**EE6225 Assignment-1**

Wu Tianwei

Matriculation No. G2101446F

e-mail: WU0008EI@e.ntu.edu.sg

The process transfer function is described as

**1.** First order plus time delay model using classic technics: two points method, log method and area method.

**Solution:**

Using Unit Step Function as input.

Matlab Code:

|  |
| --- |
| % Plot the unit step function  syms x;  figure(1);  fplot(heaviside(x));  xlim([-1, 1]);  ylim([-0.2, 1.2]);  title('Unit Step Function') |

图表

描述已自动生成

Plotting the step response of original transfer function.

Matlab Code:

|  |
| --- |
| % Plot the step response of original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  figure(2);  step(G0); |

图表, 折线图

描述已自动生成

① Two points method

Matlab Code:

|  |
| --- |
| % Find t1 & t2  steadyStateY = 0.1875;  y1 = steadyStateY \* 0.284;  y2 = steadyStateY \* 0.632;  t = 0:0.001:20;  Y0 = step(G0, t);  figure(3);  plot(t, Y0);  title('step response');  xlabel('Time (seconds)');  ylabel('Amplitude');    % Define the transfer function of FOPTD model  t1 = 3.406;  t2 = 4.269;  A = 1;  K = steadyStateY / A;  T = 1.5 \* (t2 - t1);  L = 0.5 \* (3 \* t1 - t2);  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Compare two transfer functions  figure(4);  step(G0, Gp, 20);  legend('Original','Frist order plus time delay model');  title('Two Points Method');  subtitle({'Frist order plus time delay model parameters', ['A = ', num2str(A), ', K =' num2str(K), ', T =', num2str(T), ', L =', num2str(L)]}); |

图形用户界面

低可信度描述已自动生成

The steady state value . In order to use two points method, the points of 28.4% and 63.2% of need to be found. As shown in the plot, the 28.4% point is and the 63.2% point is .

So, the parameters of FOPTD model can be computed by using the two points. According to the two points, . Thus,

The steady-state gain can be determined by the ultimate value of and the magnitude of the given input step function .

Therefore, the FOPTD model is calculated as

图片包含 图示

描述已自动生成

② Log method

Matlab code:

|  |
| --- |
| % Sampling on the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  t = 2 : 0.1 : 6;  y = step(G0, t);    % Transformation of the step-response data against time t  k = size(y);  yln = [];  tln = [];  for n = 1 : 1 : k  if y(n, 1) > 0 && 0.1875 - y(n, 1) >= 0  yln = [yln, log((0.1875 - y(n, 1)) / 0.1875)];  tln = [tln, t(1, n)];  end  end    % Do linear fitting  h = polyfit(tln', yln', 1);  figure(1);  plot(tln, yln, '\*', tln, polyval(h, tln));  grid on;  title('Linear Fitting');  subtitle(['y = ', num2str(h(1,1)), 'x + ', num2str(h(1,2))]);  xlabel('t', 'FontWeight', 'bold');  ylabel('ln((y\_{¡Þ} - y) / y\_{¡Þ})', 'FontWeight', 'bold');    % Define the transfer function of FOPTD model  steadyStateY = 0.1875;  A = 1;  K = steadyStateY / A;  T = 1 / -h(1, 1);  L = h(1, 2) \* T;  figure(2);  Gp = K \* exp(- L \* s) / (T \* s + 1);  step(G0, Gp);  legend('Original','Frist order plus time delay model');  title('Log Method');  subtitle({'Frist order plus time delay model parameters', ['A = ', num2str(A), ', K =' num2str(K), ', T =', num2str(T), ', L =', num2str(L)]}); |

The sampling starts from 2s to 6s with sampling interval 0.1s. So, the number of samples is 41.

To use the log method, the step-response data needs to be rearranged. After the transformation, a straight line can be determined by applying linear fitting, which is

According to the log method, the transformation step-response data against time is defined as

图表, 折线图, 散点图

描述已自动生成

The slope of the straight line is -0.89496. Thus,

Therefore, the FOPTD model is calculated as

折线图

低可信度描述已自动生成

③ Area method

Matlab Code:

|  |
| --- |
| % Sampling on the original step response  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  figure(1);  step(G0);  Ts = 0.01; % sampling interval  start\_time\_0 = 0;  end\_time\_0 = 12;  s\_num\_0 = (end\_time\_0 - start\_time\_0) / Ts; % total number of samples  t0 = start\_time\_0 : Ts : end\_time\_0 - Ts;  y0 = step(G0, t0);    % Calculate A0  steadyStateY = 0.1875;  A0 = Ts / 2 \* (steadyStateY - y0(1));  for i = 2 : s\_num\_0 - 1  A0 = A0 + Ts \* (steadyStateY - y0(i));  end  A0 = A0 + Ts / 2 \* (steadyStateY - y0(s\_num\_0));    % Calculate A1  A = 1;  K = steadyStateY / A;  Tar = A0 / K;  start\_time\_1 = 0;  end\_time\_1 = floor(Tar \* (1 / Ts)) / 100;  s\_num\_1 = (end\_time\_1 - start\_time\_1) / Ts;  t1 = start\_time\_1 : Ts : end\_time\_1 - Ts;  y1 = step(G0, t1);  A1 = Ts / 2 \* y1(1);  for i = 2 : s\_num\_1 - 1  A1 = A1 + Ts \* y1(i);  end  A1 = A1 + Ts / 2 \* y1(s\_num\_1);    % Define the transfer function of FOPTD model  T = (exp(1) \* A1) / K;  L = (A0 / K) - ((exp(1) \* A1) / K);  Gp = K \* exp(- L \* s) / (T \* s + 1);  figure(2);  step(G0, Gp);  legend('Original','Frist order plus time delay model');  title('Area Method');  subtitle({'Frist order plus time delay model parameters', ['A = ', num2str(A), ', K =' num2str(K), ', T =', num2str(T), ', L =', num2str(L)]}); |

To use are method, and need to be computed. According to the area method,

To have accurate , sampling should end when the process well enters the new steady state again. From the plot of original step response, the time when the process reaches a new steady state is about 12s. So, the sampling starts from 0s to 12s with sampling interval 0.01s. Using the formula above,

Then,

Like , can be computed by the integral.

Thus, and can be determined by and calculated above.

Finally, the FOPTD model is calculated as

图表

中度可信度描述已自动生成

**2.** First order plus time delay model using least squares method under open loop test in time domain.

**Solution:**

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));    % Sampling  start\_time = 2;  end\_time = 32;  Ts = 0.01;  s\_num = (end\_time - start\_time) / Ts;  t = start\_time : Ts : end\_time - Ts;  y = step(G0, t);    % Calculate psi and gamma  A = 1;  psi = zeros(s\_num, 3);  psi(1, 1) = Ts / 2 \* y(1);  for i =2 : s\_num - 1  psi(i, 1) = psi(i - 1, 1) + Ts\*y(i);  end  psi(s\_num, 1) = psi(s\_num - 1, 1) + Ts / 2 \* y(s\_num);  psi(:, 1) = -psi(:, 1);  psi(:, 2) = -A;  psi(:, 3) = A \* t;  gamma = y;    % Using least squares method to calculate the parameters  theta = ((psi' \* psi)^-1) \* psi' \* gamma;  a1 = theta(1, 1);  b1 = theta(3, 1);  L = theta(2, 1) / theta(3, 1);    Gp = b1 \* exp(- L \* s) / (s + a1);  step(G0, Gp);  legend('Original','Frist order plus time delay model');  title('Least Squares Method - Time');  subtitle({'Frist order plus time delay model parameters', ['A = ', num2str(A), ', K =' num2str(b1 / a1), ', T =', num2str(1 / a1), ', L =', num2str(L)]}); |

To get more accurate result, the sampling time starts from 2s to 32s with sampling interval 0.01s. So, the total number of samples is 3000.

After applying the least squares method in time domain,

Then, the model parameters can be computed by

Finally, the FOPTD model is calculated as

图表

描述已自动生成

**3.** First order plus time delay under open loop test using least squares method in frequency domain.

**Solution:**

Firstly, do sampling on the original step response function. The sampling time starts from 0s to 20s with sample interval 0.1s. So, the total number of samples is 200.

For input ,

Using the numerical integration method, such as the trapezoidal numerical integration, we have

where

Then recursive solution is used to calculate the parameters.

For initialization, and . For , and can be determined by the following recursion equation

The FOPTD model is defined as

Let , for frequency , the magnitude and phase of are given as

Rewrite the magnitude equation into a matric form

where

Thus, can be computed by applying least squares method

Then, and can be recovered from

The phase equation also can be rewritten into a matric form

where

Then, can be computed by using least squares method

Finally, the FOPTD model is calculated as

The step response of FOPTD model is shown in the following plot.

图片包含 图示

描述已自动生成

**4.** Using relay feedback to generate sustained oscillation and using the available information to calculate the parameters of first order plus time delay model.

**Solution:**

Firstly, to generate sustained oscillation, a relay feedback model needs to be constructed in Simulink.

The model is shown in the following image.

图示

描述已自动生成

Relay testing result:

图表

描述已自动生成

From the output, the following parameters can be determined:

, , , ,

Matlab Code:

|  |
| --- |
| % Run the simulink model  sim('relay\_feedback\_simulink');  plot(out);    % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));    % Determine parameters from the output plot  L = 2.4415;  Pu = 7.9145;  Kp = 0.1875;  h = 1;  a = 0.1815;    Wu = 2 \* pi / Pu;  Ku = 4 \* h / (a \* pi);  T = sqrt((Kp \* Ku)^2 - 1) / Wu;    Gp = Kp \* exp(- L \* s) / (T \* s + 1);  step(G0, Gp);  legend('Original','Frist order plus time delay model');  title('Relay Feedback');  subtitle({'Frist order plus time delay model parameters', ['K =' num2str(K), ', T =', num2str(T), ', L =', num2str(L)]}); |

Therefore, the FOPTD model is calculated as

图示

低可信度描述已自动生成

**5.** For each method, plot Nyquist chart for both the original transfer function and the identified transfer function to compare the identification results.

**Solution:**

1) Two points method

Using recursive solution:

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  K = 0.1875;  T = 1.2945;  L = 2.9745;  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Sampling on the original step response  start\_time = 0;  end\_time = 20;  Ts = 0.1;  s\_num = (end\_time - start\_time) / Ts;  t\_sample = start\_time : Ts : end\_time - Ts;  [y0, t0] = step(G0, t\_sample);  [yp, tp] = step(Gp, t\_sample);    % Define the trapezoidal integration  syms w;  g0(w) = (y0(s\_num) + w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* sin(w \* t0))) + 1i \* w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* cos(w \* t0));  gp(w) = (yp(s\_num) + w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* sin(w \* tp))) + 1i \* w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* cos(w \* tp));    % Set initial values  M = 10;  W0 = zeros(1, M);  Wp = zeros(1, M);  fai0 = zeros(1, M);  faip = zeros(1, M);    W0(1) = 0;  W0(2) = 10^-3;  Wp(1) = 0;  Wp(2) = 10^-3;  fai0(1) = 0;  fai0(2) = angle(g0(W0(2)));  faip(1) = 0;  faip(2) = angle(gp(Wp(2)));    % Recursive solution  for i = 3 : M  W0(i) = W0(i - 1) - ((i - 1) \* pi / (M - 1) + fai0(i - 1)) \* (W0(i - 1) - W0(i - 2)) / (fai0(i - 1) - fai0(i - 2));  fai0(i) = angle(g0(W0(i)));  Wp(i) = Wp(i - 1) - ((i - 1) \* pi / (M - 1) + faip(i - 1)) \* (Wp(i - 1) - Wp(i - 2)) / (faip(i - 1) - faip(i - 2));  faip(i) = angle(gp(Wp(i)));  end    real\_part0 = zeros(1, M);  imag\_part0 = zeros(1, M);  real\_partp = zeros(1, M);  imag\_partp = zeros(1, M);  for n = 1 : M  real\_part0(n) = real(g0(W0(n)));  imag\_part0(n) = imag(g0(W0(n)));  real\_partp(n) = real(gp(Wp(n)));  imag\_partp(n) = imag(gp(Wp(n)));  end  plot(real\_part0, imag\_part0, 'b-o');  hold on;  plot(real\_partp, imag\_partp, 'r--\*');  title('Nyquist chart');  xlabel('Real Axis');  ylabel('Imaginary Axis');  legend('Original', 'Two points method'); |

图表, 折线图

描述已自动生成

2) Log method

Using recursive solution:

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  K = 0.1875;  T = 1.1174;  L = 2.7877;  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Sampling on the original step response  start\_time = 0;  end\_time = 20;  Ts = 0.1;  s\_num = (end\_time - start\_time) / Ts;  t\_sample = start\_time : Ts : end\_time - Ts;  [y0, t0] = step(G0, t\_sample);  [yp, tp] = step(Gp, t\_sample);    % Define the trapezoidal integration  syms w;  g0(w) = (y0(s\_num) + w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* sin(w \* t0))) + 1i \* w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* cos(w \* t0));  gp(w) = (yp(s\_num) + w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* sin(w \* tp))) + 1i \* w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* cos(w \* tp));    % Set initial values  M = 10;  W0 = zeros(1, M);  Wp = zeros(1, M);  fai0 = zeros(1, M);  faip = zeros(1, M);    W0(1) = 0;  W0(2) = 10^-3;  Wp(1) = 0;  Wp(2) = 10^-3;  fai0(1) = 0;  fai0(2) = angle(g0(W0(2)));  faip(1) = 0;  faip(2) = angle(gp(Wp(2)));    % Recursive solution  for i = 3 : M  W0(i) = W0(i - 1) - ((i - 1) \* pi / (M - 1) + fai0(i - 1)) \* (W0(i - 1) - W0(i - 2)) / (fai0(i - 1) - fai0(i - 2));  fai0(i) = angle(g0(W0(i)));  Wp(i) = Wp(i - 1) - ((i - 1) \* pi / (M - 1) + faip(i - 1)) \* (Wp(i - 1) - Wp(i - 2)) / (faip(i - 1) - faip(i - 2));  faip(i) = angle(gp(Wp(i)));  end    real\_part0 = zeros(1, M);  imag\_part0 = zeros(1, M);  real\_partp = zeros(1, M);  imag\_partp = zeros(1, M);  for n = 1 : M  real\_part0(n) = real(g0(W0(n)));  imag\_part0(n) = imag(g0(W0(n)));  real\_partp(n) = real(gp(Wp(n)));  imag\_partp(n) = imag(gp(Wp(n)));  end  plot(real\_part0, imag\_part0, 'b-o');  grid on;  hold on;  plot(real\_partp, imag\_partp, 'r--\*');  title('Nyquist chart');  xlabel('Real Axis');  ylabel('Imaginary Axis');  legend('Original', 'Log method'); |

图表, 折线图

描述已自动生成

3) Area method

Using recursive solution:

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  K = 0.1875;  T = 0.9496;  L = 3.0501;  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Sampling on the original step response  start\_time = 0;  end\_time = 20;  Ts = 0.1;  s\_num = (end\_time - start\_time) / Ts;  t\_sample = start\_time : Ts : end\_time - Ts;  [y0, t0] = step(G0, t\_sample);  [yp, tp] = step(Gp, t\_sample);    % Define the trapezoidal integration  syms w;  g0(w) = (y0(s\_num) + w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* sin(w \* t0))) + 1i \* w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* cos(w \* t0));  gp(w) = (yp(s\_num) + w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* sin(w \* tp))) + 1i \* w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* cos(w \* tp));    % Set initial values  M = 10;  W0 = zeros(1, M);  Wp = zeros(1, M);  fai0 = zeros(1, M);  faip = zeros(1, M);    W0(1) = 0;  W0(2) = 10^-3;  Wp(1) = 0;  Wp(2) = 10^-3;  fai0(1) = 0;  fai0(2) = angle(g0(W0(2)));  faip(1) = 0;  faip(2) = angle(gp(Wp(2)));    % Recursive solution  for i = 3 : M  W0(i) = W0(i - 1) - ((i - 1) \* pi / (M - 1) + fai0(i - 1)) \* (W0(i - 1) - W0(i - 2)) / (fai0(i - 1) - fai0(i - 2));  fai0(i) = angle(g0(W0(i)));  Wp(i) = Wp(i - 1) - ((i - 1) \* pi / (M - 1) + faip(i - 1)) \* (Wp(i - 1) - Wp(i - 2)) / (faip(i - 1) - faip(i - 2));  faip(i) = angle(gp(Wp(i)));  end    real\_part0 = zeros(1, M);  imag\_part0 = zeros(1, M);  real\_partp = zeros(1, M);  imag\_partp = zeros(1, M);  for n = 1 : M  real\_part0(n) = real(g0(W0(n)));  imag\_part0(n) = imag(g0(W0(n)));  real\_partp(n) = real(gp(Wp(n)));  imag\_partp(n) = imag(gp(Wp(n)));  end  plot(real\_part0, imag\_part0, 'b-o');  grid on;  hold on;  plot(real\_partp, imag\_partp, 'r--\*');  title('Nyquist chart');  xlabel('Real Axis');  ylabel('Imaginary Axis');  legend('Original', 'Area method'); |

图表, 折线图

描述已自动生成

4) Least squares method in time domain

Using recursive solution:

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  K = 0.18831;  T = 1.5336;  L = 2.5490;  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Sampling on the original step response  start\_time = 0;  end\_time = 20;  Ts = 0.1;  s\_num = (end\_time - start\_time) / Ts;  t\_sample = start\_time : Ts : end\_time - Ts;  [y0, t0] = step(G0, t\_sample);  [yp, tp] = step(Gp, t\_sample);    % Define the trapezoidal integration  syms w;  g0(w) = (y0(s\_num) + w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* sin(w \* t0))) + 1i \* w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* cos(w \* t0));  gp(w) = (yp(s\_num) + w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* sin(w \* tp))) + 1i \* w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* cos(w \* tp));    % Set initial values  M = 10;  W0 = zeros(1, M);  Wp = zeros(1, M);  fai0 = zeros(1, M);  faip = zeros(1, M);    W0(1) = 0;  W0(2) = 10^-3;  Wp(1) = 0;  Wp(2) = 10^-3;  fai0(1) = 0;  fai0(2) = angle(g0(W0(2)));  faip(1) = 0;  faip(2) = angle(gp(Wp(2)));    % Recursive solution  for i = 3 : M  W0(i) = W0(i - 1) - ((i - 1) \* pi / (M - 1) + fai0(i - 1)) \* (W0(i - 1) - W0(i - 2)) / (fai0(i - 1) - fai0(i - 2));  fai0(i) = angle(g0(W0(i)));  Wp(i) = Wp(i - 1) - ((i - 1) \* pi / (M - 1) + faip(i - 1)) \* (Wp(i - 1) - Wp(i - 2)) / (faip(i - 1) - faip(i - 2));  faip(i) = angle(gp(Wp(i)));  end    real\_part0 = zeros(1, M);  imag\_part0 = zeros(1, M);  real\_partp = zeros(1, M);  imag\_partp = zeros(1, M);  for n = 1 : M  real\_part0(n) = real(g0(W0(n)));  imag\_part0(n) = imag(g0(W0(n)));  real\_partp(n) = real(gp(Wp(n)));  imag\_partp(n) = imag(gp(Wp(n)));  end  plot(real\_part0, imag\_part0, 'b-o');  grid on;  hold on;  plot(real\_partp, imag\_partp, 'r--\*');  title('Nyquist chart');  xlabel('Real Axis');  ylabel('Imaginary Axis');  legend('Original', 'Least squares method in time domain'); |

图表, 折线图

描述已自动生成

5) Least squares method in frequency domain

Using recursive solution:

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  K = 0.1897;  T = 1.0350;  L = 3.1018;  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Sampling on the original step response  start\_time = 0;  end\_time = 20;  Ts = 0.1;  s\_num = (end\_time - start\_time) / Ts;  t\_sample = start\_time : Ts : end\_time - Ts;  [y0, t0] = step(G0, t\_sample);  [yp, tp] = step(Gp, t\_sample);    % Define the trapezoidal integration  syms w;  g0(w) = (y0(s\_num) + w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* sin(w \* t0))) + 1i \* w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* cos(w \* t0));  gp(w) = (yp(s\_num) + w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* sin(w \* tp))) + 1i \* w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* cos(w \* tp));    % Set initial values  M = 10;  W0 = zeros(1, M);  Wp = zeros(1, M);  fai0 = zeros(1, M);  faip = zeros(1, M);    W0(1) = 0;  W0(2) = 10^-3;  Wp(1) = 0;  Wp(2) = 10^-3;  fai0(1) = 0;  fai0(2) = angle(g0(W0(2)));  faip(1) = 0;  faip(2) = angle(gp(Wp(2)));    % Recursive solution  for i = 3 : M  W0(i) = W0(i - 1) - ((i - 1) \* pi / (M - 1) + fai0(i - 1)) \* (W0(i - 1) - W0(i - 2)) / (fai0(i - 1) - fai0(i - 2));  fai0(i) = angle(g0(W0(i)));  Wp(i) = Wp(i - 1) - ((i - 1) \* pi / (M - 1) + faip(i - 1)) \* (Wp(i - 1) - Wp(i - 2)) / (faip(i - 1) - faip(i - 2));  faip(i) = angle(gp(Wp(i)));  end    real\_part0 = zeros(1, M);  imag\_part0 = zeros(1, M);  real\_partp = zeros(1, M);  imag\_partp = zeros(1, M);  for n = 1 : M  real\_part0(n) = real(g0(W0(n)));  imag\_part0(n) = imag(g0(W0(n)));  real\_partp(n) = real(gp(Wp(n)));  imag\_partp(n) = imag(gp(Wp(n)));  end  plot(real\_part0, imag\_part0, 'b-o');  grid on;  hold on;  plot(real\_partp, imag\_partp, 'r--\*');  title('Nyquist chart');  xlabel('Real Axis');  ylabel('Imaginary Axis');  legend('Original', 'Least squares method in frequency domain'); |

图表, 折线图

描述已自动生成

6) Relay feedback

Using recursive solution:

Matlab Code:

|  |
| --- |
| % Define the original transfer function  s = tf('s');  G0 = 3 \* exp(-2 \* s) / ((s + 4) \* (s^3 + 5\*s^2 + 7\*s + 4));  K = 0.1875;  T = 1.0763;  L = 2.4415;  Gp = K \* exp(- L \* s) / (T \* s + 1);    % Sampling on the original step response  start\_time = 0;  end\_time = 20;  Ts = 0.1;  s\_num = (end\_time - start\_time) / Ts;  t\_sample = start\_time : Ts : end\_time - Ts;  [y0, t0] = step(G0, t\_sample);  [yp, tp] = step(Gp, t\_sample);    % Define the trapezoidal integration  syms w;  g0(w) = (y0(s\_num) + w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* sin(w \* t0))) + 1i \* w \* trapz(t0, (step(G0, t0) - y0(s\_num)) .\* cos(w \* t0));  gp(w) = (yp(s\_num) + w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* sin(w \* tp))) + 1i \* w \* trapz(tp, (step(Gp, tp) - yp(s\_num)) .\* cos(w \* tp));    % Set initial values  M = 10;  W0 = zeros(1, M);  Wp = zeros(1, M);  fai0 = zeros(1, M);  faip = zeros(1, M);    W0(1) = 0;  W0(2) = 10^-3;  Wp(1) = 0;  Wp(2) = 10^-3;  fai0(1) = 0;  fai0(2) = angle(g0(W0(2)));  faip(1) = 0;  faip(2) = angle(gp(Wp(2)));    % Recursive solution  for i = 3 : M  W0(i) = W0(i - 1) - ((i - 1) \* pi / (M - 1) + fai0(i - 1)) \* (W0(i - 1) - W0(i - 2)) / (fai0(i - 1) - fai0(i - 2));  fai0(i) = angle(g0(W0(i)));  Wp(i) = Wp(i - 1) - ((i - 1) \* pi / (M - 1) + faip(i - 1)) \* (Wp(i - 1) - Wp(i - 2)) / (faip(i - 1) - faip(i - 2));  faip(i) = angle(gp(Wp(i)));  end    real\_part0 = zeros(1, M);  imag\_part0 = zeros(1, M);  real\_partp = zeros(1, M);  imag\_partp = zeros(1, M);  for n = 1 : M  real\_part0(n) = real(g0(W0(n)));  imag\_part0(n) = imag(g0(W0(n)));  real\_partp(n) = real(gp(Wp(n)));  imag\_partp(n) = imag(gp(Wp(n)));  end  plot(real\_part0, imag\_part0, 'b-o');  grid on;  hold on;  plot(real\_partp, imag\_partp, 'r--\*');  title('Nyquist chart');  xlabel('Real Axis');  ylabel('Imaginary Axis');  legend('Original', 'Relay feedback'); |

图表, 折线图

描述已自动生成

As shown in the above plots, the results of these models in the Nyquist chart are good, expect one model. The FOPTD under open loop test using LSM in time domain doesn’t perform well. So, I think this method is not suitable for the given transfer function compared to the others. The FOPTD model using area method and under open loop step test using least squares method in frequency domain performs almost perfectly.