Scilab programs

15 .Routh

```
clc;
close; // Close any open figures
// Define the transfer function H(s)
s = %s;
//H = s^4 + 2*s^3 + 3*s^2 + 4*s + 5;
H = s^5 + 7*s^4 + 6*s^3 + 42*s^2 + 8*s + 56;
// Display the given characteristic equation
disp(H, 'The given characteristics equation 1-G(s)H(s)=');
// Calculate the coefficients of H(s)
c = coeff(H);
len = length(c);
// Calculate the Routh's table
r = routh_t(H);
disp(r, 'Routh''s table=');
// Check the stability from Routh's table
x = 0;
for i = 1:len
   if(r(i,1) < 0)
      x = x + 1;
   end
end
// Display stability based on the Routh table
if x >= 1 then
   printf('From Routh''s table, it is clear that the system is unstable.\n');
else
   printf('From Routh''s table, it is clear that the system is stable.\n');
end
```

17. discrete signals

```
// UNIT IMPULSE SIGNAL
clear all;
close;
N = 5; // SET LIMIT
t1 = -5:5;
x1 = [zeros(1, N), ones(1, 1), zeros(1, N)];
subplot(2, 4, 1);
plot2d3(t1, x1);
xlabel('time');
ylabel('Amplitude');
title('Unit impulse signal');

// UNIT STEP SIGNAL
t2 = -5:5;
x2 = [zeros(1, N), ones(1, N + 1)];
subplot(2, 4, 2);
plot2d3(t2, x2);
xlabel('time');
ylabel('Amplitude');
title('Unit step signal');

// EXPONENTIALLY INCREASING SIGNAL
t3 = 0:0.1:10; // Time vector
x3 = exp(0.5 * t3); // Exponentially increasing signal

// Plot the signal
subplot(2, 3, 3); // Create subplot
plot2d3(t3, x3); // Use continuous plot function
xlabel("Time");
```

```
ylabel("Amplitude");
title("Exponentially Increasing Signal");

// Reverse X-Axis (Optional for cleaner labeling)
a = gca();
a.x_location = "top";  // Move the x-axis to the top
a.grid = [1 1];        // Add grid lines

// UNIT RAMP SIGNAL
t4 = 0:20;
x4 = t4;
subplot(2, 3, 4);
plot2d3(t4, x4);
xlabel('time');
ylabel('Amplitude');
title('Unit ramp signal');
// SINUSOIDAL SIGNAL
t5 = 0:0.04:1;
x5 = 10 * sin(2 * %pi * t5);
subplot(2, 3, 5);
plot2d3(t5, x5);
xlabel('time');
ylabel('Amplitude');
title('Sinusoidal signal');
// RANDOM SIGNAL
t6 = -10:1:20;
x6 = rand(1, 31);
subplot(2, 3, 6);
plot2d3(t6, x6);
xlabel('time');
ylabel('Amplitude');
title('Random signal');
```

18 . (a) DIT-FFT Algorithm

```
clear;
clc ;
close ;
x = [1,-1,-1,-1,1,1,1,-1];
//FFT Computation
X = fft (x , -1);
disp(X,'X(z) = ');
```

18. (b) DIF-FFT Algorithm

```
clear;
clc ;
close ;
x = [1,2,3,4,4,3,2,1];
//FFT Computation
X = fft (x , -1);
disp(X,'X(z) = ');
```

19. Design a filter using the Transformation Method.

(a) Bilinear Transformation

```
clear;
clc;
close;
s = %s;
T = 1;
H1 = (s^2 + 4.5) / (s^2 + 0.692*s + 0.504);
z = %z;
s_bilinear = (2 / T) * (1 - z^(-1)) / (1 + z^(-1));
H1_digital = horner(H1, s_bilinear);
disp("Digital Transfer Function H(z):");
disp("Numerator coefficients of H(z): " + string(coeff(H1_digital.num)));
disp("Denominator coefficients of H(z): " + string(coeff(H1_digital.den)));
```

(b) Impulse Invariant Transformation

```
// Impulse Invariant Transformation (Simplified)

// Clear previous data
clear; close; clc;

// Analog transfer function coefficients
a = 10;          // Cutoff frequency
num_a = [1];     // Numerator coefficients
den_a = [1, a];  // Denominator coefficients

// Sampling time
Ts = 0.1;

// Map analog poles to discrete poles
poles_a = roots(den_a);       // Find poles of the analog system
poles_d = exp(poles_a * Ts);  // Convert poles to discrete-time

// Form the discrete-time denominator
den_d = poly(poles_d, 'z');

// Match DC gain
gain = num_a(1) / den_a(1);   // DC gain of the analog filter
num_d = gain * den_d(1);      // Scale numerator for digital filter

// Display results
disp('Analog Transfer Function Coefficients:');
disp('Numerator: ' + string(num_a));
disp('Denominator: ' + string(den_a));

disp('Discrete Transfer Function Coefficients:');
```

```
disp('Numerator: ' + string(num_d));
disp('Denominator: ' + string(den_d));
```

20. Write the SCILAB program to design the following Butterworth filters

(a)Low pass filter

```
// First Order Butterworth Low Pass Filter
clear;
clc;
close;
s = poly(0, 's');
Omegac = 0.2 * %pi;
H = Omegac / (s + Omegac);
T = 1; // Sampling period T = 1 Second
z = poly(0, 'z');
Hz = horner(H, (2 / T) * ((z - 1) / (z + 1)));
HW = frmag(Hz(2), Hz(3), 512);
W = 0:%pi / 511:%pi;
plot(W / %pi, HW);
a = gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1);
xtitle('Magnitude Response of Single Pole LPF Filter Cutoff Frequency = 0.2*pi', ...
    'Digital Frequency--->', 'Magnitude');
disp('Hz = ', Hz);
```

(b)High pass filter

```
// First Order Butterworth Filter
// High Pass Filter Using Digital Filter Transformation
clear;
clc;
close;
s = poly(0, 's');
Omegac = 0.2 * %pi;
H = Omegac / (s + Omegac);
T = 1; // Sampling period T = 1 Second
z = poly(0, 'z');
Hz_LPF = horner(H, (2 / T) * ((z - 1) / (z + 1)));
alpha = -(cos((Omegac + Omegac) / 2)) / (cos((Omegac - Omegac) / 2));
HZ_HPF = horner(Hz_LPF, -(z + alpha) / (1 + alpha * z));
HW = frmag(HZ_HPF(2), HZ_HPF(3), 512);
W = 0:%pi / 511:%pi;
plot(W / %pi, HW);
a = gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
xgrid(1);
xtitle('Magnitude Response of Single Pole HPF Filter Cutoff Frequency = 0.2*pi', ...
    'Digital Frequency--->', 'Magnitude');
disp('HZ_HPF = ', HZ_HPF);
```

(c)Band pass filter

```
clear;
```

```
clc;
close;

omegaP = 0.2 * %pi;
omegaL = (2/5) * %pi;
omegaU = (3/5) * %pi;

z = poly(0, 'z');
H_LPF = (0.245) * (1 + (z^-1)) / (1 - 0.509 * (z^-1));

alpha = cos((omegaU + omegaL) / 2) / cos((omegaU - omegaL) / 2);
k = (cos((omegaU - omegaL) / 2) / sin((omegaU - omegaL) / 2)) * tan(omegaP / 2);

NUM = -((z^2) - ((2 * alpha * k / (k + 1)) * z) + ((k - 1) / (k + 1)));
DEN = (1 - ((2 * alpha * k / (k + 1)) * z) + (((k - 1) / (k + 1)) * (z^2)));

HZ_BPF = horner(H_LPF, NUM / DEN);
disp(HZ_BPF, 'Digital BPF IIR Filter H(Z)= ');

HW = frmag(HZ_BPF(2), HZ_BPF(3), 512);
W = 0:%pi/511:%pi;

plot(W / %pi, HW);

a = gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;

xgrid(1);
xtitle('Magnitude Response of BPF Filter', 'Digital Frequency--->', 'Magnitude');
disp("HZ_BPF", HZ_BPF);
```

(d)Band reject filter.

```
clear;
clc;
close;

omegaP = 0.2 * %pi;
omegaL = (2 / 5) * %pi;
omegaU = (3 / 5) * %pi;

z = poly(0, 'z');
H_LPF = (0.245) * (1 + (z^-1)) / (1 - 0.509 * (z^-1));

alpha = cos((omegaU + omegaL) / 2) / cos((omegaU - omegaL) / 2);
k = tan((omegaU - omegaL) / 2) * tan(omegaP / 2);

NUM = ((z^2) - ((2 * alpha / (1 + k)) * z) + ((1 - k) / (1 + k)));
DEN = (1 - ((2 * alpha / (1 + k)) * z) + (((1 - k) / (1 + k)) * (z^2)));

HZ_BSF = horner(H_LPF, NUM / DEN);
disp(HZ_BSF, 'Digital BSF IIR Filter H(Z)= ');

HW = frmag(HZ_BSF(2), HZ_BSF(3), 512);
W = 0:%pi/511:%pi;

plot(W / %pi, HW);

a = gca();
a.thickness = 3;
a.foreground = 1;
a.font_style = 9;
```

```
xgrid(1);
xtitle('Magnitude Response of BSF Filter', 'Digital Frequency--->', 'Magnitude');
disp("HZ_BSF", HZ_BSF);
```