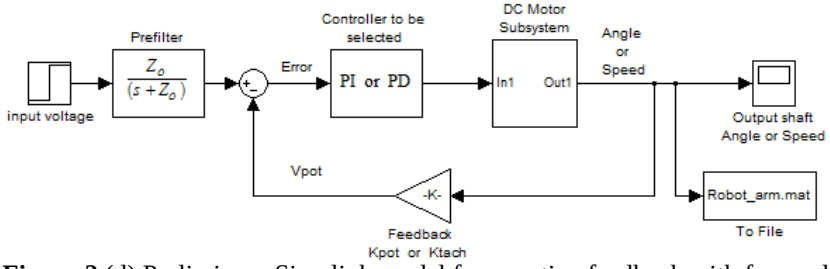

Figure 2 (d) Preliminary Simulink model for negative feedback with forward compensation and prefilter

2.I Electric machine modeling

The actuating machines most used in mechatronics motion control systems are DC machines. The PMDC motor is an example of electromechanical systems with electrical and mechanical components. In many sources a derivation of DC motor mathematical model in different

forms in terms of different relations are introduced including (Richard C. Dorf,2001)(Thomas R. Kurfess, 2005)(Mohsen Shahinpoor, 1987))(Farhan A. Salem, 2013)(Farhan A. Salem, 2013)(Farhan A. Salem, 2013)(Grzegorz Sieklucki, 2012) .Based on the Newton's law combined with the Kirchoff's law, the mathematical model of electric motor can be derived. The PMDC motor open loop transfer function without *any* load attached relating the input voltage, $V_{in}(s)$, to the motor *shaft* output angle, $\theta_m(s)$, and given by Eq. (1) and, the PMDC motor open loop transfer function relating the input voltage, $V_{in}(s)$, to the motor shaft output angular velocity, $\omega_m(s)$, given by Eq. (2):

$$G_{angle}(s) = \frac{\theta(s)}{V_{in}(s)} = \frac{K_t}{\left\{\left[\left(L_a J_m s^3 + (R_a J_m + b_m L_a)s^2 + (R_a b_m + K_t K_b)s\right]\right\}} \quad (1)$$

$$G_{speed}(s) = \frac{\omega(s)}{V_{in}(s)} = \frac{K_t}{\left\{\left[L_a J_m s^2 + (R_a J_m + b_m L_a)s + (R_a b_m + K_t K_b)\right]\right\}} \quad (2)$$

Here note that the transfer function $G_{angle}(s)$ can be expressed as: $G_{angle}(s) = G_{speed}(s) * (1/s)$. This can be obtained using MATLAB, by the following, code: >> G_angle = tf(1,[1,0])* G_speed, Where: running tf(1,[1,0]), will return (1/s). The geometry of the mechanical part determines the moment of inertia and damping, the mobile platform can be considered to be of the cuboid or cubic shape. Also, arm is considered as a rod of mass *m*, length ℓ, (so that $m = \rho * \ell * s$), this rod is rotating around the axis which passes through its center and is perpendicular to the rod , correspondingly , the total equivalent inertia, J_{equiv} and total equivalent damping, b_{equiv} at the armature of the motor with gears attaches, can be calculated from known formulae. for robotic arm application, the moment of inertia can be found by computing the following integral:

$$\int_{-l/2}^{l/2} \rho x^2 s dx = \rho s \frac{x^3}{3} \Big\|_{-l/2}^{l/2} = \frac{m}{sl} s 2 \frac{l^3/8}{3} = \frac{1}{12} m l^2$$

Also, the mobile robotic platform can be considered to be of the cuboids shape, with the inertia as calculated by:

$$J_{load} = \frac{bh^3}{12}$$

Where: *h* is platform width, h platform height. The total equivalent inertia, J_{equiv} and total equivalent damping, b_{equiv} at the armature of the motor with gears attached, equations for J_{equiv} and, b_{equiv} are given by:

$$b_{equiv} = b_m + b_{Load}\left(\frac{N_1}{N_2}\right)^2 \quad \Leftrightarrow \quad J_{equiv} = J_m + J_{Load}\left(\frac{N_1}{N_2}\right)^2 \qquad (3)$$

$$J_{mobile} = \frac{bh^3}{12} \quad \Leftrightarrow \quad J_{arm} = \frac{1}{12} m l^2$$

The total equivalent inertia, J_{equiv} and total equivalent damping, b_{equiv} at the armature of the motor are ,J_{equiv} =0.2752 kg.m^2 , b_{equiv} = 0.3922 N.m.s. and for used robot arm ,calculating and substituting values, gives:

J_{Load}= (8*(0.4) ^2)/12 = 0.1067 kg.m^2

Therefore, J_{equiv} ,to be :

$J_{equiv} = J_m + J_{load} *(1/1)$

J_{equiv} =0.02+0.107= 0.1267 kg·m²

Obtaining the total damping, b_{total} , gives:

$b_{equiv} = b_m + b_{load}(1/1)$

$\mathbf{b}_{equiv\_arm}$ = 0.03 + 0.09 = 0.12 N.sec/m

2.2 Modeling of system dynamics

When deriving an accurate mathematical model for motion system it is important to study and analyze dynamics between system and surroundings and considering all the forces applied upon the system. Depending on system applications parameters, shape and dimensions, dynamic model can be derived, in the suggested model the load torque inertia and damping characteristics are to be defined , for specific applications, for mobile robotic applications. In (Farhan A. Salem, 2013)(Farhan A. Salem, 2013)(Farhan A. Salem, 2013)(Grzegorz Sieklucki, 2012), an accurate derivation of all forces acting on mobile platform system, when it is running are introduced. For mobile robotic platform, considering dimensions, and for simplicity, the following most acting forces and corresponding torques, can be considered:

Rolling resistance force, $\mathbf{F_{rolling}}$: depending on the mobile platform speed and it is proportional to the vehicle platform, and is given by:

$$F_{rolling} = F_{normal\_force} * C_r = M * g * C_r * \cos(\alpha) \quad (4)$$

In terms of the vehicle linear speed Eq.(4) becomes:

$$F_{rolling} = M * g * (C_{r0} - C_{r1} * \upsilon) * sign(\upsilon)$$

The rolling resistance torque is given by:

$$T_{rolling} = (M * g * C_r * \cos(\alpha)) * r_{wheel}$$

Where : M : The mass of the SMEV and cargo (*Kg*). g .C_r The rolling resistance coefficients is calculated by the following expression:

$$C_r = 0.01\left(1 + \frac{3.6}{100}\upsilon_{robot}\right)$$

The hill-climbing resistance force F_{climb}; while robot platform is moving up or down a hill, the weight of the robot will create a hill-climbing resistance force directed downward, this force will oppose or contribute to the motion, the component of gravity in the dimension of travel is the hill-climbing resistance force and is given by:

$$F_{climb} = M * g * \sin(\alpha) \quad (5)$$

Where: M : The mass of the system and cargo (*Kg*). *g:* The gravity acceleration (m/s^2). α:Road or the hill climbing angle, *road slope* (Rad.). If we assume the mobile platform system is on a level surface, this force is zero, $= 0\ \alpha, sin(0)=0$

The hill-climbing resistance, *slope*, torque, is given by:

$$F_{climb} = F_{slope} = \left(M * g * \sin(\alpha)\right) * r_{wheel} \qquad (6)$$

The total **inertia force** of the mobile platform,

$$F_{inertia} = F_{slope} = M\frac{d\upsilon}{dt}$$

The inertia torque is given by:

$$F_{inertia} = F_{slope} = r^2 M\frac{d\upsilon}{dt}$$

Aerodynamic Drag force , F_{aerod}: the force opposing the motion of the vehicle due to air drag, wind resistance, consists mainly of two components; shape drag and skin friction. the aerodynamic drag force is function of mobile platform linear velocity, *v* and given by:

$$F_{aerod} = 0.5 * \rho * A * C_d * \upsilon_{vehicl}^2 \qquad (7)$$

Considering platform and wind speed Eq.(7) become:

$$F_a = 0.5\rho A C_d * \left(\upsilon_{robot} + \upsilon_{wind}\right)^2 sign(\upsilon_{robot})$$

$$F_{aerod} = 0.5\rho A C_d \left(\upsilon_{vehicl} + \upsilon_{wind}\right)^2 sign\left(\upsilon_{vehicl} + \upsilon_{wind}\right)$$

The aerodynamics torque is given by:

$$T_{aerod} = \left(\frac{1}{2} * \rho * A * C_d * \upsilon_{vehicle}^{\ 2}\right) * r_r$$

Where: C_d : Aerodynamic drag coefficient characterizing the shape of the mobile platform

The angular acceleration force F_{acc_angle} , is the force required by the wheels to make angular acceleration and is given by:

$$F_{acc\_angle} = J\frac{G^2}{r_{wheel}^2}\alpha$$

The angular acceleration torque is given by:

$$T_{acc\_angle} = r_{wheel} * J\frac{G^2}{r_{wheel}^2}\alpha = J\frac{G^2}{r_{wheel}}\alpha \qquad (8)$$

3. Controller design for desired deadbeat response.

The control of mechatronics system's motion is simplified to electric machines motion control that may or not include gear system, electric machine is powered and desired output movements will rely on how the electric motor is commanded. The most basic design requirements of a given electric DC machine are to rotate at desired angular speed $\omega = d\theta/dt$ and/or

to achieve desired angular position, θ , at the minimum possible steady-state error e_{ss} , also the machine must accelerate to its steady-state speed, $\alpha=d^2\theta/dt^2$ as soon as it turns on, this means it is desirable to have a minimum suitable settling time , T_s that will not damage the equipment (e.g. T_s in less than 2 sec), and the minimum suitable overshoot, *Mp* (e.g. *Mp* less than 5%)**,** such suitable response can be achieved applying design for deadbeat response.

3.1 Design for deadbeat response

To be able to control a motion process in DC motor system, the precise output of system (speed or position) needs to be measurable. Feedback comparison of the desired and actual output is then a natural step in implementing a motion control system. This comparison generates an error signal that may be used to correct the system motion, thus yielding repeatable and accurate results. The applied controller's gains, poles and zeros are responsible for achieving the optimal desired system response, e.g. deadbeat response. In controller design for deadbeat response approach, the controller's coefficients,*(controller gains, poles and zeros)*, depend on the physical parameters of the system. To determine the optimal coefficients, that yield the optimal deadbeat response, the system's overall equivalent closed-loop transfer function, *T(s)* is compared with standard, of corresponding order, and normalized transfer function, (particularly the characteristic equations are compared). In this approach, to determine the coefficients, that yield the optimal deadbeat response, the *standard* transfer function is normalized; and is to be used as the system's overall and desired closed-loop transfer function. The coefficients of the normalized standard transfer function are then assigned the values necessary to meet deadbeat response requirements. These coefficients were selected to achieve deadbeat response and minimize settling time T_s and rise time, T_R to 100% of the desired command; the coefficients of the *normalized* standard transfer function are recorded in Table 1(Richard C. Dorf,2001). For example; The desired Standard *third* order closed-loop transfer function for achieving desired deadbeat response specifications can be rewritten to have the form:

$$\mathrm{T(s)} = \frac{\omega_n^3}{s^3 + \alpha\omega_n s^2 + \beta\omega_n^2 s + \omega_n^3} \qquad (9)$$

Dividing the numerator and denominator by ω_n^3, and letting $\bar{s} = s / \omega_n$, substituting, gives:

$$\mathrm{T(s)} = \frac{\omega_n^3 / \omega_n^3}{\frac{s^3}{\omega_n^3} + \frac{\alpha\omega_n s^2}{\omega_n^3} + \frac{\beta\omega_n^2 s}{\omega_n^3} + \frac{\omega_n^3}{\omega_n^3}} = \frac{1}{\bar{s}^3 + \alpha\bar{s}^2 + \beta\bar{s} + 1}$$

For a higher-order system, the same method is used to derive the normalized equation (Richard C. Dorf,2001) . The desired Standard *second* order closed-loop transfer function for achieving desired deadbeat response specifications is given by:

$$T(s) = \frac{\omega_n^2}{s^2 + \alpha\omega_n s + \omega_n^2} \qquad (10)$$

Lets assume that , it is required, for a given third, order plant, to design a control system to have deadbeat characteristics response to step input and meeting desired settling time of 0.5 sec., the design can be accomplished as follows; From table 1, substituting coefficients values, for third order system, that is α=1.9 , β =2.20, gives:

$$T(s) = \frac{\omega_n^3}{s^3 + 1.9\omega_n s^2 + 2.20\omega_n^2 s + \omega_n^3}$$

Then, we calculate and chose undamped natural frequency, ω_n , based on the desired settling time or rise time: for desired settling time of *0.5* second, the normalized settling time (from table 1) is given by:

$$\omega_n T_s = 4.04 \Rightarrow \omega_n = \frac{4.04}{T_s} = \frac{4.04}{0.5} = 8.08 \approx 8.0 \text{rad / sec}$$

Now, the complete closed-loop transfer function with all variables (coefficients defined) is known, and given by:

$$T(s) = \frac{512}{s^3 + 15.2s^2 + 121.6s + 512} \qquad (11)$$

Also, from table 1, the step response of this system has a deadbeat response with an overshoot of 1.65% , undershoot of 1.36% and a settling time of 0.5 second. Finally, to find the applied appropriate controller and its optimal coefficients; gains, poles and zeros, we compare transfer function given by Eq (15) with system's overall equivalent closed loop transfer function. The following nominal values for the various parameters of two different eclectic motor are to be used and tested:

First motor: V_{in}=12 Volts; J_m=0.02 kg·m²; b_m =0.03;K_t =0.023 N-m/A; K_b =0.023 V-s/rad; R_a =1 Ohm; L_a=0.23 Henry; T_{Load}. *Second motor*: V_{in}=12 Volts; Motor torque constant, Kt = 1.1882 Nm/A; Armature Resistance, R_a = 0.1557 Ohms (Ω) ; Armature Inductance, L_a = 0.82 MH ;Geared-Motor Inertia: J_m = 0.271 kg.m^2, Geared-Motor Viscous damping b_m = 0.271 N.m.s; Motor back EMF constant, K_b = 1.185 rad/s/V.

To test and verify all of the proposed design, calculation, built in function and simulink model of actuating machine, the design will be applied to two different mechatronics motion control applications; mobile robot and single joint robot arm, to achieve desired deadbeat response with desired settling time, the disturbance in the form of load torque is to be defined and calculated and defined. The *mobile* platform, has the following nominal

dimensions and parameters values; wheel radius, *r =0.075 m*, platform height, h= 0.920 m, platform width, b = 0.580 m, the distance between wheels centers = 0.4 m. The desired suitable linear output speed, for domestic, mobile robot, is to move with 0.5 meter per second, that is ω=V/r = 0.5/ 0.075 = 6.6667 rad/s,. Tachometer constant, *and the corresponding sensor, tachometer,* for maximum applied input voltage of V=12, K_{tac} = 12 / 6.6667=1.8 *(rad/sec*). The *robot arm* system to be designed, has the following nominal values; arm mass, M= 8 Kg, arm length, L=0.4 m, and viscous damping constant, b = 0.09 N.sec/m. so that a voltage range of 0 to 12 volts corresponds linearly of an Robot arm output angle range of 0 to 180, that is to move the robot arm to the desired output angular position, θ_L , corresponding to the applied input voltage ,V_{in} , for simplicity, gear ratio, n=1,10.

Table 1 The coefficients of the normalized standard transfer function

System order	Optimal coefficients					Percent Overshoot	Percent Undershoot	Rise,90%	Rise,100%	Settling
	α	β	γ	δ	ε	OS%	PU%	T_R	T_R	T_S
2nd	1.82					0.10%	0.00%	3.47	6.58	4.82
3nd	1.90	2.20				1.65%	1.36%	3.48	4.32	4.04
4nd	2.20	3.50	2.80			0.89%	0.95%	4.16	5.29	4.81
5nd	2.70	4.90	5.40	3.40		1.29%	0.37%	4.84	5.73	5.43
6nd	3.15	6.50	7.55	7.55	4.05	1.63%	0.94%	5.49	6.31	6.04

3.2 Proportional -Integral (PI) controller

PI controller is widely used in variable speed applications and current regulation of electric motors, because of its simplicity and ease of design. PI controller transfer function is given by:

$$G_{PI}(s) = K_P + \frac{K_I}{s} = \frac{(K_P s + K_I)}{s} = \frac{K_P\left(s + \frac{K_I}{K_P}\right)}{s} = \frac{K_P(s + Z_o)}{s} \qquad (12)$$

Where, Zo: Zero of the PI-controller, K_P: The proportional gain, K_{PI}: The proportional coefficient; T_I: time constant. This transfer function, shows that, PI controller represents a pole located at the origin and a stable zero placed near the pole, at Z_o=- K_I/ K_P, resulting in drastically eliminating steady state error due to the fact that the feedback control system type is increased by one. The PI pole and zero will affect the response, mainly the PI zero, Z_o=- K_I/ K_P, will significantly and inversely effect the response and should be cancelled by prefilter given by Eq. (15).

3.3 Proportional -Derivative - PD controller:

The transfer function of PD-controller is given by :

$$G_{PD}(s) = K_P + K_D s$$

Rearranging, we have the following form:

$$G_{PD}(s) = K_P + K_D s = K_D(s + \frac{K_P}{K_D}) = K_D(s + Z_{PD}) \qquad (13)$$

The PD-controller is equivalent to the addition of *a simple zero* at: $Z_{PD} = K_P / K_D$.The addition of zeros to the open-loop has the effect of pulling the root locus to the left, or farther from the imaginary axis, resulting in improving the transient response

3.4 Systems design with prefilter

Prefilter is defined as a transfer function $G_p(s)$ that filters the input signal *R(s) prior* to calculating the error signal. Adding a control system to plant, will result in the addition of poles and/or zeros, that will effect the response, mainly the added zero, will significantly inversely effect the response and should be cancelled by prefilter, therefore the required prefilter transfer function to cancel the zero is given by Eq.(14). In general. The prefilter is added for systems with *lead* networks or *PI* compensators. A prefilter for a system with a lag network, mainly, is not , since we expect the effect of the zero to be insignificant.

3.5 PI controller with deadbeat response design:

With PI controller with deadbeat response design, the order of overall closed-loop transfer function, T(s), will be increased by one and contain a new added PI-controller zero at $Z_{PI}=K_I/K_P$. this zero will significantly and inversely affect the response of the closed-loop system, T(s), and should be eliminated while maintaining the proportional gain ,K_P, of the closed-loop system, this can be achieved by a prefilter. Therefore the required prefilter transfer function to cancel the zero is given by Eq.(15):

3.6 PD controller with deadbeat response design:

For systems *with PD* compensators, a prefilter is used to eliminate any undesired effects of the term *s + z* introduced in the closed-loop transfer function, the required prefilter transfer function is given by Eq.(16):

$$G_{Prefilter}(s) = \frac{Z_0}{(s + Z_0)} \qquad (14)$$

$$G_{PI\_Prefilter}(s) = \frac{Z_{PI}}{(s + Z_{PI})} \qquad (15)$$

$$G_{PD\_Prefilter}(s) = \frac{Z_{PD}}{(s + Z_{PD})} \qquad (16)$$

3.7 Position and velocity feedback sensors modeling; Potentiometer and Tachometer modeling.

To calculate the error, we need to convert the actual output (arm position, or mobile speed) into voltage, V, then compare this voltage with the input voltage V_{in}, the difference is the error signal in volts.

Potentiometer is a sensor used to measure the actual output position, θ_L ,convert into corresponding volt, V_p and then feeding back this value, the Potentiometer output is proportional to the actual position, θ_L, this can be accomplished as follows: the output voltage of potentiometer is given by:

$V_p = \theta_L * K_{pot}$

Where: θ_L :The actual position. K_{pot} the potentiometer constant; it is equal to the ratio of the voltage change to the corresponding angle change, depending on maximum desired output angle, the potentiometer can be chosen. For our case, input volt range V_{in}*= 0:12,* and output angle range θ*= 0:180* degrees, substituting, we have:

$$K_{pot} = \frac{\text{(Voltage change)}}{\text{(Degree change)}} = \frac{(12 - 0)}{(180 - 0)} = 0.0667 \text{ V / degree}$$

Potentiometer constant K_{pot} = 0.0667 V/degree. This value (0.0667), means that each one input volt corresponds to *180/12= 15* output angle in radians , to obtain a desired output angular position of 180 , we need to apply 12 volts, to obtain an angular position of 90 we need to apply (*90*0.0667= 6.0030* Volts).

Tachometer is most used sensor to measure the actual output angular speed, ω_L .. Dynamics of tachometer can be represented using the following equation:

$V_{out}(t) = K_{tac} * d\,\theta(t)/dt = V_{out}(t) = K_{tac} * \omega$

The transfer function of the tachometer is given by:

$V_{out}(s) / \omega(s) = K_{tac}$

A suitable linear output speed of e.g. of domestic mobile robot is to move with 0.5 m/s, that is:

$$\omega = \frac{V}{r} = \frac{0.5}{0.075} = 6.6667 \text{ rad / s},$$

Tachometer constant, K_{tac} = 12 / 6.6667=1.8

3.8 Design for PD controller with desired deadbeat response specifications in terms of desired output angular position.

Considering that system dynamics and disturbance torques depends on application shape and dimensions (robot arm, conveyer), the mechanical DC motor part, will have the form:

$K_t * i_a = T_\alpha + T_\omega + T_{load} + T_f$

The coulomb friction can be found at steady state,*to be:*

$K_t * i_a - b*\omega = T_f$

In the following calculation the disturbance torque, T, is all torques including coulomb friction, and given by ($T=T_{load}+T_f$) . Applying PD controller with deadbeat response design for output desired output angular position, The open-loop transfer function of the PMDC, is given by:

$$G_{open}(s)=\frac{K_t}{(L_a s+R_a)(J_m s^2+b_m s)+(L_a s+R_a)(T)+K_b K_t}$$

Manipulating for *forward and closed* loop transfer functions, including disturbance torques, sensor and gear ratio, gives:

$$G_{open}(s)=\frac{K_t}{(L_a s+R_a)(J_m s^2+b_m s)+(L_a s+R_a)(T)+K_b K_t}$$

$$G_{open}(s)=\frac{nK_{pot}K_t}{L_a J_m s^3+(R_a J_m+b_m L_a)s^2+(R_a b_m+L_a T)s+(R_a T+K_b K_t)}$$

$$G_{open}(s)=\frac{nK_{pot}K_t}{TJ_m s^3+\{(T_a)R_a+(T\}s^2+(R_a T+K_b K_t)T}$$

$$G_{f\ orward}(s)=\frac{nK_{pot}K_t K_D s+nK_{pot}K_t K_P}{L_a J_m s^3+(R_a J_m+b_m L_a)s^2+(R_a b_m+L_a T)s+(R_a T+K_b K_t)}$$

$$G_{closed}(s)=\frac{nK_{pot}K_t K_D s+nK_{pot}K_t K_P}{L_a J_m s^3+(R_a J_m+b_m L_a)s^2+(R_a b_m+L_a T+nK_{pot}K_t K_D)s+R_a T+K_b K_t+nK_{pot}K_t K_P}$$

$$G_{closed}(s)=\frac{\frac{nK_{pot}K_t K_D s+nK_{pot}K_t K_P}{L_a J_m}}{s^3+\frac{(R_a J_m+b_m L_a)}{L_a J_m}s^2+\frac{(R_a b_m+L_a T+nK_{pot}K_t K_D)}{L_a J_m}s+\frac{(R_a T+K_b K_t+nK_{pot}K_t K_P)}{L_a J_m}}$$

Comparing with standard normalized third order system and manipulating, for K_D and K_P, given by Eq.(9)

$$T(s)=\frac{\omega_n^3}{s^3+\alpha\omega_n s^2+\beta\omega_n^2 s+\omega_n^3}$$

$$\beta\omega_n^2=\frac{(R_a b_m+L_a T+nK_{pot}K_t K_D)}{L_a J_m}\Leftrightarrow K_D=\frac{\beta\omega_n^2 L_a J_m-(R_a b_m+L_a T)}{nK_{pot}K_t}$$

$$\omega_n^3=\frac{(R_a T+K_b K_t+nK_{pot}K_t K_P)}{L_a J_m}\Leftrightarrow K_P=\frac{\omega_n^3 L_a J_m-(R_a T+K_b K_t)}{nK_{pot}K_t}$$

For design consideration, and based on required design accuracy we can use the simplified second order model, assuming La =0, and manipulating for closed loop with PD controller, gives:

$$G_{angle}(s)=\frac{\theta(s)}{V_{in}(s)}=\frac{K_t}{\left\{\left[L_aJ_ms^3+(R_aJ_m+b_mL_a)s^2+(R_ab_m+K_tK_b)s\right]\right\}}\Leftrightarrow G_{angle}(s)=\frac{\frac{K_t}{R_aJ_a}}{s\left[s+\frac{1}{J_m}\left(b_m+\frac{K_tK_b}{R_a}\right)\right]}$$

$$G_{angle}(s)=\frac{\theta(s)}{V_{in}(s)}=\frac{nK_t}{s\left[(R_aJ_m)s^2+(R_ab_m+K_tK_b)\right]}$$

$$G_{forward}(s)=\frac{\theta(s)}{V_{in}(s)}=\frac{K_tK_P+K_tK_Ds}{s\left[(R_aJ_m)s+(R_ab_m+K_tK_b)\right]}$$

$$G_{closed}(s)=\frac{\theta(s)}{V_{in}(s)}=\frac{nK_tK_P+nK_tK_Ds}{s\left[(R_aJ_{equiv})s+(R_ab_{equiv}+K_tK_b+nK_{pot}K_tK_D)\right]+nK_{pot}K_tK_P}$$

Comparing with standard normalized second order system and manipulating, for K_D and K_P, given by Eq.(10)

$$G_{closed}(s)=\frac{\frac{nK_tK_P+nK_tK_Ds}{(R_aJ_{equiv})}}{s^2+\left(\frac{R_ab_{equiv}+K_tK_b+nK_{pot}K_tK_D}{R_aJ_{equiv}}\right)s+\frac{nK_{pot}K_tK_P}{(R_aJ_{equiv})}}=\frac{\omega_n^2}{s^2+\alpha\omega_ns+\omega_n^2}$$

$$\alpha\omega_n=\frac{R_ab_{equiv}+K_tK_b+nK_{pot}K_tK_D}{R_aJ_{equiv}}\Leftrightarrow K_D=\frac{\alpha\omega_nR_aJ_{equiv}-(R_ab_{equiv}+K_tK_b)}{nK_{pot}K_t}$$

$$\omega_n^2=\frac{nK_{pot}K_tK_P}{(R_aJ_{equiv})}\Leftrightarrow K_P=\frac{R_aJ_{equiv}\omega_n^2}{nK_{pot}K_t}$$

$$Z_{PD}=\frac{K_P}{K_D}$$

From Table 1, substituting coefficients values, for second order system, that is $\alpha=1.82$, $Ts=\omega_n/4.82$, the prefilter transfer function is obtained

3.9 Design for PI controller with desired deadbeat response specifications in terms of desired output linear speed.

Based on equations that describes DC motor, system dynamics and sensor modeling, the open loop transfer function, relating the armature input terminal voltage, $V_{in}(s)$ to the output terminal voltage of the tachometer $V_{tach}(s)$, with load and corresponding torques applied and considered, will be given by Eq.(17) :

$$G_{open}(s)=\frac{2K_{tach}*K_t}{2b_{equiv}L_as^2+r^2ML_as+2b_{equiv}R_as+r^2MR_as+C_rL_as+2J_{equiv}L_as+2K_bK_t+C_rR_a+2J_{equiv}R_a} \quad (17)$$

Applying PI controller with deadbeat response design, for desired output speed , the system's closed-loop transfer function, T(s), is given by Eq.(18) :

$$G_{closed}(s)=\frac{\dfrac{2K_{tach}*K_tK_P}{\left(2b_{equiv}+r^2M\right)L_a}(s+Z_{PI})}{s^3+\left[\dfrac{R_a}{L_a}+\dfrac{\left(C_r+2J_{equiv}\right)}{\left(2b_{equiv}+r^2M\right)}\right]s^2+\dfrac{\left[\left(C_r+2J_{equiv}\right)R_a+2K_t\left(K_b+K_{tach}K_P\right)\right]}{\left(2b_{equiv}+r^2M\right)L_a}s+\dfrac{2K_{tach}K_tK_I}{\left(2b_{equiv}+r^2M\right)L_a}} \quad (18)$$

The prefilter transfer function is given by Eq.(15). Applying deadbeat design approach to determine the optimal coefficients,*(controller gains, poles and zeros)*, that yield the optimal deadbeat response, by comparing Eq. (18), with Eq(10) gives:

$$K_P=\frac{\left[\beta\omega_n^2L_a\left(2b_{equiv}+r^2M\right)\right]-\left[\left(C_r+2J_{equiv}\right)R_a+2K_tK_b\right]}{2K_tK_{tach}}$$

$$K_I=\frac{\omega_n^3L_a\left(2b_{equiv}+r^2M\right)}{2K_tK_{tach}}$$

$$Z_{PI}=\frac{K_I}{K_P}=\frac{\omega_n^3L_a\left(2b_{equiv}+r^2M\right)}{\left[\beta\omega_n^2L_a\left(2b_{equiv}+r^2M\right)\right]-\left[\left(C_r+2J_{equiv}\right)R_a+2K_tK_b\right]}$$

From Table 1, the required coefficients are: $\alpha = 1.9$, $\beta = 2.2$ and $\omega_n T_s = 4.04$, $T_s = 4.04/\omega_n$

4. Simulation

The simulink model of DC motor system for desired deadbeat response using PI or PD, is shown in Figure .3, in presented model three most suitable controllers for DC motor output control are applied; PID, PI and PD, where three forms of simulink PID model are used, as well as separate PI and PD models are used , two prefilter blocks are added with PI zero and with PD zero, also disturbance torques are included. This model can be switched from one controller to another, from filter to another, from controlling output speed or position and from input type to another, by the use of manual switches. The presented model is also supported with visual readings of steady state final values. This model can be used as separate tool, as well as supporting model for design for deadbeat response testing and verification. To use this model, load of attached system (application) and used DC motor parameters, must be first defined in MATLAB, directly or using MATLAB m.file, also using calculated optimal values that yields the desired deadbeat response, obtained using the new built-in function. Running this model will return angular and linear speed/time, angular and linear position/time, torque/time, current/time , control signal/time response curves also visual readings of steady state final values, after running the simulink model for a given motor parameters and desired deadbeat response , the next code can be used to plot all response plots:

```
load PMDC1, load PMDC2,load PMDC3,load PMDC4,load PMDC5,load
PMDC6
subplot(3,2,1); plot(DC_angle), ylabel('\Theta ,
Rad'),xlabel('  '), title('Angular position Rad '), grid,
subplot(3,2,2); plot(Angualr_speed), ylabel('Angular speed,
\omega Rad/s '),xlabel('    '), title('Angular speed Rad/Time,
'), grid,
subplot(3,2,4); plot(torque), ylabel('Motor Torque,
Nm'),xlabel('  '), title('Motor Torque Nm/Time '), grid,
subplot(3,2,3); plot(DC_linear_speed), ylabel(' Angle linear
speed. M/s'),xlabel('    '), title(' linear speed M/s '),
grid,
subplot(3,2,6);plot(Current), ylabel('Current,  Amp
'),xlabel(' Time(sec)'), title('Current Amp/s,  '), grid,
subplot(3,2,5); plot(controller_signal), ylabel('
controller_action'),xlabel('Time(sec)'), title('controller
signal (mA)'), grid,
```

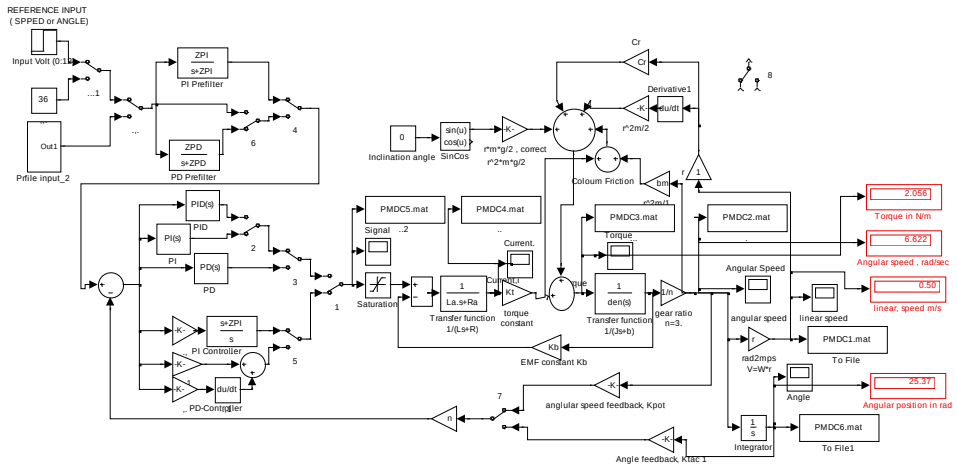

Figure 3 Simulink model for motion control design for desired deadbeat response specifications.

5. MATLAB built-in function script

The equations derived, was used to design a new MATLAB built-in function with specific purpose, that is design and verification for desired deadbeat response specification, particularly for desired settling time. The presented built-in function is named deadbeat(a), the input argument *a*, can have the value of 1 or 2; where 1 for electric motor system output speed control with PI-controller design for deadbeat response with desired settling time. The input argument 2 for electric motor system output angle control with PD-controller design for deadbeat response with desired settling time. Using this new built-in function user is to define the used DC motor parameters, gear ratio, application system shape, dimensions, and corresponding torque disturbance, by running the function it will return and plot each of the following; the open loop motor transfer in terms of output

speed and position, the closed loop transfer in terms of selected output speed or position for specific application, corresponding response deadbeat response curves, determines the coefficients (gains pole and zero) that yield the optimal deadbeat response for desired output speed or angle control, as well as performance specifications in terms of Mp, Ess, Ts, final output value. The built-in function script is written below:

The coefficients (gains pole and zero), from running built in function is to be used to be defined in MATLAB workspace to run the simulink model for design verification.

```
function     deadbeat(a)
% DEADBEAT    Determines the coefficients that yield the optimal
%             deadbeat response for desired output speed or angle
%             control of a given PMDC motor with gears attached.
%             Also calculate transfer functions ( open and closed),
%             of motor used, controller, as well as mobile robot
%             system, Prefilter.
%              Also plots system and corresponding deadbeat response
                plots.
% deadbeat(1)  used for output speed control ( e.g. MOBILE robot,
%             system and controller design) Determines the
              Coefficients that yield the optimal proportional-
               Integral (PI) controller with P deadbeat response,
for
              a given MOBILE robot system and plots deadbeat
response.
% deadbeat(2) used for output position and controller design for
%               given PMDC with gears e.g. for single joint robotic
arm
%             system and controller design Determines the
%             coefficients that yield the optimal Proportional- with
%             deadbeat response, Derivative (PD) controller for a
              given PMDC motor system and plots corresponding
              deadbeat response.
clc, close all, format short
disp( '  ')
disp( ' ==================================================')
disp('                  Please define PMDC parameters     ')
disp( ' ==================================================')
Jm = input('  Enter  Motor armature moment of inertia (Jm) :');
bm = input('  Enter  damping constant of the motor system (bm):');
Kb = input('  Enter  ElectroMotive Force constant (Kb):');
Kt = input('  Enter  Torque constant (Kt):');
Ra = input('  Enter electric resistance of the motor armature ,ohms,
(Ra):');
La =input('  Enter  electric inductance  of the motor armature
,Henry,(La) :');
n =input('   Enter  gear ratio ,(n) :');
disp('    ')
    disp( ' ==================================================')
num1 = 1;
den1= [La ,Ra];
num2 = 1;
den2= [Jm ,bm];
A = conv( [La ,Ra], [Jm ,bm]);
```

```
TF1 =tf(Kt*n, A);
% Obtaining open and closed loop transfer functions of DC motor
system  and step response
disp('DC motor open loop transfer function in terms of:Speed/Volt:
')
Gv= feedback(TF1,Kt);
Gv1=zpk(Gv )
disp( ' ================================================')
disp('DC motor OPEN loop transfer function in terms of:Angle/Volt:
')
Ga=tf(1,[1,0] )*Gv;
Ga1=zpk(Ga )
Figure ure,subplot(2,1,1),step(V*Ga), title( '(ANGLE) Step response
of used DC motor open loop transfer function '), xlabel(' Time '),
ylabel(' DC motor output angle \theta ')
subplot(2,1,2),step(V*Gv), title( '(ANGULAR SPEED) Step response of
used DC motor open loop transfer function'), xlabel(' Time '),
ylabel(' DC motor output speed \omega ')
if a==1;
    home
    disp( '  ')
    disp( ' ================================================')
  disp('                    Design for PMDC output angular and
linear SPEED ')
    disp('          PI with desired deadbeat reponse, e.g. mobile
robot output speed design and control :')
    disp( ' ================================================')
    disp('                    Define plant parameters ( e.g. mobile
robot')
    disp( ' ================================================')
mobile_robot_height= input( ' Enter system's Height, in meters : '
);
mobile_robot_wedth= input( ' Enter system's  Width, in meters : ' );
b_load=input( ' Enter Load damping constant : ' );
J_load =input( ' Enter Load inertia reflected to the motor armature
shaft : ' );
wheel_radius= input( ' Enter wheel radius, in meters :  ' );
desired_linear_speed= input('Enter desired output linear speed:  '
);
m= input( ' Enter system's total mass :  ' );
friction= input( ' Enter rolling friction coefficient :  ');
V = input(' Enter applied input voltage Vin :');
inclination_angle=input('  Enter inclination angle, if exist :');
%  for example: Jtotal_mobile_robot =(mobile_robot_wedth*
(mobile_robot_height)^3)/12;
    Jtotal= Jm+ J_load/(n)^2;
    btotal = bm + b_load/(n)^2;
     desired_angular_speed= (linear_speed)/r;
    Ktach =Vin/ desired_angular_speed ;%tachometer constant
    disp(' ================================================')
disp( ' ------------------------------------------------------')
    Ts= input( ' Enter desired settling time in seconds, Ts :  ' );
    disp( ' ------------------------------------------------')
    % mobile robot open loop TF
    num_mobile =Kt*n;    den_mobile =[La*Jtotal
,(Ra*Jtotal+btotal*La),(Ra* btotal+Kt*Kb)];
mobile_open_tf=tf(num_mobile,den_mobile);
```

```
    alpha=1.9;bita=2.2;omega_n=4.04/Ts;
    Kp1 =(( bita*omega_n^2*La*(r^2*M + btotal))-
(Ra*(Cr+2*Jtotal)+Kt*Kb))/(2*Kt*Ktach);
    Kp=abs(Kp1);
    Ki= omega_n^3*La*(r^2*M+btotal)/((2*Kt*Ktach));
    ZPI=Ki/Kp; num_PI=Kp*[1 ZPI]; den_PI= [ 1  0];
PI_tf=tf(num_PI,den_PI); num_filter1=ZPI; den_filter1=[ 1   ZPI];
    G_fil1=tf(num_filter1,den_filter1); T_mobile_PI=series(PI_tf,
mobile_open_tf); T_mobile_PI_closed=feedback(T_mobile_PI, Ktach);
T_mobile_deadbeat_PI_closed_prefilter = series
(T_mobile_PI_closed,G_fil1 );
    cc=zpk(  T_mobile_deadbeat_PI_closed_prefilter);
Figure
ure,subplot(2,2,1),step(V*r*T_mobile_deadbeat_PI_closed_prefilter),
    ylabel('System linear speed  \nu '),xlabel(' Time  '),
title('Output linear speed, WITH prefilter'), grid
    %Figure ure,
subplot(2,2,3), step(V*r*T_mobile_PI_closed);
ylabel('System linear speed  \nu '),xlabel(' Time  '), title('Output
linear speed, WITHOUT prefilter'), grid
subplot(2,2,2),step(V*T_mobile_deadbeat_PI_closed_prefilter),
ylabel('System angular speed  \omega '),xlabel(' Time  '),
title('Output angular speed, WITH prefilter'), grid
subplot(2,2,4),step(V*r*Gv);ylabel('DC motor speed  \nu '),xlabel('
Time  '),title('DC motor open loop tf linear speed: ')
    steady=dcgain(V*r*T_mobile_deadbeat_PI_closed_prefilter);
    Ess= steady - desired_linear_speed;
    t=0:0.01:100; [ww,
yyy]=step(V*r*T_mobile_deadbeat_PI_closed_prefilter);MM=max(ww);Mp=
MM-desired_linear_speed;
    home
    disp( ' ------------------------------------------------------')
    disp( '               System Performance specifications :    ')
    disp( ' ------------------------------------------------------')
    %fprintf(' Kp =%g , Kd=%g ,ZPD=%g \n',Kp,Kd,ZPD);
    fprintf(' Final steady state output value =%g \n',steady);
    fprintf(' Steady state error   Ess=%g  \n',Ess);
    fprintf(' Percent overshoot,   Mp=%g  \n',Mp);
    fprintf(' Tachometer constant, Ktach =%g  \n',Ktach );
    fprintf(' Angular speed, Omega =%g  \n',desired_angular_speed );
disp( ' ======================================================')
    disp( ' Gains and zero of: PI controller &  prefilter :')
disp( ' ======================================================')
    %fprintf(' Kp =%g , Ki=%g ,ZPI=%g \n',Kp,Ki,ZPI);
    fprintf('   Kp  =%g  \n',Kp);
    fprintf('   Ki  =%g  \n',Ki);
    fprintf('   ZPI =%g  \n',ZPI);
    disp( ' ------------------------------------------------------')
    disp('  PI Controller transfer function ')
    WWWW=zpk(PI_tf)
    disp( ' ------------------------------------------------------')
    disp('  Prefilter transfer function ')
    G_filter=tf(num_filter1,den_filter1);
    WWW=zpk(G_fil1)
    disp( ' ------------------------------------------------------')
    disp(' Closed loop transfer function, PI WITHOUT prefilter ')
    WW=zpk(T_mobile_PI_closed )
```

```
    disp( ' ------------------------------------------------------')
    disp('     Closed loop transfer function, PI WITH  prefilter ')
    W=zpk( T_mobile_deadbeat_PI_closed_prefilter)
disp( ' ------------------------------------------------------')
elseif a==2
    home
    disp( '  ')
    disp( ' ======================================================')
    disp('   Designing for output angular displacement with deadbeat
reponse')
    disp('           e.g. Robotic arm angular displacement    :')
    disp( ' ------------------------------------------------------')
    disp( '  ')
M= input('  Enter Load  mass , M=');
Jequiv= input('  Enter Load  inertia , Jload=');
Length = input('  Enter  robot arm Length, L=');
b_load= input('  Enter Load damping factor , b_load =');
n= input( 'Enter gear ratio, n = ');
V = input('  Enter  applied input voltage, Vin :');
V_max = input('  Enter  maximum allowed voltage , V max :');
Angle_max = input( ' Enter desired maximum allowed output angle  :
');
T_load= input( ' Enter Load torque : ');
% for example fro robot arm:  Jequiv=((M* Length^2)/12)+Jm;
    Kpot=V_max /angle_max;  % pot constsnt
    bequiv= bm + b_load;
    Jequiv= Jm + J_load;
    T=T_load; %+ angular_speed
    num_arm_open = [Kt*n*(180/pi)];
    den_arm_open=[La*Jequiv  (Ra*Jequiv+La*bequiv)
(Ra*bequiv+Kt*Kb)   0];
    disp(' ');
    home
    disp( ' ======================================================')
    disp(' Open loop transfer function in terms of output Angular
Position, : ')
    Ga_arm_open=tf(num_arm_open,den_arm_open);
    Ga_arm_open1=tf(num_arm_open,den_arm_open)*(r);
    Q=zpk(Ga_arm_open1)
    disp( ' ------------------------------------------------')
  disp(' Closed loop transfer function in terms of output Angular
Position,: ')
    G_close_arm= feedback(Ga_arm_open,(Kpot/n));
    Q=zpk(G_close_arm)
disp( ' ------------------------------------------------')
disp(' Closed loop transfer function in terms of output Linear
Position, : ')
    G_close_arm_linear= feedback(Ga_arm_open1,(Kpot/n));
    Q=zpk(G_close_arm_linear)
    disp( ' ------------------------------------------------------')
    disp( ' Response curves plotted ')
    disp( ' ------------------------------------------------------')
    Ts= input( ' Enter desired settling time in seconds, Ts :  ' );
    disp( ' ------------------------------------------------------')
    alph=1.90 ,bita= 2.20;  omega_n=Ts/4.04
    Kd1=(bita*omega_n*La*Jequiv -(Ra*La + La*T))/(n*Kpot*Kt);
    Kp1= omega_n^3*La*Jequiv -(Ra*T+Kb*Kt)/(n*Kpot*Kt);
```

```
    Kp=abs(Kp1);
    Kd=abs(Kd1);
    ZPD=(Kp/Kd)/(1.510);
    ZPI=ZPD;
    G_PD=Kd*[ 1  ZPD];
    prefilter_num=[ZPD];
    prefilter_den=[1  ZPD];
    G_prefilter=tf(prefilter_num,prefilter_den);
    num_arm_PD =G_PD* [Kt*n];
den_arm_PD=[La*Jequiv  (Ra*Jequiv+La*bequiv) ,(Ra*bequiv+Kt*Kb)
0];
    G_arm_PD_tf= tf(num_arm_PD, den_arm_PD);
    G_arm_PD_dead_closed= feedback(G_arm_PD_tf, Kpot);
    G_arm_PD_dead_overall=series(G_prefilter, G_arm_PD_dead_closed);
    G_arm_PD_dead_overal2=series(G_prefilter,
G_arm_PD_dead_closed)*(r);
    steady=dcgain(V*G_arm_PD_dead_overall);
    Ess= angle_max-steady;
    home
disp( ' ============================================================')
    disp( ' PD controller & Deadbeat prefilter gains and zero are
:')
    %fprintf(' Kp =%g , Kd=%g ,ZPD=%g \n',Kp,Kd,ZPD);
    fprintf(' Kp  =%g \n',Kp);
    fprintf(' Kd  =%g  \n',Kd);
    fprintf(' ZPD =%g \n',ZPD);
[ ww, yy]=step(V*G_arm_PD_dead_overall);MM=max(ww);Mp=MM-angle_max;
disp( ' ============================================================')
    disp( '                 Performance specifications are :')
    disp( ' ------------------------------------------------------
')
    %fprintf(' Kp =%g , Kd=%g ,ZPD=%g \n',Kp,Kd,ZPD);
    fprintf(' Final steady state output value =%g \n',steady);
    fprintf(' Steady state error,     Ess =%g  \n',Ess);
    fprintf(' Percent overshoot,      Mp =%g  \n',Mp);
    fprintf(' Potentiometer constant, Kpot =%g  \n',Kpot );
    fprintf(' Desired output angle,   Theta =%g  \n',Angle_max);
disp( ' ============================================================')
disp( ' PD transfer function, G = Kd(s + Kp/Kd)= Kd(s + ZPD),is
given by :')
fprintf( ' \t\t\t\t')
fprintf( ' Gpd(s)=%g(s+%g)\n',Kd,ZPD)
disp( ' ============================================================')
disp( ' Prefilter transfer function :')
G_prefilter=tf(prefilter_num,prefilter_den);
QQ=zpk(G_prefilter)
disp( ' ============================================================')
disp(' Overall closed loop transfer function in terms of output
Angular displacement :')
G_arm_PD_dead_overall=series(G_prefilter, G_arm_PD_dead_closed);
QQQ=zpk(G_arm_PD_dead_overall)
Figure ure, subplot(2,2,1),
step(V*G_arm_PD_dead_overall),ylabel('Mobile speed \omega'),xlabel('
Time  '), title(' Angular displacement, PD with deadbeat'), grid
subplot(2,2,2),step(V*G_arm_PD_dead_overal2),ylabel('Mobile speed
\nu'),xlabel(' Time  '), title('linear displacement, PD with
deadbeat'), grid
```

```
subplot(2,2,3), step(V*Ga_arm_open);ylabel('Arm  position,\theta Rad
'),xlabel(' Time '), title('OPEN loop Angular displacement   '),
grid
subplot(2,2,4), step(V*G_close_arm);ylabel('Arm  position,\theta Rad
'),xlabel(' Time '), title('CLOSED loop Angular displacement   '),
grid
disp( ' ========================================================')
else
    error (' Enter 1  for Speed control and 2 for Position control
')
end
```

6. Testing and analysis

6.1 Testing and analysis of PI controller with desired deadbeat response specifications in terms of desired output linear speed

Running the new built-in function for input argument 1, deadbeat(1),that is design of PI controller with deadbeat response, for desired output linear speed of 0.5 m/s, and settling time of 4 seconds and correspondingly defining used DC motor application's and sensor's parameters, dimensions and load torques, built-in function will return the below data including the coefficients that yield the optimal PI controller with desired deadbeat response, also system, controller, and prefilter transfer functions, as well as corresponding response curves of open loop motor system response (Figure 4(a)) and designed system deadbeat response (Figure 4(b)), with and without prefilter. the resulted response curves show achieving the desired fast deadbeat response with desired settling time of 4.1 seconds ,also the following performance specifications; Final steady state output value =0.5 m/s, Steady state error ,Ess=-5.55112e-017,Percent overshoot, Mp=-0.00158942 , Angular speed, Omega =6.66667, Kp =0.194489, Ki =0.217222 , ZPI =1.11689 , Tachometer constant, Ktach =1.8

To verify the design, we can use the simulink model, by defining the obtained coefficients Kp, Ki, ZPI, in MATLAB workspace, switching simulink model to PI controller and correspondingly switching sensor to speed measuring, and running simulation will result in response curves shown in Figure 4(c), the resulted response linear speed/time, angle/time, current/time and torque/ time curves show achieving the desired fast deadbeat response with desired settling time of 4 seconds. Here notice that the simulink model includes the coulomb friction.

Running the new built-in function for desired output linear speed of 0.5 m/s, for PI design with deadbeat response, but for desired settling time of **1** seconds, will result in response curves shown in Figure 5(a), the resulted curves show achieving the desired fast deadbeat response with desired settling time of 4.1 seconds the desired response is achieved and Final steady state output value =0.5, Steady state error Ess=-5.55112e-017, Percent

overshoot, Mp=-0.00148821, Tachometer constant, Ktach =1.8, Angular speed, Omega =6.66667, Kp =6.90286 , Ki =13.9022 ,ZPI =2.01398 . Now, defining the obtained coefficients Kp , Ki , ZPI , in MATLAB workspace, switching simulink model to PI controller and correspondingly switching sensor to speed measuring, and running it , will return response curves shown in Figure 5(b), curves show the identity and correctness of calculations using both built-in function and simulink model

```
======================================================================
                  Please define PMDC parameters
======================================================================
DC motor open loop transfer function in terms of: Speed/Volt:
Zero/pole/gain:
                              50
                        -------------------
                        (s+3.861) (s+1.987)
======================================================================
 DC motor OPEN loop transfer function in terms of: Angle/Volt:
Zero/pole/gain:
                              50
                       ----------------------
                       s (s+3.861) (s+1.987)
======================================================================
=
             Design for PMDC output angular and linear SPEED
PI with desired deadbeat response, e.g. mobile robot output speed
design and control:
======================================================================
                    Define plant parameters ( e.g. mobile robot
======================================================================
 ----------------------------------------------------------------------
-
 Enter desired settling time in seconds, Ts :  4
 ----------------------------------------------------------------------
-
               System Performance specifications:
 ----------------------------------------------------------------------
-
 Final steady state output value =0.5
 Steady state error   Ess=-5.55112e-017
 Percent overshoot,   Mp=-0.00158942
 Tachometer constant, Ktach =1.8
 Angular speed, Omega =6.66667

======================================================================
 Gains and zero of: PI controller & prefilter:

======================================================================
   Kp  =0.194489
   Ki  =0.217222
   ZPI =1.11689

======================================================================
  PI Controller transfer function
Zero/pole/gain:
```

```
                          0.19449 (s+1.117)
                          -----------------
                                  s
-----------------------------------------------------------------------
  Prefilter transfer function
Zero/pole/gain:
                                1.1169
                               ---------
                               (s+1.117)
-----------------------------------------------------------------------
 Closed loop transfer function, PI WITHOUT prefilter
Zero/pole/gain:
                          9.545 (s+1.117)
                  --------------------------------
                  (s+1.011) (s^2 + 4.853s + 18.98)
-----------------------------------------------------------------------
Closed loop transfer function, PI WITH prefilter
Zero/pole/gain:
                          10.6607 (s+1.117)
              ------------------------------------------
              (s+1.117) (s+1.011) (s^2 + 4.853s + 18.98)
-----------------------------------------------------------------------
```

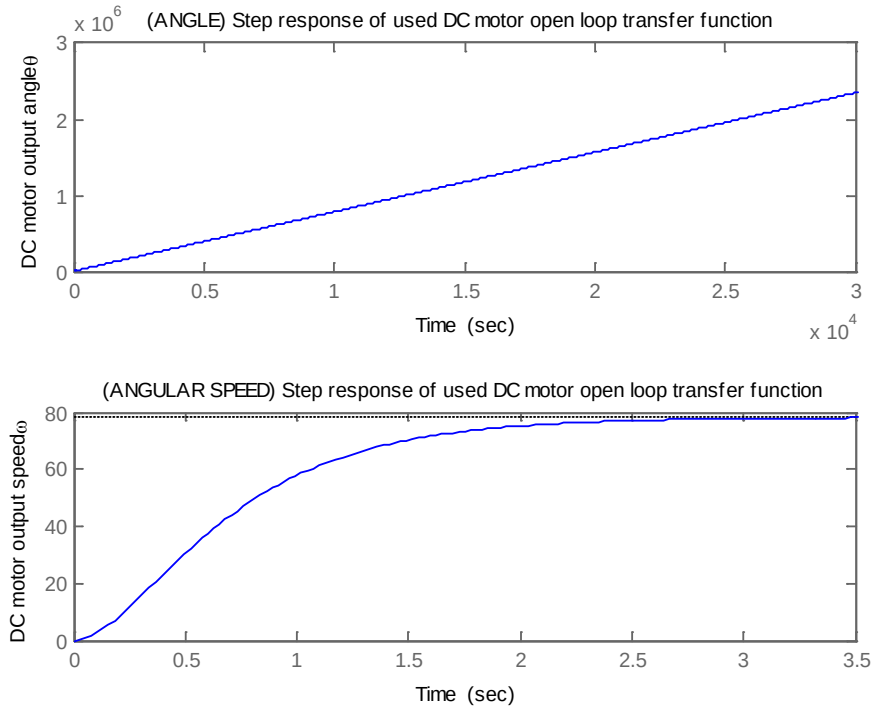

Figure 4 (a) Using built-in function: response curves of open loop motor system

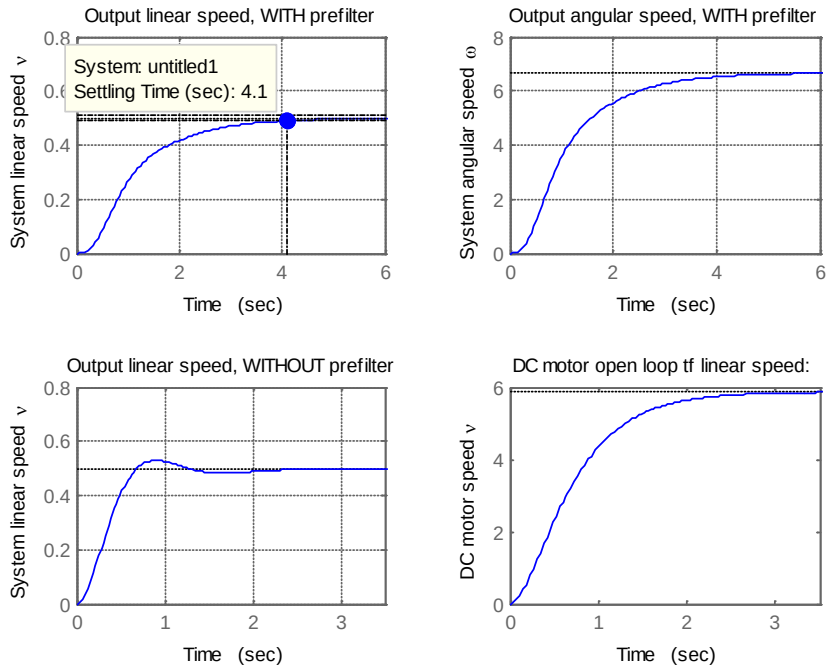

Figure 4(a) Using built-in function : designed system deadbeat response with desired T_s=4 s , linear and angular speed, with and without prefilter

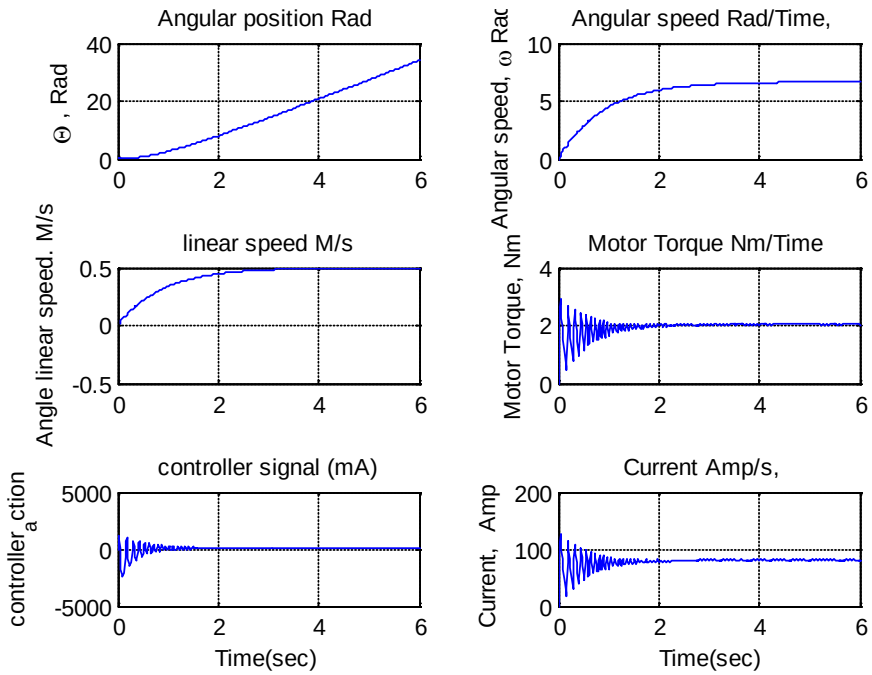

Figure 4 (c) Using simulink model : designed system deadbeat response with desired T_s=1 s , linear and angular speed, with and without prefilter, also current and torque and controller signal

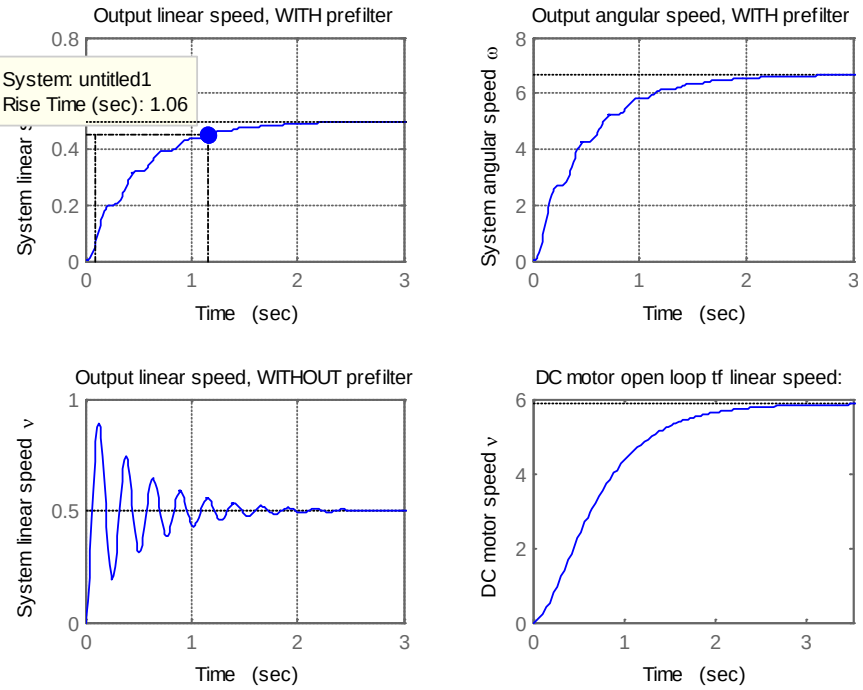

Figure 5(a) Built-in function : designed system deadbeat response with desired T_s=3 s , linear and angular speed, with and without prefilter

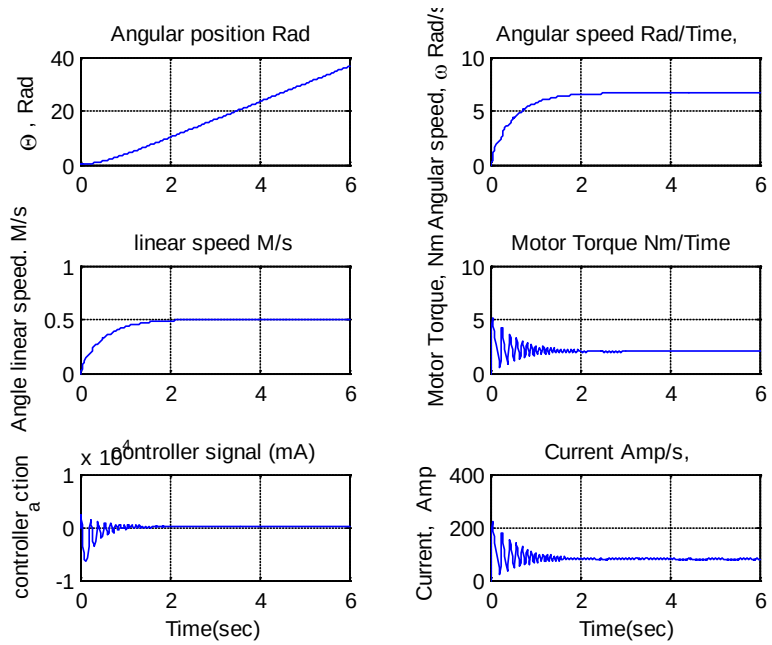

Figure 5(b) Using simulink model : designed system deadbeat response with desired T_s=1 s , linear and angular speed, with and without prefilter, also current and torque and controller signal

6.2 Testing and analysis of PD controller with desired deadbeat response specifications in terms of desired output position

Running the new built-in function for input argument 2, deadbeat(2),that is design of PD controller with deadbeat response, for desired output angle of 90, and settling time of 2 seconds and correspondingly defining used DC motor application's and sensor's parameters, dimensions and load torques, built-in function will return response curves of open loop motor system response and designed system deadbeat response (Figure 6(a)),the resulted response curves show achieving the desired fast deadbeat response with desired settling time of 1.8 (error of 0.2) seconds ,also the following performance specifications; Final steady state output value =90, Steady state error ,Ess = -2.84217e-014,Percent overshoot, Mp= -0.334091 , Angular displacement, theta =90, Kp = 32.6243, Kd =14.5295 , ZPD =2.24539 , Tachometer constant, Ktach =0.1333**.** To verify the design, using the simulink model, define the obtained coefficients Kp, Kd, ZPD, in MATLAB workspace, switching simulink model to PD controller and correspondingly switching sensor to angle measuring, and running simulation will result in response curves shown in Figure 6(b).

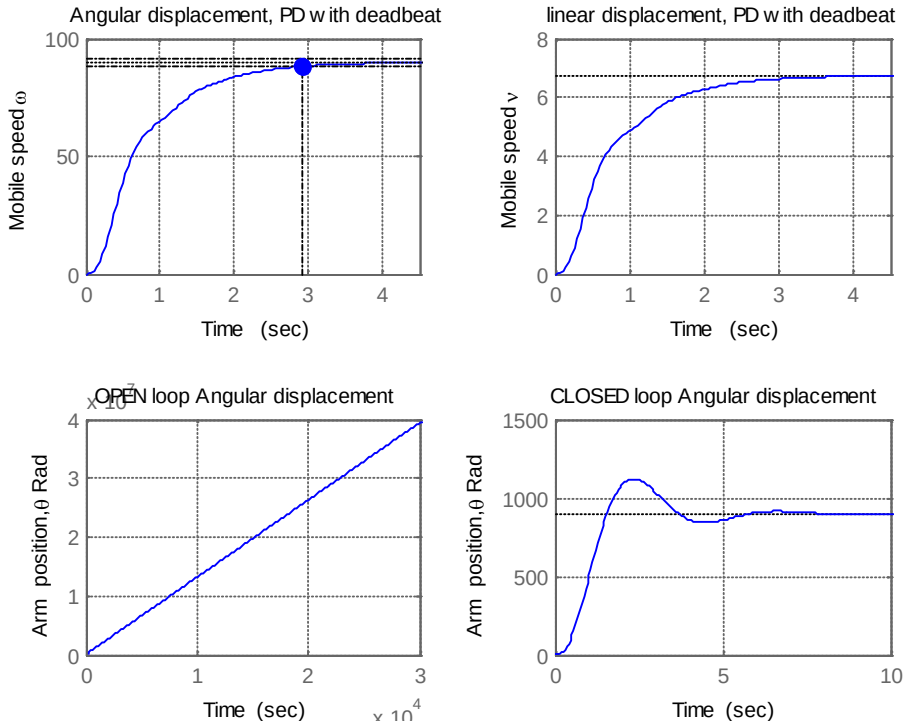

Figure 6 (a) Built-in function : designed system deadbeat response with desired T_s=3 s , linear and angular speed, with and without prefilter

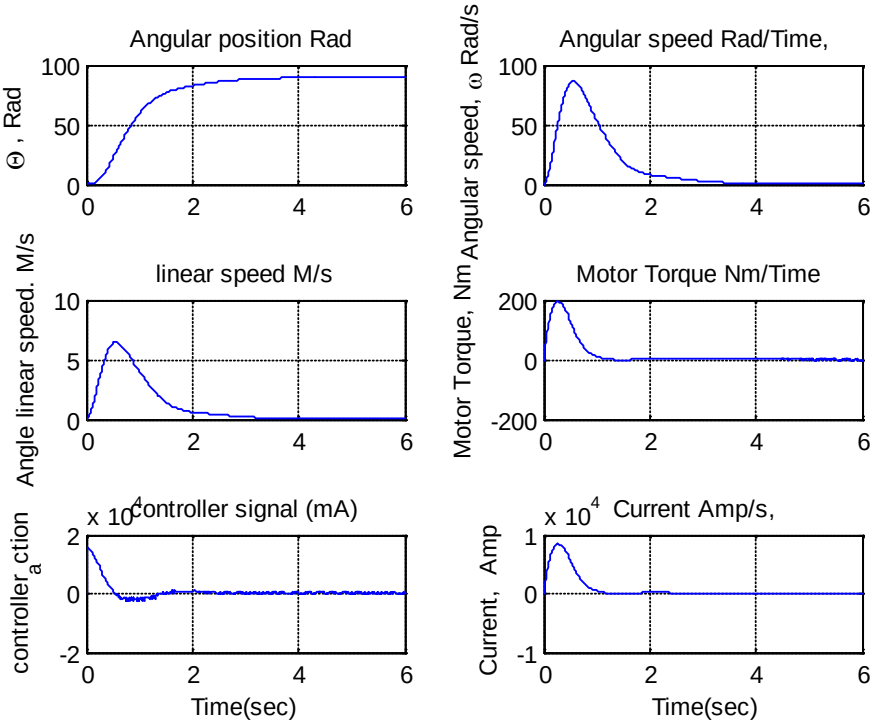

Figure 6(b) Using Simulink model : designed system deadbeat response with desired T_s=2 s , linear and angular speed, , also current and torque and controller signal

Conclusion

Mechatronics design of most used electric DC machine and corresponding motion control in terms of desired output position or velocity for desired deadbeat response specifications is proposed, the design is facilitated using either or both proposed new MATLAB built-in function named deadbeat( ), and new Simulink model, both oriented on Mechatronics design of both electric machine and motion control systems. The introduced new MATLAB built-in function determines the coefficients that yield the optimal deadbeat response for desired output speed or angle control of a given electric machine, also calculates transfer functions (open and closed), of used motor, controller, prefilter and overall system, also, plots overall system deadbeat response and all corresponding response curves and deadbeat response specification, particularly *Mp*, *Ess*, *Ts*.

The proposed models and the supporting new MATLAB built-in function were tested for different electric DC machines and for different applications, the results show the applicability and accuracy of the proposed designs in Mechatronics motion control applications for achieving desired deadbeat specification, resulting in simplification and acceleration of Mechatronics motion control system design process. The proposed design, model and function, can be used for various Mechatronics motion control

design applications, where the proper selection of actuating machine and design of precise motion control system are of concern.

References:

Richard C. Dorf, Robert H. Bishop, Modern Control Systems 12 Ed, Pearson Education, Inc., 2001

Thomas R. Kurfess, "Robotic and Automation Handbook," Washington, D.C., United States of America, 2005.

Mohsen Shahinpoor, "A Robot Engineering Textbook,"HAPPER & ROW, Publishers, Inc.,10 East 53rd street, New York, NY10022, 1987.

Bashir M. Y. Nouri ,modeling and control of mobile robots Proceeding of the First International Conference on Modeling, Simulation and Applied Optimization, U.A.E. Feb. 1-3, 2005.

Chun Htoo Aung, Khin Thandar Lwin, and Yin Mon Myint, Modeling Motion Control System for Motorized Robot Arm using MATLAB, World Academy of Science, Engineering and Technology 42, 2008.

Ramjee Prased, modeling of robotic arm,10804900, Rh680B54, B.TECH (LEET) BCE

Ahmad A. Mahfouz ,Mohammed M. K., Farhan A. Salem, Modeling, Simulation and Dynamics Analysis Issues of Electric Motor, for Mechatronics Applications, Using Different Approaches and Verification by MATLAB/Simulink (I). IJISA, Vol. 5, No. 5, 39-57 April 2013 .

Wai Phyo 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

Farhan A. Salem; Refined models and control solutions for Mechatronics design of mobile robotic platforms, Estonian *Journal of Engineering*, 2013.

Farhan A. Salem; Modeling, simulation and control issues for a Robot arm; Education and Research (III). International Journal of Intelligent Systems and Applications(IJISA),2013

Farhan A. Salem, Mechatronics design of Small electric Vehicle's, International Journal of Mechanical & Mechatronics Engineering ,1310701-5252 IJMME / IJENS, 2013-02-13

M.P.Kazmierkowski, H.Tunia "Automatic Control of Converter-Fed Drives", Warszawa 1994.

R.D. Doncker, D.W.J. Pulle, and A. Veltman. Advanced Electri-cal Drives: Analysis, Modeling, Control. Springer, 2011.

Grzegorz Sieklucki, Analysis of the Transfer-Function Models of Electric Drives with Controlled Voltage Source PRZEGL ˛ AD ELEKTROTECHNICZNY (Electrical Review), ISSN 0033-2097, R.88NR7a/2012.

Hedaya Alasooly, Control of DC motor using different control strategies, global journal of technology and optimization, vol. 2 , 2011.
Farhan A. Salem, Modeling controller selection and design of electric DC motor for Mechatronics applications, using different control strategies and verification using MATLAB/Simulink, submitted to International Journal of Intelligent Systems and Applications(IJISA),2013
Math Works, 2001, Introduction to MATLAB, the Math Works, Inc. Control System Toolbox, the Math Works, Inc.