Studi Pengendalian PID untuk Manipulator 1-DoF Berbasis *Forward Kinematics*

Zinadine Zidan Alsyahana (163221014)
Teknik Robotika dan Kecerdasan Buatan
Fakultas Teknologi Maju dan Multidisiplin
zinadine.zidan.alsyahana-2022@ftmm.unair.ac.id

Stevanus Saut Hamonangan Gultom (163221050)
Teknik Robotika dan Kecerdasan Buatan
Fakultas Teknologi Maju dan Multidisiplin
stevanus.saut.hamonangan-2022@ftmm.unair.ac.id

Jeffrey Nobel Martin (163221040)
Teknik Robotika dan Kecerdasan Buatan
Fakultas Teknologi Maju dan Multidisiplin
jeffrey.nobel.martin-2022@ftmm.unair.ac.id

Affa Ndaru Rabbany Wijaya (163221061)
Teknik Robotika dan Kecerdasan Buatan
Fakultas Teknologi Maju dan Multidisiplin
affa.ndaru.rabbany-2022@ftmm.unair.ac.id

Pada studi ini kami mencoba mendesain sebuah *proportional-integral-derivative* (PID) pengontrol manipulator satu *degree of freedom* (DoF). Pengontrol tidak dapat selalu berjalan dengan output yang kami inginkan, maka dari itu kami mendesain PID kontrol dengan melihat model matematika yang bekerja pada manipulator. Model matematika pada studi kami didapatkan dengan menggunakan referensi yaitu membandingkan output dan input pada manipulator. Penghitungan fungsi alih didapatkan melalui pergerakan manipulator dengan metode *forward kinematics*. Dengan pendekatan ini, kami berupaya untuk meningkatkan akurasi dan efisiensi pengendalian manipulator, memastikan bahwa sistem dapat beroperasi dengan stabil dan sesuai dengan kebutuhan yang ditetapkan.

Kata Kunci—Kontrol Proportional-Integral-Derivative (PID); Manipulator; Degree of Freedom (DoF); Forward Kinematics ; Transfer Function.

I. Pendahuluan

Robot manipulator adalah salah satu teknologi yang paling populer dan digunakan secara luas dalam berbagai industri, termasuk kedirgantaraan, manufaktur, industri otomotif, dan aplikasi medis. Manipulator ini dirancang untuk beroperasi dalam lingkungan yang berbahaya, tidak dapat diprediksi, dan tidak ramah bagi manusia, memungkinkan mereka untuk melakukan tugas-tugas yang sulit dan berulang-ulang dengan kecepatan, akurasi, presisi, dan daya tahan yang melebihi kemampuan manusia. Dalam perspektif mekanis, manipulator robot terdiri dari kumpulan tautan serial atau paralel yang dihubungkan dengan sambungan *revolute* dan/atau prismatik antara pangkalan dan bingkai efektor akhir. Seperti tangan manusia, lengan manipulator robot mampu menggerakkan ujung efektornya dari satu tempat ke tempat lain untuk mengambil dan menempatkan benda, mengelas, menulis, melukis, dan melakukan berbagai tugas lainnya. Beberapa peneliti telah mengembangkan berbagai metode untuk mengidentifikasi faktor-faktor yang mempengaruhi kinerja dan akurasi manipulator robot, untuk terus meningkatkan efisiensinya dalam berbagai aplikasi industri [1].

Saat ini, robot mendorong perkembangan industri dengan meningkatkan efisiensi produksi, menunjukkan keunggulannya dibandingkan tenaga manusia. Robot manipulator atau robot lengan adalah jenis yang paling umum digunakan di berbagai sektor industri. Robot manipulator memiliki gerakan yang menyerupai lengan manusia, menjadikannya ideal untuk menggantikan pekerja manusia dalam berbagai tugas. Tantangan utama dalam penggunaan robot ini adalah memahami dengan tepat kinematika robot dan kemudian mengendalikannya untuk menghasilkan gerakan yang tepat. Agar robot dapat melakukan gerakan dengan akurat, perlu dipahami jalur yang akan dilaluinya; oleh karena itu, perencanaan jalur yang baik sangat penting untuk mengontrol robot secara efektif [2].

Untuk menguji kinerja keseluruhan dari manipulator, kami menginstruksikan manipulator untuk melakukan suatu pekerjaan melalui fungsi alih referensi [3] dan kemudian menghitung output dan input manipulator untuk menentukan nilai kesalahan. Metode kontrol *proportional-integral-derivative* (PID) adalah teknik yang menggunakan parameter proporsional, integral, dan derivatif untuk mengatur penguatan sinyal input ke aktuator. Metode ini umum digunakan dalam banyak aplikasi kontrol karena kemampuannya untuk membuat kecepatan manipulator menjadi lebih lambat saat mendekati posisi yang diinginkan, sehingga meningkatkan akurasi. Penggunaan kontroler PID bertujuan untuk mengurangi dan menghilangkan beberapa parameter seperti waktu naik, waktu penyelesaian, *overshoot*, dan kesalahan kondisi tunak, yang semuanya berkontribusi terhadap stabilitas dan ketahanan sistem secara keseluruhan. Selain itu, kontroler PID dikenal karena biayanya yang rendah, kemudahannya dalam implementasi, dan kenyamanannya untuk diintegrasikan ke dalam sistem kontrol yang sudah ada. Hal ini menjadikan metode kontrol PID sebagai pilihan yang sangat efisien dan efektif dalam pengendalian manipulator [4].

Penyetelan PID dengan metode Ziegler-Nichols [5] adalah cara populer untuk menyesuaikan parameter pengontrol PID. Metode ini menawarkan dua pendekatan: metode reaksi langkah dan metode osilasi berkelanjutan. Dalam metode reaksi langkah, kita memberikan sinyal input pada sistem dan mencatat responnya untuk menentukan parameter-parameter

kunci. Sedangkan dalam metode osilasi berkelanjutan, kita meningkatkan pengaturan sampai sistem mulai berosilasi dengan stabil, lalu menggunakan informasi ini untuk menyetel parameter PID. Kelebihan dari metode Ziegler-Nichols adalah kesederhanaannya dan kemampuannya memberikan titik awal yang cepat dan relatif akurat untuk penyetelan PID. Ini sangat berguna karena dapat menghemat waktu dan usaha dalam proses penyetelan awal, meskipun mungkin masih memerlukan penyesuaian tambahan untuk hasil yang optimal.

Dalam studi ini, kami mengusulkan penggunaan pengontrol PID pada manipulator dengan satu *degree of freedom* (DoF) yang disetel menggunakan metode Ziegler-Nichols, dengan pendekatan *forward kinematics* untuk mengoptimalkan pengendalian gerakan manipulator. Pendekatan *forward kinematics* melibatkan seperangkat persamaan yang menggambarkan arah dan posisi efektor akhir (*end-effector*) berdasarkan sudut-sudut sendi yang diberikan. Dengan kata lain, ini memungkinkan kita menghitung posisi dan orientasi efektor akhir dari manipulator menggunakan informasi sudut dari setiap sendi [6-8]. Sebaliknya, pendekatan *inverse kinematics* adalah proses untuk menentukan sudut-sudut sendi yang diperlukan agar efektor akhir mencapai posisi dan orientasi yang diinginkan dalam ruang kartesian. Dalam *inverse kinematics*, posisi dan arah efektor akhir yang dimanipulasi dikonversi dari koordinat kartesian ke koordinat sendi [9,10]. Meskipun kedua pendekatan ini penting dalam manipulator, dalam studi ini, kami memfokuskan pada penggunaan *forward kinematics* untuk memastikan kontrol yang akurat dan efisien pada manipulator satu *degree of freedom* (DoF).

II. Model Matematika Sistem

A. *Persamaan Sistem Motor DC*

Adapun persamaan sistem dari motor DC, yaitu:

$$L_{an}\frac{dI_{an}}{dt} + R_{an}I_{an} + K_{en}\dot{\theta}_{mn} = V_{an}$$

Gambar 2.1 Persamaan Sistem Motor DC

Adapun penjelasan terkait persamaan motor DC, yakni:

L_{a1}: induktansi armatur

R_{a1}: Resistansi armatur

J_{m1}: Inersia motor

B_{m1}: Damping koefisien

K_{e1}: Konstanta EMF

K_{m1}: Konstanta torsi motor

Gambar 2.2 Keterangan Persamaan Motor DC

B. *Transfer Function*

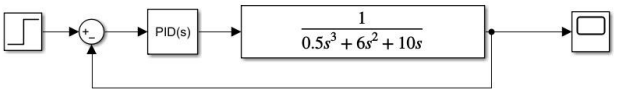
Dari persamaan motor DC, didapatkan fungsi alih sistem, yaitu:

$$G(s) = \frac{\theta_1(s)}{V_{a1}(s)} = \frac{K_m}{(L_{a1}s + R_{a1})(J_{m1}s + B_{m1})K_{e1}K_{m1}}$$

$$G(s) = \frac{1}{0.5s^3 + 6s^2 + 10s}$$

Gambar 2.3 Fungsi alih sistem

Model matematika sistem yang didapatkan dari fungsi alih sistem tersebut dapat dilihat pada gambar berikut.



Gambar 2.4 Diagram Blok Sistem

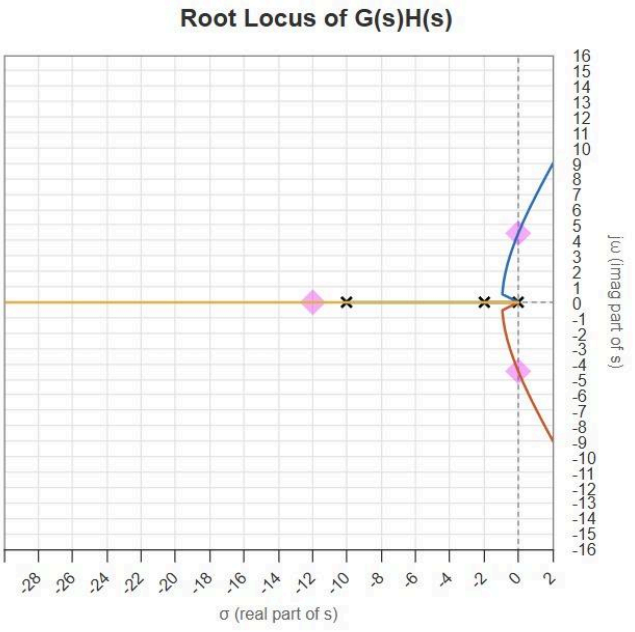
III. Analisis Data

Dari fungsi alih yang didapatkan dari model matematika sistem tersebut, dapat dilakukan tuning PID menggunakan Ziegler-Nichols dengan membuat close-loop function, yaitu:

$$G(s) = \frac{Kp}{Transfer\ Function + Kp}$$

Gambar 3.1. Persamaan Close-Loop Function

Nilai Kp harus berada pada batasan antara stabil dan tidak stabil. Untuk menentukan nilai Kp, maka harus membuat diagram Root Locus terlebih dahulu. Berikut ini adalah diagram root locus yang didapatkan dari Drawing the root locus (swarthmore.edu).



Gambar 3.2 Diagram Root Locus dari Transfer Function

Dari diagram tersebut, nilai Kp yang paling mendekati titik antara kestabilan dan ketidakstabilan adalah 120. Setelah mendapatkan nilai Kp, masukkan nilai Kp tersebut ke dalam persamaan close-loop function, sehingga persamaan yang didapatkan, yaitu:

$$G(s) = \frac{120}{0.5s^3 + 6s^2 + 10s + 120}$$

Gambar 3.3 Persamaan Close-Loop Function Menggunakan Kp = 120

Menggunakan Transfer Function Analysis and Design Tool - Result - (okawa-denshi.jp), didapatkan frekuensi osilasi yaitu:

Oscillation frequency:

$f = 0.711762543417 18[\text{Hz}]$

Gambar 3.4 Frekuensi Osilasi

Pembulatan nilai osilasi frekuensi tersebut menjadi 0.7, sehingga Tuning PID menggunakan Ziegler Nichols dapat dilihat dari gambar 3.5 berikut ini.

ZIEGLER-NICHOLS TUNING

Type of Controller	Kp	Ti	Td
P	0.5 Kcr	∞	0
PI	0.45 Kcr	$\frac{1}{1.2}$ Pcr	0
PID	0.6 Kcr	0.5 Pcr	0.125 Pcr

Kcr = Kp = 120

f ≈ 0.7 Hz

Pcr = $\frac{1}{0.7} \approx 1.4$ s

Sehingga didapatkan:

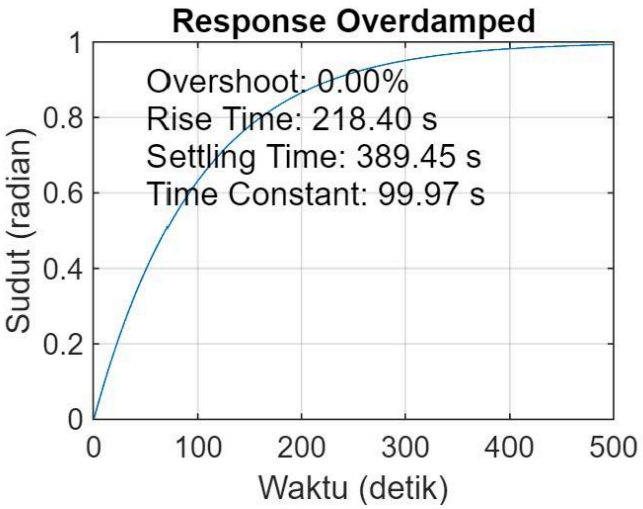
Kp = 72

Ti = 70

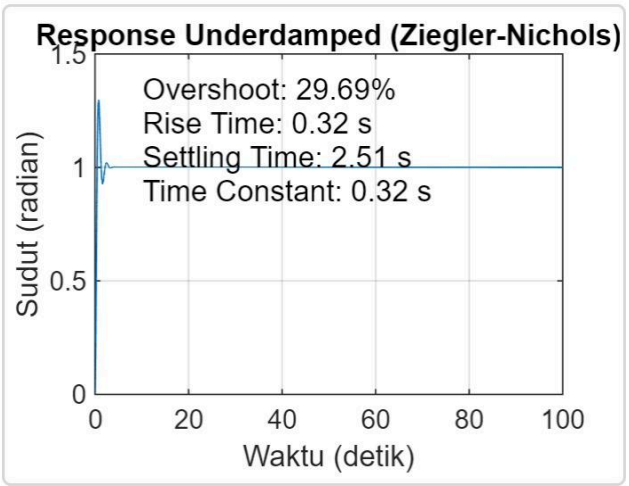
Td = 0.175

Gambar 3.5 Hasil Tuning PID menggunakan Ziegler-Nichols

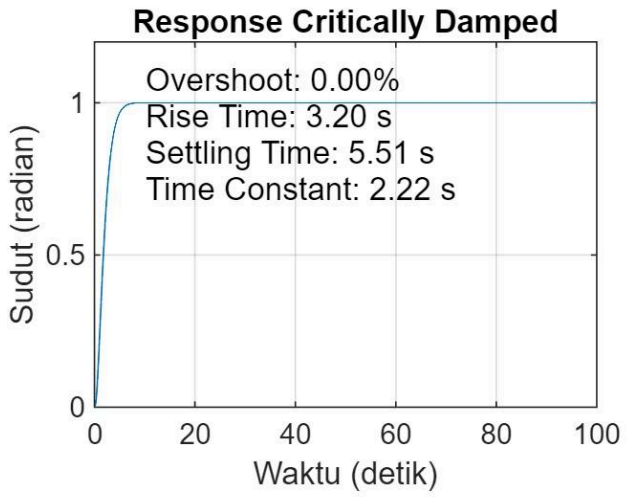
Kemudian, dengan menggunakan MATLAB, didapatkan berbagai nilai respon sistem, seperti pada beberapa gambar berikut.



Gambar 3.6 Respon Sistem Overdamped

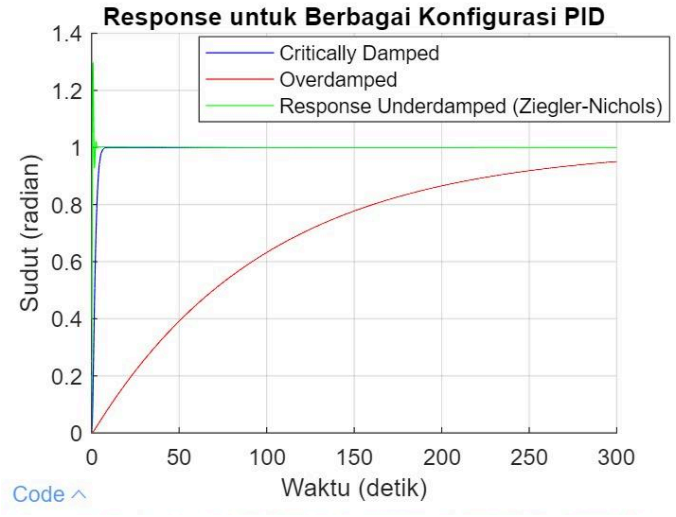


Gambar 3.7 Respon Sistem Underdamped



Gambar 3.8 Respon Sistem Critically Damped

Kemudian, untuk perbandingan dari seluruh data respon sistem, dapat dilihat pada gambar berikut.



Gambar 3.9 Respon Sistem Berbagai Konfigurasi PID

IV. Hasil & Diskusi

Berdasarkan hasil *Tuning Ziegler-Nichols*, respon sistem yang muncul adalah *underdamped*, di mana sistem tersebut berosilasi terlebih dahulu sebelum menjadi stabil.

Berdasarkan perhitungan pada respon sistem *overdamped*, ketika Kp dinaikin secara konstan sebesar 0.1, *rise time* akan berkurang dan menuju ke *critically damped*, lalu ketika Kd dinaikin secara perlahan, *rise time* bertambah lama, namun dengan konsekuensi sudut mulai dari lengan tidak tepat pada titik awal.

Berdasarkan perhitungan pada respon sistem *critically damped*, ketika Kp naik melebihi nilai 4.9 terjadi *overshoot* sebesar 0.01% dan ketika Kd ditambahkan, *rise time* akan bertambah dan titik mulainya tidak pada 0.

Setelah melalui *tuning* PID menggunakan metode *Ziegler-Nichols Tuning*, dapat dilihat berdasarkan grafik yang didapatkan, maka pergerakan manipulator di atas akan mengalami osilasi lalu segera memperbaiki ke pada posisi yang diinginkan.

V. Kesimpulan

Setelah melakukan percobaan tuning PID menggunakan *Ziegler-Nichols Tuning* dan percobaan menggunakan MATLAB, dapat disimpulkan bahwa hasil tuningan menggunakan metode *Ziegler-Nichols Tuning* membutuhkan *Rise Time*, *Settling Time*, dan *Time Constant* yang jauh lebih cepat dibandingkan metode percobaan menggunakan MATLAB. Akan tetapi, *overshoot* pada respon sistem *underdamped* memiliki nilai 29.69%, berbeda dengan respon sistem lainnya. Hal ini disebabkan karena respon sistem underdamped berosilasi terlebih dahulu sebelum stabil, berbeda dengan respon sistem lainnya. Perlu diketahui bahwa metode *Ziegler-Nichols Tuning* ini hanya berdampak pada respon sistem *underdamped*. Dapat dilihat dari grafik respon sistem *underdamped* yang berosilasi terlebih dahulu, kemudian menjadi stabil. Setelah melalui *tuning* PID ini, maka pergerakan manipulator akan bergerak perlahan dari yang tidak stabil menjadi stabil.

Daftar Pustaka

[1] I. Agustian, N. Daratha, R. Faurina, A. Suandi, dan S. Sulistyaningsih, "Robot Manipulator Control with Inverse Kinematics PD-Pseudoinverse Jacobian and Forward Kinematics Denavit Hartenberg," *Jurnal Elektronika dan Telekomunikasi*, vol. 21, no. 1, p. 8, 2021, doi: https://doi.org/10.14203/jet.v21.8-18.

[2] T. H. Noventino, M. R. Rosa, dan A. Z. Fuadi, "PID Control Design and Kinematic Modelling of 3-DoF Robot Manipulator," *2022 International Conference on Electrical Engineering, Computer and Information Technology (ICEECIT)*, Jember, Indonesia, no. pp. 88-94, 2022, doi: https://doi.org/10.1109/iceecit55908.2022.10030325.

[3] B. Hekimoglu, "Optimal Tuning of Fractional Order PID Controller for DC Motor Speed Control via Chaotic Atom Search Optimization Algorithm," *IEEE Access*, vol. 7, pp. 38100–38114, 2019, doi: https://doi.org/10.1109/access.2019.2905961.

[4] X. Wei, G. Jin, W. Zhan, B. Chang, dan Y. Song, "Dynamics Analysis of the Rigid-Flexible Coupling Lifting Comprehensive Mechanism for a Rotary Dobby," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–14, 2019, doi: https://doi.org/10.1155/2019/4896162.

[5] J. G. Ziegler dan N. B. Nichols, "Optimum Settings for Automatic Controllers," *Journal of Fluids Engineering*, vol. 64, no. 8, pp. 759–765, 1942, doi: https://doi.org/10.1115/1.4019264.

[6] A. T. Jawad, N. S. Ali, A. N. Abdullah, dan N. H. Alwash, "Design of Adaptive Controller for Robot Arm Manipulator Based on ANN with Optimized PID by IWO Algorithm," *2021 International Conference on Advanced Computer Applications (ACA)*, Maysan, Iraq, 2021, doi: https://doi.org/10.1109/aca52198.2021.9626781.

[7] M. Heidar, A. Zalaghi, S. G. R. A. G. Atigh, dan Z. Torkani, "Design of PID and Fuzzy-PID Controllers for Agile Eye Spherical Parallel Manipulator," *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, Tehran, Iran, pp. 113–117, 2019, doi: https://doi.org/10.1109/kbei.2019.8735095.

[8] S. Savin, O. Balakhnov, dan A. Klimchik, "Energy-based local forward and inverse kinematics methods for tensegrity robots," *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan, pp. 280–284, 2020, doi: https://doi.org/10.1109/irc.2020.00051.

[9] Y. Cao, W. Wang, L. Ma, dan X. Wang, "Inverse Kinematics Solution of Redundant Degree of Freedom Robot Based on Improved Quantum Particle Swarm Optimization," *2021 IEEE 7th International Conference on Control Science and Systems Engineering (ICCSSE)*, Qingdao, China, pp. 68–72, 2021, doi: https://doi.org/10.1109/iccsse52761.2021.9545199.

[10] A. A. Hassan, M. El-Habrouk, dan S. Deghedie, "Inverse Kinematics of Redundant Manipulators Formulated as Quadratic Programming Optimization Problem Solved Using Recurrent Neural Networks: A Review," *Robotica*, vol. 38, no. 8, pp. 1495–1512, 2019, doi: https://doi.org/10.1017/s0263574719001590.

Lampiran

Lampiran 1: Kode MATLAB Respon Sistem Underdamped

```
% Parameter dari manipulator
L = 0.5; % Panjang lengan dalam meter
m = 6; % Berat dalam kilogram

% Momen inersia untuk batang dengan distribusi massa seragam (asumsi sederhana)
I = (1/3) * m * L^2; % Momen inersia

% Transfer function dari motor DC yang tertera pada gambar
num_motor = 1;
den_motor = [I, 6, 10, 0];
motor_tf = tf(num_motor, den_motor);
% Menampilkan transfer function motor
disp('Transfer function dari motor DC:');
disp(motor_tf);

% Menentukan parameter PID berdasarkan metode Ziegler-Nichols
Kp = 72;
Ti = 70;
Td = 0.175;

% Membuat PID controller
PID_controller = pid(Kp, Kp/Ti, Kp*Td);

% Menampilkan parameter PID
disp('Parameter PID yang dituning menggunakan Ziegler-Nichols:');
disp(['Kp: ', num2str(Kp)]);
disp(['Ti: ', num2str(Ti)]);
disp(['Td: ', num2str(Td)]);

% Simulasi sistem dengan PID controller
sys_cl_pid = feedback(PID_controller * motor_tf, 1);

% Setpoint 1 radian
setpoint = 1; % dalam radian

% Membuat step input dengan nilai setpoint 1 radian
time = 0:0.01:100; % waktu simulasi dari 0 hingga 100 detik
input_signal = setpoint * ones(size(time));

% Simulasi sistem
[response, t] = step(setpoint * sys_cl_pid, time);

% Mengambil informasi step response
info = stepinfo(response, t, setpoint);
overshoot = info.Overshoot;
riseTime = info.RiseTime;
settlingTime = info.SettlingTime;

% Menghitung time constant
% Asumsi time constant adalah waktu yang dibutuhkan untuk mencapai 63.2% dari nilai akhir
final_value = setpoint;
time_constant_index = find(response >= 0.632 * final_value, 1);
time_constant = t(time_constant_index);

% Plot hasil simulasi
figure;
plot(t, response);
xlabel('Waktu (detik)');
ylabel('Sudut (radian)');
title('Response Underdamped (Ziegler-Nichols)');
grid on;

% Menampilkan overshoot, rise time, settling time, dan time constant pada grafik
text(0.1, 0.9, sprintf('Overshoot: %.2f%%', overshoot), 'Units', 'normalized');
text(0.1, 0.8, sprintf('Rise Time: %.2f s', riseTime), 'Units', 'normalized');
text(0.1, 0.7, sprintf('Settling Time: %.2f s', settlingTime), 'Units', 'normalized');
text(0.1, 0.6, sprintf('Time Constant: %.2f s', time_constant), 'Units', 'normalized');
```

Lampiran 2: Kode MATLAB Respon Sistem Critically Damped

```
% Parameter dari manipulator
L = 0.5; % Panjang lengan dalam meter
m = 6; % Berat dalam kilogram

% Momen inersia untuk batang dengan distribusi massa seragam (asumsi sederhana)
I = (1/3) * m * L^2; % Momen inersia

% Transfer function dari motor DC yang tertera pada gambar
num_motor = 1;
den_motor = [I, 6, 10, 0];
motor_tf = tf(num_motor, den_motor);

% Menampilkan transfer function motor
disp('Transfer function dari motor DC:');
disp(motor_tf);
```

```
% Menentukan parameter PID
Kp = 4.9;
Ki = 0;
Kd = 0;

% Membuat PID controller
PID_controller = pid(Kp, Ki, Kd);

% Menampilkan parameter PID
disp(['Kp: ', num2str(Kp)]);
disp(['Ki: ', num2str(Ki)]);
disp(['Kd: ', num2str(Kd)]);

% Simulasi sistem dengan PID controller
sys_cl_pid = feedback(PID_controller * motor_tf, 1);

% Setpoint 1 radian
setpoint = 1; % dalam radian

% Membuat step input dengan nilai setpoint 1 radian
time = 0:0.01:100; % waktu simulasi dari 0 hingga 10 detik
input_signal = setpoint * ones(size(time));

% Simulasi sistem
[response, t] = step(setpoint * sys_cl_pid, time);

% Mengambil informasi step response
info = stepinfo(response, t, setpoint);
overshoot = info.Overshoot;
riseTime = info.RiseTime;
settlingTime = info.SettlingTime;

% Menghitung time constant
% Asumsi time constant adalah waktu yang dibutuhkan untuk mencapai 63.2% dari nilai akhir
final_value = setpoint;
time_constant_index = find(response >= 0.632 * final_value, 1);
time_constant = t(time_constant_index);

% Plot hasil simulasi
figure;
plot(t, response);
xlabel('Waktu (detik)');
ylabel('Sudut (radian)');
title('Response Critically Damped');
grid on;

% Menampilkan overshoot, rise time, settling time, dan time constant pada grafik
text(0.1, 0.9, sprintf('Overshoot: %.2f%%', overshoot), 'Units', 'normalized');
text(0.1, 0.8, sprintf('Rise Time: %.2f s', riseTime), 'Units', 'normalized');
text(0.1, 0.7, sprintf('Settling Time: %.2f s', settlingTime), 'Units', 'normalized');
text(0.1, 0.6, sprintf('Time Constant: %.2f s', time_constant), 'Units', 'normalized');
```

Lampiran 3: Kode MATLAB Respon Sistem Overdamped

```
% Parameter dari manipulator
L = 0.5; % Panjang lengan dalam meter
m = 6; % Berat dalam kilogram

% Momen inersia untuk batang dengan distribusi massa seragam (asumsi sederhana)
I = (1/3) * m * L^2; % Momen inersia

% Transfer function dari motor DC yang tertera pada gambar
num_motor = 1;
den_motor = [I, 6, 10, 0];
motor_tf = tf(num_motor, den_motor);

% Menampilkan transfer function motor
disp('Transfer function dari motor DC:');
disp(motor_tf);


% Menentukan parameter PID
Kp = 0.1;
Ki = 0;
Kd = 0;

% Membuat PID controller
PID_controller = pid(Kp, Ki, Kd);

% Menampilkan parameter PID
disp('Parameter PID yang dituning menggunakan Ziegler-Nichols:');
disp(['Kp: ', num2str(Kp)]);
disp(['Ki: ', num2str(Ki)]);
disp(['Kd: ', num2str(Kd)]);

% Simulasi sistem dengan PID controller
sys_cl_pid = feedback(PID_controller * motor_tf, 1);

% Setpoint 1 radian
setpoint = 1; % dalam radian

% Membuat step input dengan nilai setpoint 1 radian
time = 0:0.01:500; % waktu simulasi dari 0 hingga 100 detik
input_signal = setpoint * ones(size(time));

% Simulasi sistem
```

```
[response, t] = step(setpoint * sys_cl_pid, time);

% Mengambil informasi step response
info = stepinfo(response, t, setpoint);
overshoot = info.Overshoot;
riseTime = info.RiseTime;
settlingTime = info.SettlingTime;

% Menghitung time constant
% Asumsi time constant adalah waktu yang dibutuhkan untuk mencapai 63.2% dari nilai akhir
final_value = setpoint;
time_constant_index = find(response >= 0.632 * final_value, 1);
time_constant = t(time_constant_index);

% Plot hasil simulasi
figure;
plot(t, response);
xlabel('Waktu (detik)');
ylabel('Sudut (radian)');
title('Response Overdamped');
grid on;

% Menampilkan overshoot, rise time, settling time, dan time constant pada grafik
text(0.1, 0.9, sprintf('Overshoot: %.2f%%', overshoot), 'Units', 'normalized');
text(0.1, 0.8, sprintf('Rise Time: %.2f s', riseTime), 'Units', 'normalized');
text(0.1, 0.7, sprintf('Settling Time: %.2f s', settlingTime), 'Units', 'normalized');
text(0.1, 0.6, sprintf('Time Constant: %.2f s', time_constant), 'Units', 'normalized');
```

Lampiran 4: Kode MATLAB Respon Sistem Berbagai Konfigurasi PID

```
% Parameter dari manipulator
L = 0.5; % Panjang lengan dalam meter
m = 6; % Berat dalam kilogram

% Momen inersia untuk batang dengan distribusi massa seragam (asumsi sederhana)
I = (1/3) * m * L^2; % Momen inersia

% Parameter dari motor DC (termasuk konstanta untuk menyertakan efek panjang dan berat)
num_motor = 1;
den_motor = [I, 6, 10, 0]; % Menggunakan momen inersia dalam denominator

motor_tf = tf(num_motor, den_motor);

% Menampilkan transfer function motor
disp('Transfer function dari motor DC dengan panjang dan berat manipulator:');
disp(motor_tf);

% Variasi parameter PID untuk berbagai kondisi
PID_configs = {
    4.9, 0, 0, 'Critically Damped'; % Kp, Ki, Kd, Title
    0.1, 0, 0, 'Overdamped';
          72.0, 1.02857, 12.6, 'Response Underdamped (Ziegler-Nichols)'
};

% Warna untuk plot
colors = ['b', 'r', 'g', 'm'];

figure;
hold on;

for i = 1:size(PID_configs, 1)
    % Menentukan parameter PID
    Kp = PID_configs{i, 1};
    Ki = PID_configs{i, 2};
    Kd = PID_configs{i, 3};

    % Membuat PID controller
    PID_controller = pid(Kp, Ki, Kd);

    % Menampilkan parameter PID
     disp(['Parameter PID untuk konfigurasi ', PID_configs{i, 4}, ':']);
    disp(['Kp: ', num2str(Kp)]);
    disp(['Ki: ', num2str(Ki)]);
    disp(['Kd: ', num2str(Kd)]);

    % Simulasi sistem dengan PID controller
    sys_cl_pid = feedback(PID_controller * motor_tf, 1);

    % Setpoint 1 radian
    setpoint = 1; % dalam radian

    % Membuat step input dengan nilai setpoint 1 radian
     time = 0:0.01:300; % waktu simulasi dari 0 hingga 300 detik
    input_signal = setpoint * ones(size(time));

    % Simulasi sistem
    [response, t] = step(setpoint * sys_cl_pid, time);

     % Plot hasil simulasi dengan warna berbeda untuk setiap konfigurasi
          plot(t, response, 'Color', colors(i), 'DisplayName', PID_configs{i, 4});
end
```

```
xlabel('Waktu (detik)');
ylabel('Sudut (radian)');
title('Response untuk Berbagai Konfigurasi PID');
legend show;
```

```
grid on;
hold off;
```