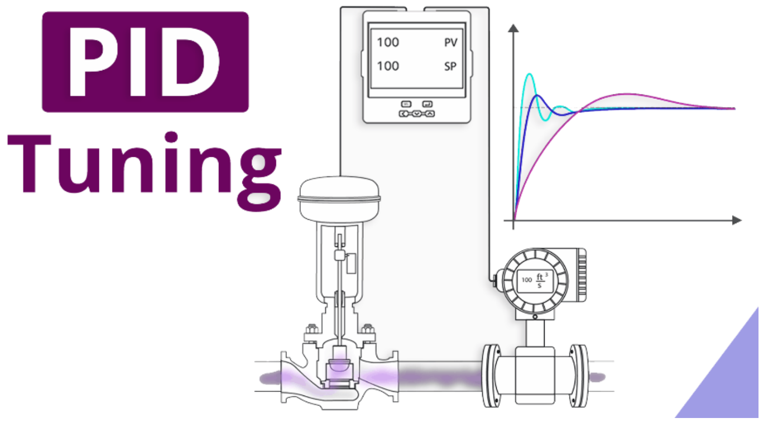


PID Tuning – Frequency Response Method

PRESENTED TO DR. AHMAD OTHMAN

Ahmad Ibrahim Kasim Mohammed | 6302
Ahmad El-Zaki Ahmad Hassan | 6303

1. INTRODUCTION

PID controllers are fundamental in control systems due to their versatility and ease of implementation. Among various tuning methods, the **frequency-response approach** provides a direct link between time-domain performance and frequency-domain stability measures, such as phase margin and crossover frequency.

This report first outlines the standard frequency-domain PID tuning method. Then, it presents a semi-automated implementation based on **second-order system approximations**, enabling control over **maximum overshoot** and **settling time** through systematic calculations.

---

2. CLASSICAL FREQUENCY-RESPONSE PID TUNING

2.1 Design Specifications

```
%% 1) User Specs and Plant Definition
Mp_req   = 0.1;      % Desired max overshoot (fraction, e.g., 0.05 for 5%)
Ts_req   = 6;       % Desired max settling time (s)

% Define plant G(s) here.
G    = tf([1 4], conv([1 1], conv([1 2], [1 6])));

%% 2) Compute damping ratio and exact phase margin for overshoot spec
zeta    = -log(Mp_req) / sqrt(pi^2 + (log(Mp_req))^2); % From second order systems
phi_PM  = atan2(2*zeta, sqrt(sqrt(1+4*zeta^4) - 2*zeta^2)) * (180/pi); % From second order systems
```

Define time-domain specifications:

- Maximum overshoot: `M_p` (e.g., 0.05 for 5%)
- Settling time: `T_s` (in seconds)

These are then converted into frequency-domain targets:

- **Damping ratio** `\zeta` based on overshoot:

$$\zeta = \frac{-\ln(M_p)}{\sqrt{\pi^2 + \left(\ln M_p\right)^2}}$$

- **Required phase margin** `\phi_{PM}` for a second-order approximation:

$$\phi_{PM} = \tan^{-1}\left(\frac{2\zeta}{\sqrt{\sqrt{1+4\zeta^4}-2\zeta^2}}\right)$$

- **Target open-loop phase** at crossover frequency:

$$\angle L(j\omega_c) = -180^\circ + \phi_{PM}$$

2.2 Crossover Frequency Selection

```
%% 2.1) Frequency response of the plant and crossover frequency
wgrid = logspace(-2, 2, 5000);
[~, ph] = bode(G, wgrid); ph = squeeze(ph);
logw    = log10(wgrid);
logw_c  = interp1(ph, logw, targetPhase, 'linear');
wc      = 10^logw_c;
magGwc  = abs(evalfr(G, 1j*wc)); % Evaluates the plant G magnitude at wc in frequency domain.
fprintf('-->Computed crossover frequency: wc = %.3f rad/s\n', wc);
```

The crossover frequency `\omega_c` is chosen where the plant's phase response matches the above target phase. At that frequency, the magnitude of the open-loop transfer function is forced to 1 by adjusting the proportional gain `K_p`.

2.3 Controller Formulas

```
case 'PID'
        Kp       = 1 / magGwc;
        Ti_base = 10 / wc;       Ti = Ti_base * scaleIntTI;
        Ki       = Kp / Ti;
        Td       = 1 / (10*wc);
        Kd       = Kp * Td;
        C        = pid(Kp, Ki, Kd);
        ctrlStr  = sprintf('-->PID (Kp=%.3f, Ki=%.3f, Kd=%.3f, Ti=%.3f)', ...
                        Kp, Ki, Kd, Ti);
```

Based on the plant characteristics and the selected `\omega_c`, controller gains are calculated as follows:

- **P Controller:**

$$K_p = \frac{1}{|G(j\omega_c)|}$$

- **PI Controller:**

$$Cs) = K_p\left(1 + \frac{1}{T_i s}\right), \quad T_i = \frac{10}{\omega_c}$$

- **PID Controller:**

$$C(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right), \quad T_i = \frac{10}{\omega_c}, \quad T_d = \frac{1}{10\omega_c}$$

2.4 Performance Evaluation

```
% Compute performance metrics
   info     = stepinfo(T);
   Tr       = info.RiseTime;          % Rise Time (10%-90%)
   Ts       = info.SettlingTime;
   Mp       = info.Overshoot;
   ess      = abs(1 - dcgain(T));
   [Gm, PM, Wgm, Wpm] = margin(L);
```

The closed-loop system is:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}$$

Performance metrics such as rise time `T_r`, settling time `T_s`, overshoot `M_p`, and steady-state error `e_{ss}` are computed from the step response. Phase margin and gain margin are evaluated from the loop transfer function.

---

3. SEMI-SMART PID TUNING USING SECOND-ORDER APPROXIMATIONS

This MATLAB-based method automates tuning and checks performance specifications through a frequency-domain approach, incorporating:

- Explicit **overshoot and settling time constraints**

- **Second-order approximation** of the system for performance estimation
- Handling of **integrators** in the plant

3.1 Inputs and Setup

- Specify desired `M_p` and `T_s`
- Define the plant `G(s)` using `tf(...)`
- Detect and account for **integrators** by scaling `T_i`

3.2 Frequency Domain Computation

- Calculate `\zeta` and `\phi_{PM}` from `M_p`
- Use interpolation to find the crossover frequency `\omega_c` where the plant phase matches `-180^\circ + \phi_{PM}`
- Evaluate plant magnitude at `\omega_c` to compute controller gains

3.3 Controller Loop

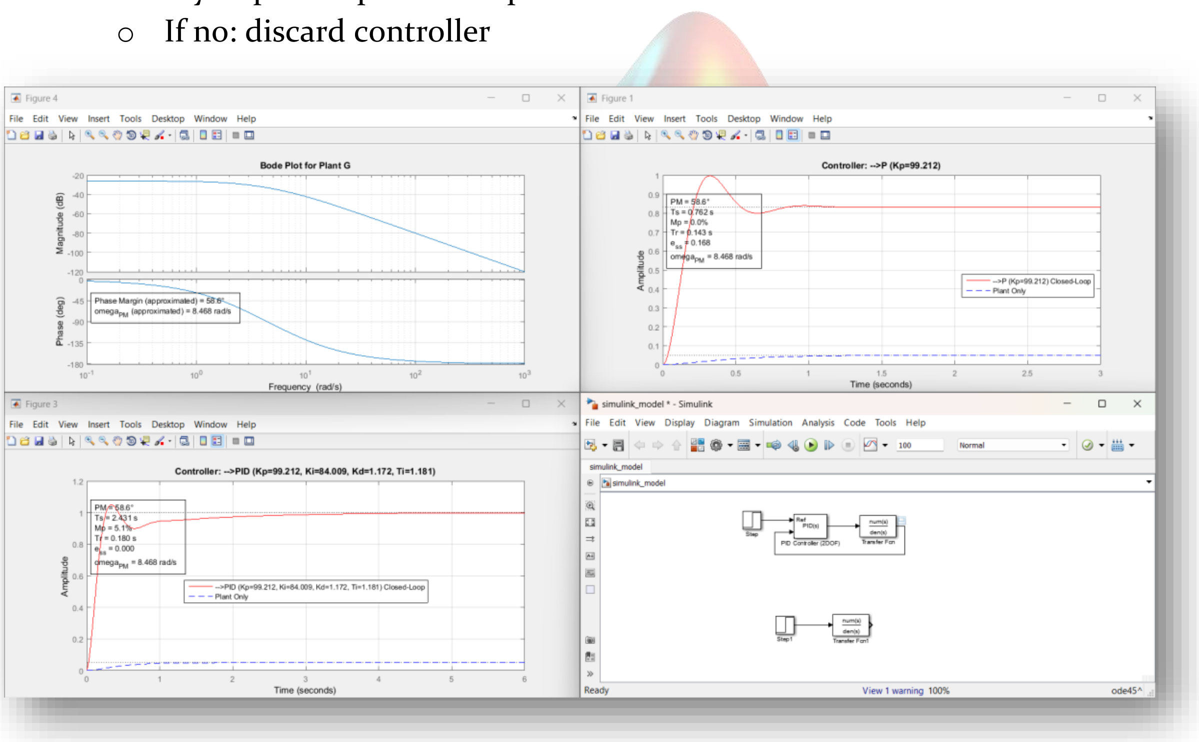
```
%% 3) Loop over controllers
controllers = {'P','PI','PID'};
figCount    = 0;
for i = 1:length(controllers)
    switch controllers{i}
      case 'P'
        Kp  = 1 / magGwc;
        C = Kp;
        ctrlStr = sprintf('-->P (Kp=%.3f)', Kp);

      case 'PI'
        Kp       = 1 / magGwc;
        Ti_base = 10 / wc;       Ti = Ti_base * scaleIntTI;
        Ki       = Kp / Ti;
        C        = tf([Kp*Ti, Kp], [Ti, 0]);
        ctrlStr  = sprintf('-->PI (Kp=%.3f, Ki=%.3f, Ti=%.3f)', Kp, Ki, Ti);

      case 'PID'
        Kp       = 1 / magGwc;
        Ti_base = 10 / wc;       Ti = Ti_base * scaleIntTI;
        Ki       = Kp / Ti;
        Td       = 1 / (10*wc);
        Kd       = Kp * Td;
        C        = pid(Kp, Ki, Kd);
        ctrlStr  = sprintf('-->PID (Kp=%.3f, Ki=%.3f, Kd=%.3f, Ti=%.3f)', ...
                           Kp, Ki, Kd, Ti);
    end
```

For each controller type (P, PI, PID):

- Compute gains using formulas above
- Build open-loop transfer function `L(s) = C(s)G(s)`
- Evaluate closed-loop `T(s)` and extract:
    - `T_s` (settling time)
    - `M_p` (overshoot)
    - `T_r` (rise time)
    - `e_{ss}` (steady-state error)
- Check if specs are met:
    - If yes: plot response and print info
    - If no: discard controller



3.4 Visualizations and Results

- Annotated step response for successful controllers
- Bode plot of the plant with target phase margin and frequency
- Automatic rejection of controllers that do not meet both `M_p` and `T_s`
- Simulink simulation if a controller is found viable.

---

4. SUMMARY OF WORKFLOW

1. Define `M_p` and `T_s`
2. Model `G(s)`
3. Compute `\zeta`, `\phi_{PM}`, and target phase
4. Interpolate plant phase to find `\omega_c`
5. Compute controller gains
6. Simulate step response
7. Evaluate and select only valid controllers
8. Showcase a quick Simulink simulation if a controller is found viable.

---

5. ADVANTAGES OF THE SEMI-SMART METHOD

- Directly links time-domain specifications to controller design
- Automates gain calculation and performance verification
- Adapts to integrating plants by modifying integral time
- Avoids over-tuning or under-tuning by filtering out unsuitable designs
- Easily extendable to more sophisticated methods or additional constraints

---

6. CONCLUSION

This report covers both traditional frequency-domain PID tuning for 2^{nd} order systems and extends it in a practical semi-automated method using second-order system approximations for higher order systems. By integrating time-domain specifications into frequency-based controller design, this approach simplifies the tuning process while maintaining robust performance. The MATLAB implementation provides a flexible framework for applying these principles to a wide range of control systems.