# PID Controller Design for Reference Tracking of Single Link Manipulator


**Sayan Das and Naiwrita Dey**

**Abstract** This paper is aimed to design a PID controller for reference signal tracking. Design problem is formulated to control the angular position of a single-link manipulator at any particular desired set value. Implementation of the overall feedback control system and the proposed controller has been carried out in Python language. Additive uncertainty is considered with the transfer function model. Simulation results illustrate the usefulness of the proposed controller for angular velocity tracking in presence of uncertainty. Average mean square error and settling time have been calculated. Overall stability of the feedback system has been tested by obtaining the eigenvalues of the closed-loop system.

## 1 Introduction

Robotics is getting more and more popular in industrial field as it reduces manpower required for doing heavy jobs. Main types of robots used in industrial applications are "multiple-link manipulators" with different degrees of freedom. These kinds of robots can move according to their DOF and complete many tasks. Single-link manipulator model is made out of one fixed base, on which the whole system resides, a revolute joint and an arm, which can move freely on the x–y plane. DC motor or servomotors are mainly used to rotate the arms of manipulators. Here, a simple DC motor with high torque constant is used to move the arm of the manipulator. First, the dynamic model of that motor is designed [1]. Then, control logic is applied on the motor to move the arm to a specific angular position [2–4]. For tuning control parameters, simple tuning method is used [5]. Nowadays, in many industrial applications, rigid link manipulators cannot satisfy the needs. So, "Flexible link manipulators" have become an option. In comparison with rigid body manipulators, flexible manipulators have many advantages including higher efficiency, more flexibility, high load capacity,

S. Das
Department of AEIE, RCCIIT, Kolkata, India

N. Dey (✉)
Department of ECE, RCCIIT, Kolkata, India



J. Gu et al. (eds.), *Communication and Control for Robotic Systems*, Smart Innovation, Systems and Technologies 229, https://doi.org/10.1007/978-981-16-1777-5_3

low energy consumption and larger application field. Flexible link manipulators have wide applications in aerospace engineering, precision instrument production, construction and many other fields [6]. With every practical system, there remains some uncertainty with the parameter values of that system. So, the control model needs to be more accurate to induce its controllability on those uncertain models. There are various control methods like "Robust control" or "Fractional control" methods to deal with parameter uncertainty or external disturbances [7] which are available for controlling these types of manipulators [8–10]. This paper also presents results of controlling an uncertain model of rigid body single-link manipulator. The stability of the system has been taken in account at the time of testing uncertain models.

This paper presents simulations of dynamic behavior of a rigid body single-link manipulator controlled by simple PID logic. PID control is a very popular control method applicable for controlling linear systems. By tuning its parameters, desired controlled output can be generated.

## 2 Mathematical Modeling of Single-Link Manipulator

The physical system consists of a fixed base, a DC motor and an arm. The arm is movable in x–y plane. Here, the arm acts as the "mechanical load" of the motor. The mathematical modeling of a single-link manipulator can be constructed by modeling the motor part, which is giving driving torque to the manipulator and then the arm part.

### *2.1 Transfer Function Model of the System*

DC motor converts electrical energy into mechanical energy. It has a permanent magnet that produces magnetic field around the armature circuit. According to "Fleming's Left-hand rule" that magnetic field along with the electric field of the armature inductance creates rotary motion of the rotor. When the motor spins with its full speed, a "Back EMF" is generated that decreases the applied armature voltage, which decreases the current flowing in the armature circuit. At points $M_1$ and $M_2$, mechanical load is connected. Figures 1 and 2 represent the armature circuit of DC motor and diagram of single link manipulator respectively.

**Parameters**:

$$m = \text{Mass of the arm}$$

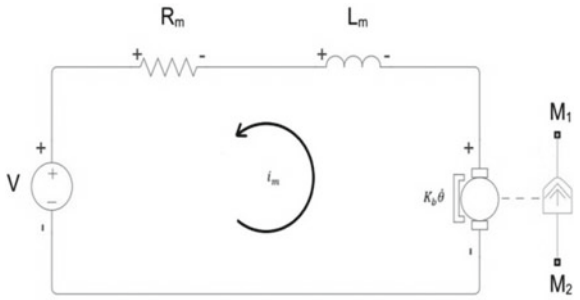



**Fig. 1** DC motor armature circuit

$g$ = Gravitational acceleration

$l$ = Length of the arm

$V$ = Applied armature voltage,

$i_m$ = Armature current,

$R_m$ = Armature resistance,

$L_m$ = Armature inductance,

$K_b$ = Back emf constant,

$\theta$ = Angular position of rotor,

$\dot{\theta}$ = Angular velocity of rotor,

$$J = \text{Rotor inertia},$$

$$b = \text{Motor friction constant},$$

$$K_T = \text{Motor torque constant},$$

$$T_m = \text{Torque produced by motor}$$

Applying KVL to the armature circuit,

$$V = L_m \frac{di_m}{dt} + K_b\dot{\theta} + R_m i_m \tag{1}$$

Taking Laplace transform of Eq. (1),

$$\begin{aligned} V(s) &= L_m s.I_m(s) + K_b s.\theta(s) + R_m.I_m(s) \\ &=> V(s) = I_m(s)[R_m + L_m s] + K_b s.\theta(s) \end{aligned}$$

So,

$$I_m(s) = \frac{V(s) - K_b s.\theta(s)}{(R_m + L_m s)} \tag{2}$$

Torque produced by rotor,

$$T_m = K_T.i_m$$

Taking Laplace transform of it,

$$T_m(s) = K_T.I_m(s)$$

**Kinematics**:

For the arm in Fig. 2,

$$x = \frac{l}{2}\cos\theta$$

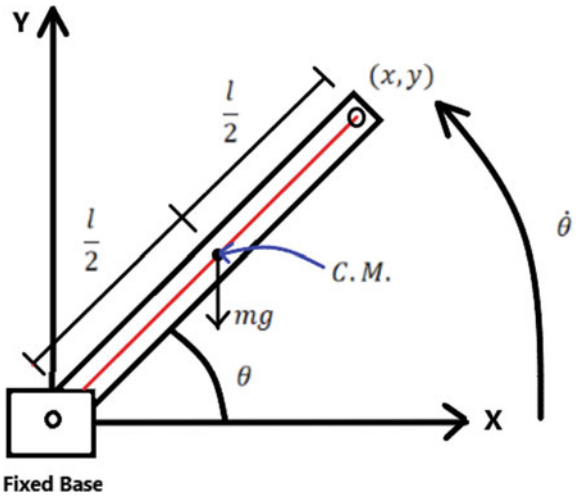



**Fig. 2** Single-link manipulator

$$y = \frac{l}{2}\sin\theta$$

Velocity along x and y axis is, respectively,

$$v_x = \dot{x} = -\frac{l}{2}\dot{\theta}\sin\theta$$

$$v_y = \dot{y} = \frac{l}{2}\dot{\theta}\cos\theta$$

Overall velocity of the arm is,

$$v = \sqrt{v_x^2 + v_y^2} = \frac{l}{2}\dot{\theta}$$

Now, the kinetic energy of the system,

$$E_k = \frac{1}{2}\left(mv^2 + I\dot{\theta}^2\right) = \frac{1}{6}ml^2\dot{\theta}^2$$

Potential energy of the system,

$$E_p = \frac{1}{2}\text{mgl}\sin\theta$$

So, Lagrangian of the system is,

$$\mathcal{L} = E_k - E_p = \frac{1}{6}\text{ml}^2\dot{\theta}^2 - \frac{1}{2}\text{mgl}\sin\theta$$

From Lagrange equation for torque,

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = T_m$$

$$\frac{1}{3}ml^2\ddot{\theta} + \frac{1}{2}\text{mlg}\cos\theta = T_m \tag{3}$$

For linearization, assuming $\theta$ is very small, $\cos\theta \approx 1$.
So, Eq. (3) becomes,

$$\frac{1}{3}\text{ml}^2\ddot{\theta} + \frac{1}{2}\text{mlg}\theta + b\dot{\theta} = T_m$$
$$=> 2\text{ml}^2\ddot{\theta} + 6b\dot{\theta} + 3\text{mlg}\theta = 6T_m$$

where $b\dot{\theta}$ = damping part from the driving motor.
Dynamic equation of the arm can be expressed as,

$$J\ddot{\theta} + 6b\dot{\theta} + p\theta = 6T_m \tag{4}$$

where, $J = 2\text{ml}^2$, $p = 3\text{mlg}$.
Taking Laplace transform of Eq. (4),

$$\left\{Js^2\theta(s) - \theta(0) - \dot{\theta}(0)\right\} + 6bs.\theta(s) + p\theta(s)$$
$$= 6T_m(s) = 6K_T.I_m(s)$$
$$=> \theta(s)\left[Js^2 + 6bs + p\right] = 6K_T.I_m(s)$$
$$=> I_m(s) = \frac{\theta(s).\left[Js^2 + 6bs + p\right]}{6K_T} \tag{5}$$

Comparing (2) and (5), transfer function $Y(s)$ can be obtained,

$$Y(s) = \frac{\theta(s)}{V(s)} = \frac{6K_T}{\left[(R_m + L_m s)\left(Js^2 + 6bs + p\right) + s.K_b.K_T\right]} \tag{6}$$

State-space form of the above system has been obtained as,

$$\frac{\text{d}}{\text{d}t}\begin{pmatrix}\theta\\ \dot{\theta}\\ i_m\end{pmatrix} = \begin{pmatrix}0 & 1 & 0\\ -\frac{p}{J} & \frac{-6b}{J} & \frac{6K_T}{J}\\ 0 & \frac{-K_b}{J} & \frac{-R_m}{L_m}\end{pmatrix}.\begin{pmatrix}\theta\\ \dot{\theta}\\ i_m\end{pmatrix} + \begin{pmatrix}0\\ 0\\ \frac{1}{L_m}\end{pmatrix}.V \tag{7}$$

$$y = \begin{pmatrix}0\ 1\ 0\\ 0\ 0\ 0\\ 0\ 0\ 0\end{pmatrix}\begin{pmatrix}\theta\\ \dot{\theta}\\ i_m\end{pmatrix}$$

### 2.2 Uncertainty Modeling of the System

The system represented by the transfer function $Y(s)$ cannot be considered as the practical system. Practical systems can have different uncertainties which can deviate the nature of the practical system from the ideal one. Such uncertainties come mainly from external disturbances or real parameter variations. Here, the components of the system have some variation or uncertainty in their parameter values. So, parameters like m, $R_m$ and $L_m$ contributing to the transfer function $Y(s)$ have some uncertainty in their value. The practical parameters of the system can be written as:

$m_p = m + \Delta_m$, $R_{mp} = R_m + \Delta_{R_m}$, $L_{mp} = L_m + \Delta_{L_m}$, $b_p = b + \Delta_b$ where $\Delta_r =$ uncertainty of the parameter "r" and $r_p =$ practical value of the parameter "r." So the explicit form of state-space equation of the uncertain single-link manipulator model becomes:

Finally, the transfer function of the system with uncertainty is given by,

$$Y_U(s) = \frac{6K_T}{\left[\left(R_{mp} + L_{mp}s\right)\left(2m_p l^2 s^2 + 6bs + 3m_p gl\right) + s.K_b.K_T\right]} \tag{8}$$

## 3 Controller Design

### 3.1 Block Diagram of Single-Link Manipulator Angular Position Closed-Loop System

The closed loop angular position control system for single link manipulator is shown in Fig. 3.

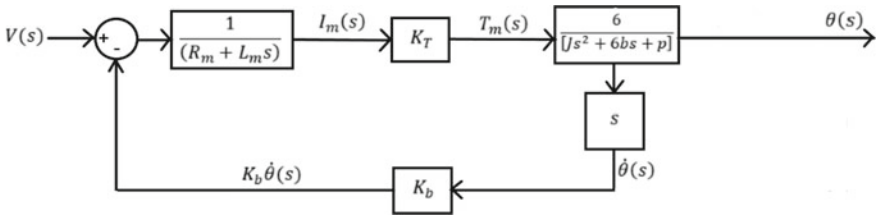



**Fig. 3** Single-link manipulator angular position closed-loop block diagram

### 3.2 Implementing PID Control to the Nominal Manipulator System

The term PID stands for "Proportional Integral Derivative." PID controllers are flexible, simple and easy to implement. By using proper optimization, PID controller can control and regulate time-domain response of wide range of processes. The time-domain equation of PID controller is as follows:

$$\boldsymbol{u(t) = u_{\text{bias}} + K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}} \tag{9}$$

where $u(t)$ = controller output, $u_{\text{bias}}$ = initial value of controlled output (let $u_{\text{bias}} = 0$ for this case), $e(t)$ = error term, i.e., difference between set point (SP) and process variable (PV), $K_p$ = proportional gain, $K_i = \frac{K_p}{\tau_i}$ ($\tau_i$ is integral time constant), and $K_d = K_p \tau_d$ ($\tau_d$ is derivative time constant). Figure 4 shows the closed loop angular position control system with PID controller.

Laplace transforming (8),

$$U(s) = E(s)\left[K_p + \frac{K_i}{\mathrm{s}} + K_d \mathrm{s}\right]$$

Control transfer function,

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{\mathrm{s}} + K_d \mathrm{s}$$

So, the overall open-loop transfer function of the controlled model becomes (Fig. 4),

$$G(s) = Y(s).C(s)$$

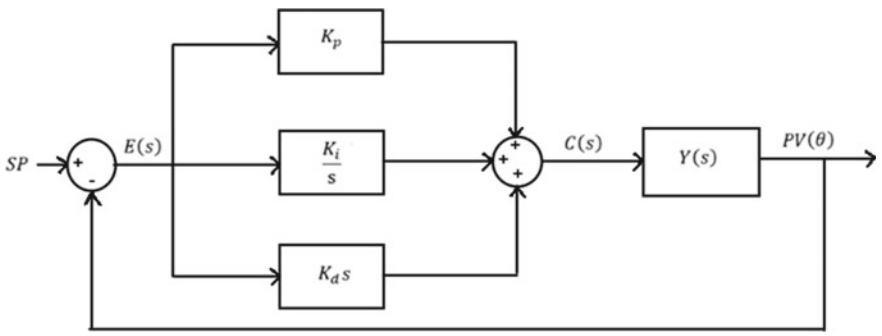



**Fig. 4** Block diagram of PID control in single-link manipulator for controlling angular position

**Table 1** Tuning PID parameters

| Parameter change | Rise time (s) | Maximum overshoot (%) | Settling time (s) | Steady-state error |
|---|---|---|---|---|
| Increasing $K_p$ | Decrease | Increase | Small change | Decrease |
| Increasing $K_i$ | Decrease | Increase | Increase | Eliminate |
| Increasing $K_d$ | Small change | Decrease | Decrease | Small change |

**Tuning PID parameters**: To get desired response, the control parameters like $K_p$, $K_i$ and $K_d$ should be tuned properly. By checking the output response plot, it can be pointed out values of which parameters should be increased or decreased. Table 1 shows the same.

## 4 Implementation of Proposed Controller in Python

Python, as an object-oriented programming language, is simple and versatile. All the simulations presented in this paper have been done in Python language. For simulating system response in different control environments, Python's dedicated control system library is used [11]. For some mathematical purposes, different available modules are used. All the programming has been done using "Anaconda Navigator" and "Jupyter Notebook."

Anaconda is an open-source software which directly comes with different Python and R language IDEs or platforms preinstalled. Anaconda is best used for machine learning and data analysis tasks. The software also comes with common modules and libraries installed but Anaconda's Python package does not include control system library by default, so that it is need to be installed first using "Anaconda Prompt." By writing "pip install control" command in the prompt, the library can be installed easily as shown in Fig. 5.

Then, after opening Anaconda Navigator, Jupyter notebook is to be selected, and it will automatically open Jupyter notebook in the browser at predefined path. There is also another Python IDE "Spyder" is available. Anaconda makes Jupyter notebook and Spyder databases interconnected, so anyone can easily switch in between them.

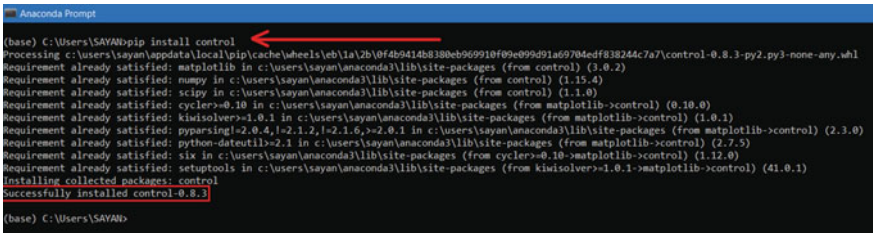



**Fig. 5** Installing control system library in Anaconda

The following figure shows the Anaconda Navigator and Jupyter notebook interface. Figures 6 and 7 show the navigator and editor interface respectively.

Jupyter notebook is preferable for testing programs as it allows users to run different sections of program at any time. So, it is easy to find errors in the code.

Python control systems library is a Python package that emulates maximum functions from MATLAB's control system toolbox and implements techniques for analyzing any control system object.

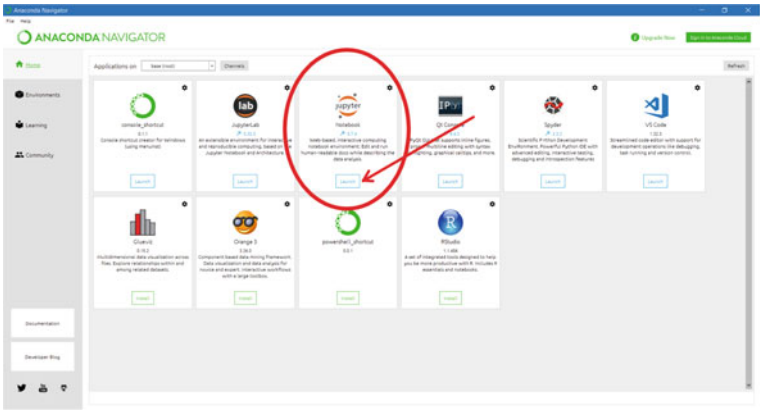



**Fig. 6** Launching Jupyter notebook from Anaconda Navigator

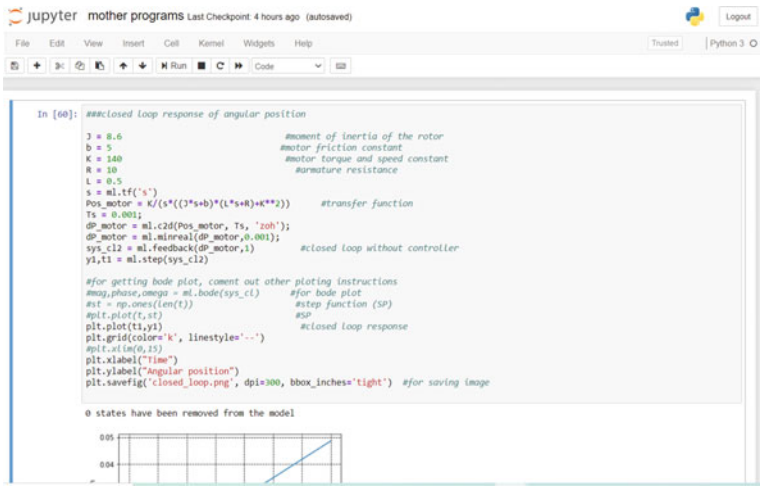
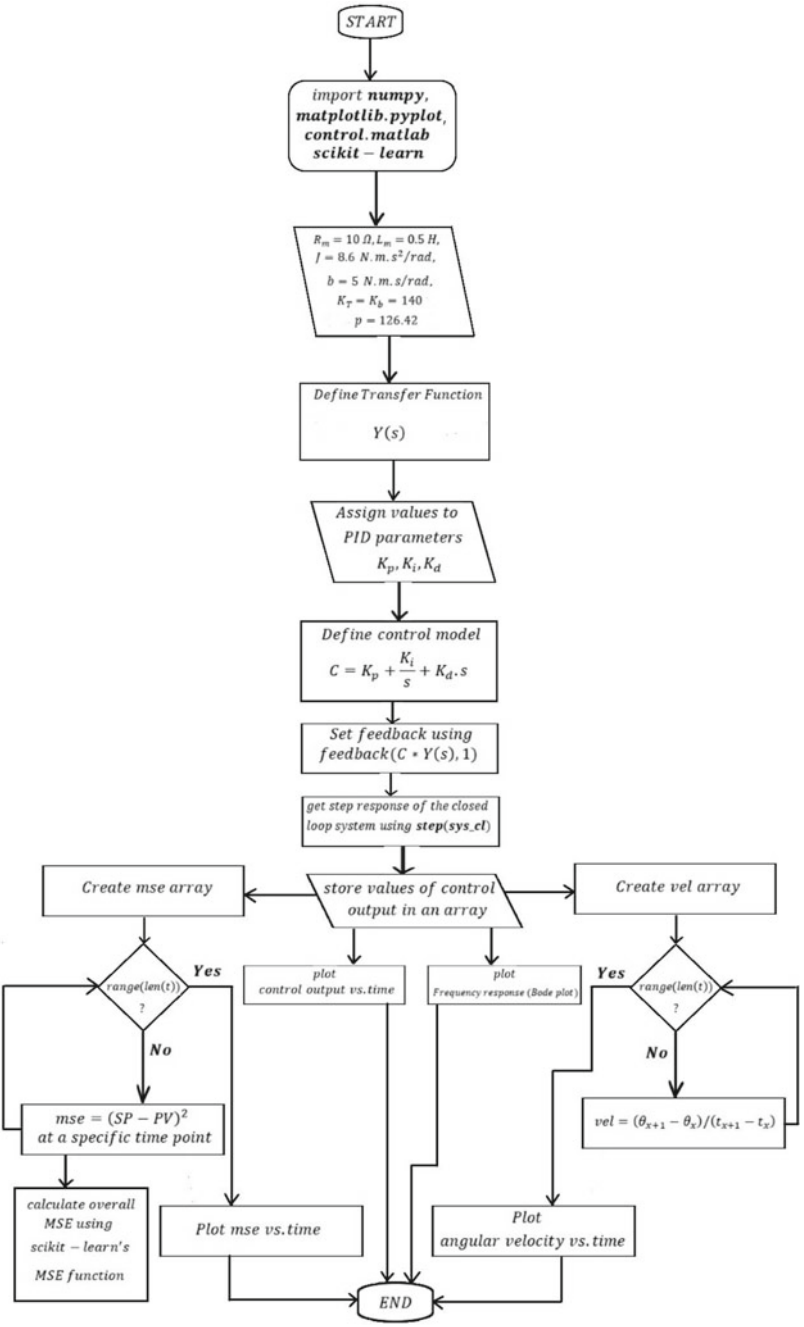



**Fig. 7** Jupyter notebook interface

## *4.1 Flowchart of the Simulation Program*



### 4.2 Programming for Simulating Uncertain Model

For simulating the uncertain model in Python, following steps have been taken for developing the program:

1. A constant variation is applied to every uncertain parameter. As an example, values of all uncertain parameters would vary within ± 5% of their nominal value.
2. As stated in point 1, a function is created to produce an array consisting five different values of each uncertain parameter within that range. By implementing multiprocessing, the number of values in those arrays can be increased.
3. A for loop is built to run the simulation program described in the above flowchart using single values from those arrays in each iteration.
4. Outside the for loop two null arrays, one for MSE and another for settling time is created. For each iteration, MSE and settling time are added to those arrays, respectively.
5. During each iteration, plots of control signal or frequency response are plotted as per requirement.
6. Average MSE and average settling time for that constant variation are calculated.
7. At last, a small code snippet is created for evaluating stability of the system for those values of uncertain parameters. The flowchart in Fig. 8 describes the same.

## 5 Stability Analysis of the Overall Closed Loop System

Stability of an LTI system can be defined by calculating the eigenvalues of the "state matrix" of that system. For any bounded initial condition and zero input, an LTI system is called "Lyapunov stable" if the state remains bounded. For any bounded initial condition and zero input, an LTI system is called "Asymptotically stable" if the state converges to zero.

For evaluating stability of the single-link manipulator model, nominal values of the system parameters are listed in Table 2.

Substituting these values in the state-space model (7),

$$\frac{d}{dt}\begin{pmatrix}\theta\\ \dot{\theta}\\ i_m\end{pmatrix} = \begin{pmatrix}0 & 1 & 0\\ -14.7 & -3.4884 & 97.674\\ 0 & -16.279 & -20\end{pmatrix}.\begin{pmatrix}\theta\\ \dot{\theta}\\ i_m\end{pmatrix} + \begin{pmatrix}0\\ 0\\ 2\end{pmatrix}.V$$

Here, state matrix $A = \begin{pmatrix}0 & 1 & 0\\ -14.7 & -3.4884 & 97.674\\ 0 & -16.279 & -20\end{pmatrix}$.

Calculated eigenvalues of A are:

$\lambda_1 = -11.6562 + 39.1731i$, $\lambda_2 = -11.6562 - 39.1731i$ and

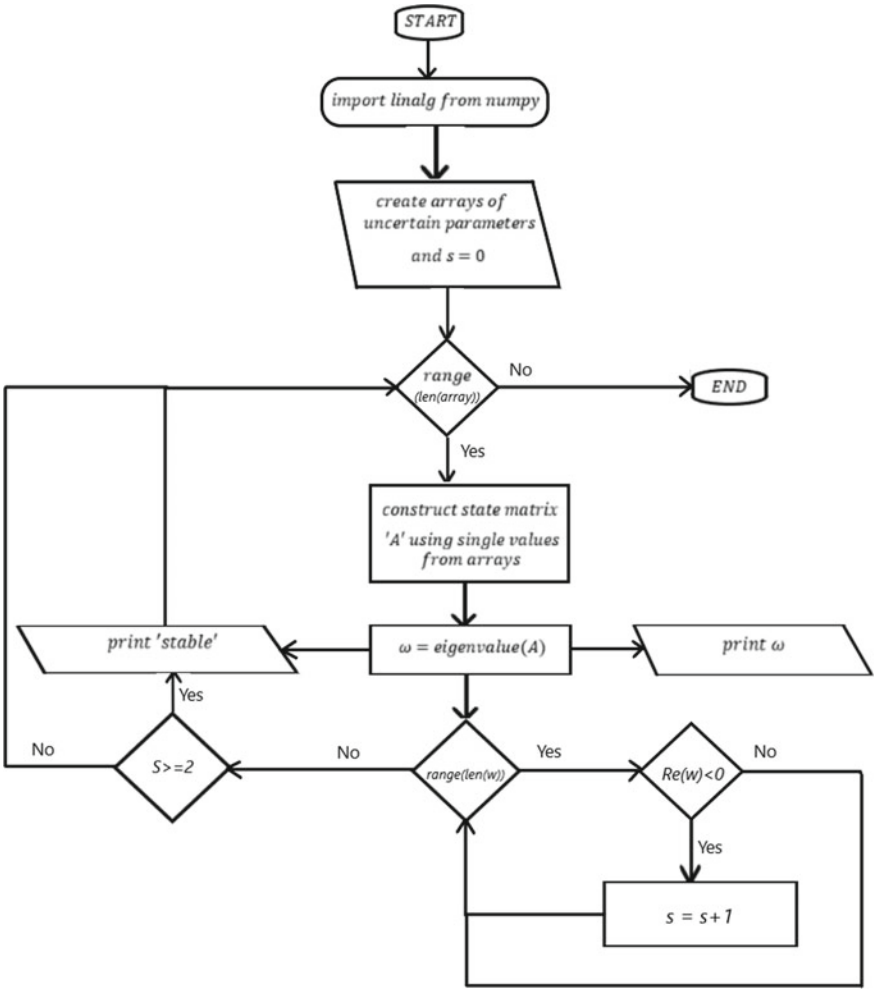



**Fig. 8** Flowchart of program for evaluating stability of the system

$$\lambda_3 = -0.176006$$

So,

1. Two eigenvalues $\lambda_1$ and $\lambda_2$ are complex conjugates, and $\lambda_3$ is a negative real number.
2. $Re(\lambda_1) = Re(\lambda_2) < 0$ and $Re(\lambda_3) < 0$.

The negative real part of the eigenvalues implies the asymptotic stability of the overall closed-loop system.

**Table 2** Nominal values of parameters

| Parameter | Value |
|---|---|
| $R_m$ | 10 Ω |
| $L_m$ | 0.5 H |
| $m$ | 1 kg |
| $b$ | 5 N.m.s/rad |
| $K_T$ | 140 N.m/A |
| $K_b$ | 140 V.s/rad |
| $l$ | 2.07 m |
| $J$ | 8.6 |
| $p$ | 126.42 |

# 6 Simulation Results

## *6.1 Plot of Closed-Loop Step Response of the System (Without Any Controller)*

In open-loop condition, the arm rotates without stopping, and the angular position of the arm increases as long as the electrical energy is provided, as it can be seen from the Fig. 9. Due to arm's weight, the curve is slightly bent (Fig. 9).

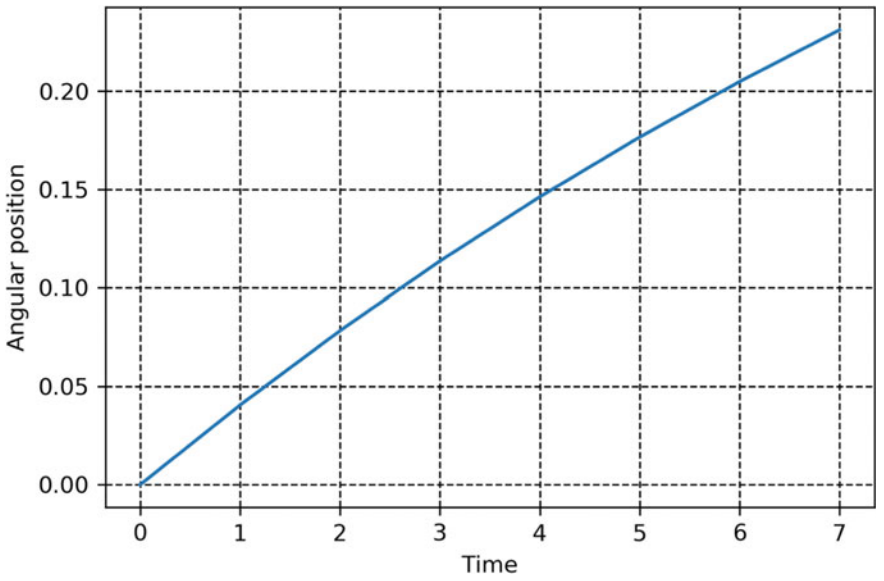



**Fig. 9** Open-loop step response of the system

**Table 3** Values of PID parameters, MSE and settling time for nominal system

| PID parameters | MSE | Settling time (s) |
|---|---|---|
| $K_p = 100$<br>$K_i = 50$<br>$K_d = 1$ | 0.01536 | 1.9768 |

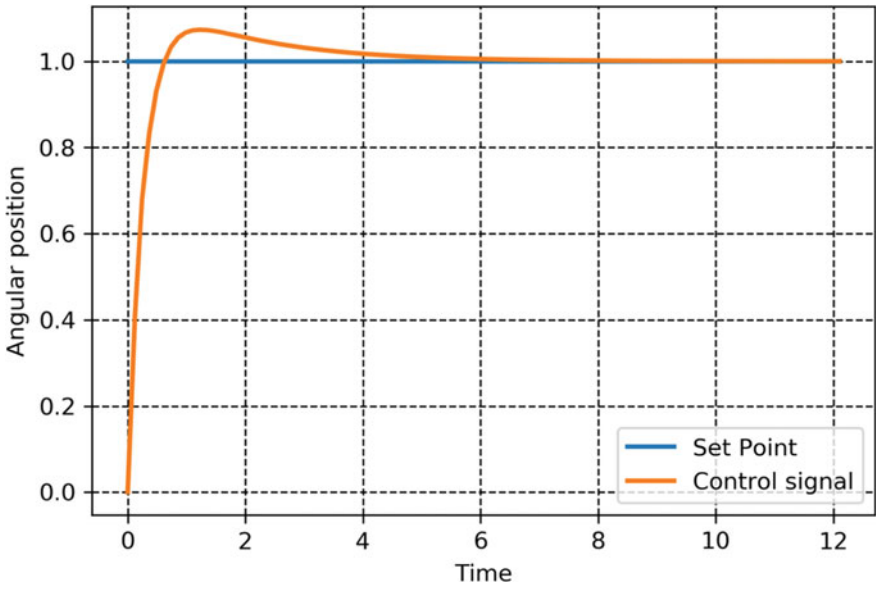



**Fig. 10** PID controlled step response of the system

## 6.2 *Plot of PID Controlled Step Response of the Nominal System*

In case of choosing PID gains, "Manual tuning" has been taken into account. Such a case study with the values of PID parameters is used for simulating; MSE and settling time are presented in Table 3.

Figure 10 shows how perfectly the implemented PID controller fixes the angular position of the arm according to the set point.

## 6.3 *Plot of Squared Error with Time of the Controlled Response*

Figure 11 shows how the squared error of the PID controlled system (i.e., SP-PV) is made zero with time.

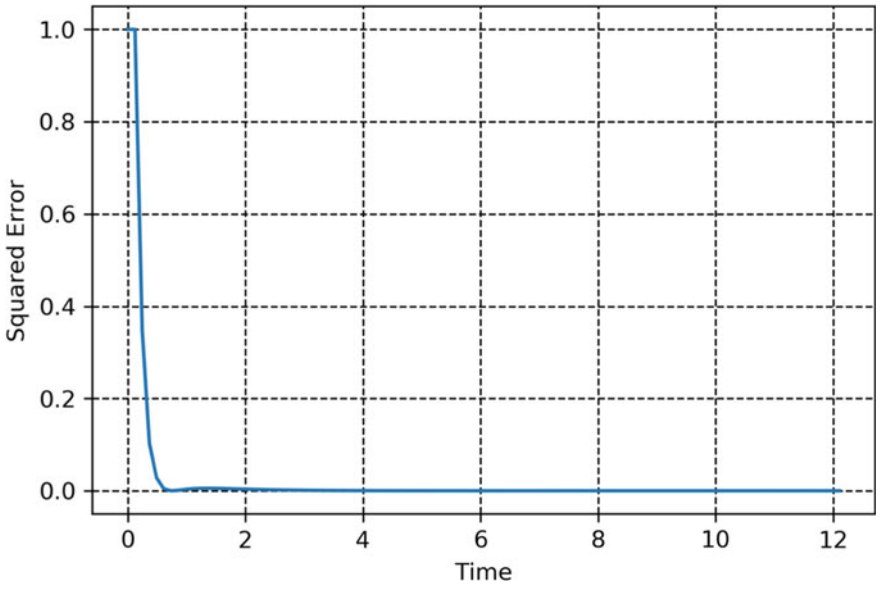



**Fig. 11** Squared error versus time plot

## *6.4 Plot of Angular Velocity with Time*

In Fig. 12, the change of angular velocity while PID controller is fixing the desired angular position is shown. At around 1.7 to 1.8 s, the angular velocity goes slightly negative and then settles to 0 as angular position is fixed.

## *6.5 Frequency Response of the Uncertain Model*

For simulation, two different values for $\Delta$ are taken:

Different case studies for these values of uncertainties are shown in Tables 4 and 5 for considering different values of $\Delta$. Figure 13 depicts the frequency response of the uncertain system.

**For $\Delta = \pm 5\%$**

**For $\Delta = \pm 7\%$**

From the obtained results, it can be said that parameter uncertainty within $\pm$ 7% of their value is permeable for the model, the model remains stable, and the angular position reference command is tracked within allowable range of mean square error.

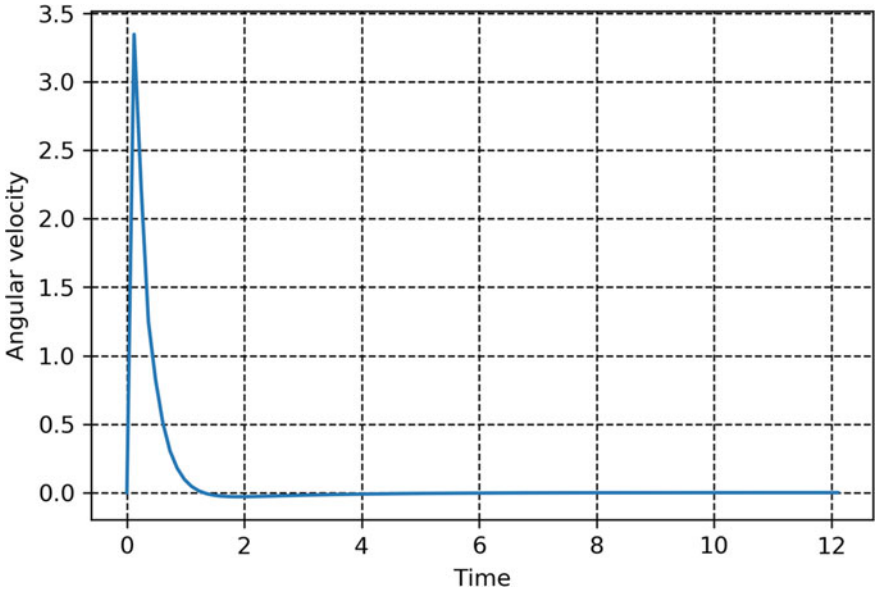



**Fig. 12** Angular velocity versus time plot

**Table 4** Values of PID parameters, MSE and Settling time for uncertainty, $\Delta = \pm$ 5%

| PID parameters | MSE | Average settling time (sec) | Stability |
|---|---|---|---|
| $K_p = 25$<br>$K_i = 50$<br>$K_d = 50$ | 0.0234 | 2.3781 | Stable |
| $K_p = 50$<br>$K_i = 80$<br>$K_d = 2$ | 0.02707 | 1.8247 | Stable |
| $K_p = 100$<br>$K_i = 50$<br>$K_d = 1$ | 0.01161 | 1.5589 | Stable |

**Table 5** Values of PID parameters, MSE and Settling time for uncertainty, $\Delta = \pm$ 7%

| PID parameters | Average MSE | Average settling time (s) | Stability |
|---|---|---|---|
| $K_p = 25$<br>$K_i = 50$<br>$K_d = 5$ | 0.02330 | 1.09128 | Stable |
| $K_p = 50$<br>$K_i = 50$<br>$K_d = 2$ | 0.02705 | 1.8247 | Stable |

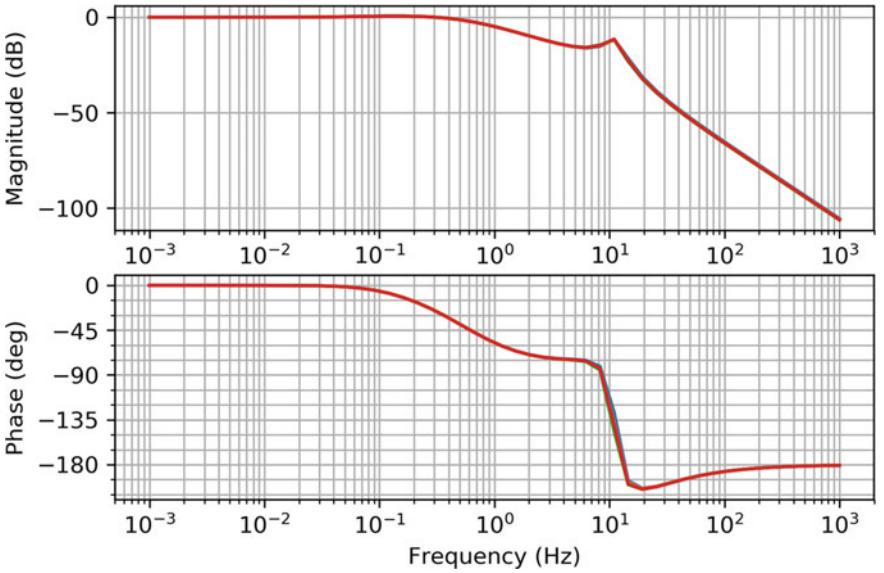



**Fig. 13** Bode plot of the uncertain model ($\Delta = \pm 5\%$, $K_p = 100$, $K_i = 50$ and $K_d = 1$)

## 7 Conclusion

In this paper, PID controller is designed and implemented to control the angular position of a single-link manipulator system which is a very popular robotic application. The manipulator system has been modeled considering additive uncertainty which makes the system closer to practical system. Simulation experimentation is performed for different values of uncertain parameters. Mean square error is calculated as performance metric along with settling time. As the closed-loop system stability is a major concern, the calculated eigenvalues containing negative real part assures the same for this present work. Application of control system toolbox in Jupyter notebook is another important feature of this work. This work can be further easily extended to embed the simulated controller on a Linux-based system for standalone application.

## References

1. Deb PB, Saha O, Sajan S (2017) Dynamic model analysis of a dc motor in MATLAB. Int J Sci Eng Res 8(3)
2. Yime E, Villa JL, Paez J (2014) Design of a brushed DC motors PID controller for development of low-cost robotic applications. III international congress of engineering mechatronics and automation (CIIMA). https://doi.org/10.1109/ciima.2014.6983471
3. Ulm, Steve F (2020) DC motor position control. Computer and electrical engineering technology & information systems and technology senior design projects

4. Elsrogy WM, Fkirin MA, Hassan MAM (2013) Speed control of DC motor using PID controller based on artificial intelligence techniques. International conference on control, decision and information technologies (CoDIT). doi: https://doi.org/10.1109/codit.2013.6689543
5. Abdulameer, A., Sulaiman, M., Aras, M., Saleem, D.: Tuning methods of PID controller for DC motor speed control. Indonesian J Elec Eng Comput Sci **3**, 343 (2016)
6. Ata AA, Haraz EH, Rizk AEA, Hanna SN (2012) Kinematic analysis of a single link flexible manipulator. IEEE Int Conf Ind Technol. https://doi.org/10.1109/icit.2012.6210045
7. Fareh R (2019) Control of a single flexible link manipulator using fractional active disturbance rejection control. 6th international conference on control, decision and information technologies (CoDIT), pp 900–905. https://doi.org/10.1109/codit.2019.8820708
8. Zhang H, Gao X, Xu G (2019) Research on Improved PD control of flexible manipulator. Chinese control and decision conference (CCDC). https://doi.org/10.1109/ccdc.2019.8832942
9. Zhang C, Yang T, Sun N, Zhang J (2019) A simple control method of single-link flexible manipulators. 2019 3rd international symposium on autonomous systems (ISAS). https://doi.org/10.1109/isass.2019.8757711
10. Yan Z, F., Lai X, S., Meng Q, T., Wu M, Ft.: A Novel Robust Control Method for Motion Control of Uncertain Single-Link Flexible-Joint Manipulator. IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1–8 (2019)
11. Python Control Systems Library Documentation page, https://python-control.readthedocs.io/en/latest/control.html, last accessed 2020/08/06