Roll: 1715006 Name: Adowan Shahriar LAB: 3.

Objectives:

- To find the pole locations of various systems.
- To find the pole locations of unity feedback system.
- To find the steady state errors of system from transfer function.
- To be able to evaluate K_p, K_v and K_a.

Introduction: -

Stability is the most important system specification. If a system is unstable transient response and steady state errors are moot points. The total response of a system is the sum of the forced and natural responses on,

$$c(t) = c_{forced}(t) + c_{natural}(t)$$

A linear time-invariant system is stable if the natural responses approaches zero as time approaches infinity. A linear time-invariant system is unstable if the natural responses grow without bound as time approaches infinity.

A ~~time~~ linear time-invariant system is marginally stable if the natural response neither decays nor grows but remains constant or oscillates as time approaches infinity.

Thus the definition of stability implies that only the forced response remains as the natural response approaches zero.

These definitions rely on a description of the natural response. When one is looking at the total response, it may be difficult to separate the natural response from the forced response. However we realize that if the input is bounded and the total response is not approaching infinity as time approaches infinity, then the natural response is obviously not approaching infinity. If the input is unbound, we see an ~~a~~ unbounded total response and we can not arrive at any conclusion about the stability of the system; we can not tell whether the total response is unbound or because the natural response is unbound.

Thus our alternate definition of stability, one that regards the total response and implies the first definition based upon the natural response, is this:

A system is stable if every bounded input yields a bounded output.

We cal this statement the bounded-input, bounded output (BIBO) definition of stability.

Task 1:

Editor - C:\Users\DELL\Untitled.m

Untitled.m

```
numg=-6; % Define numerator of G(s).
deng=conv ([1 0],[1 1 -6 0 1 -6] ); % Define denominator of G (s).
G=tf(numg,deng); % Create G(s) object.
'T(s)' % Display label.
 % Calculate closed-loop T(s)
% object.
T=feedback(G,1)
% Negative feedback is default
% when there is no sign parameter.
poles=pole(T) % Find poles of T(s).
```

Output:

```
>> Untitled

ans =

    'T(s)'


T =

                 -6
  ---------------------------------
  s^6 + s^5 - 6 s^4 + s^2 - 6 s - 6

Continuous-time transfer function.


poles =

  -2.9453 + 0.0000i
   2.1339 + 0.0000i
   0.6478 + 0.9622i
   0.6478 - 0.9622i
  -0.7421 + 0.3984i
  -0.7421 - 0.3984i
```

Task2.a:

Editor - C:\Users\DELL\Untitled.m

Untitled.m

```
clc
numg=240; % Define numerator of G(s).
deng=conv ([1 0],[1 10 35 50 24]); % Define denominator of G (s).
G=tf(numg,deng); % Create G(s) object.
'T(s)' % Display label.
T=feedback(G, 1) % Calculate closed-loop T(s)
% object.
% Negative feedback is default
% when there is no sign parameter.
poles=pole(T) % Find poles of T(s).
```

Output:

```
ans =

    'T(s)'


T =

                      240
  -------------------------------------------
  s^5 + 10 s^4 + 35 s^3 + 50 s^2 + 24 s + 240

Continuous-time transfer function.


poles =

  -5.3341 + 0.0000i
  -3.0221 + 2.5327i
  -3.0221 - 2.5327i
   0.6891 + 1.5554i
   0.6891 - 1.5554i
```

Task 2.b:

Editor - C:\Users\DELL\Untitled.m

Untitled.m

```
clc
numg=8; % Define numerator of G(s).
deng=conv ([1 0],[1 -2 -1 2 4 -8 4 0]); % Define denominator of G (s).
G=tf(numg,deng); % Create G(s) object.
'T(s)' % Display label.
  % Calculate closed-loop T(s)
% object.
T=feedback(G, 1)
% Negative feedback is default
% when there is no sign parameter.
poles=pole(T) % Find poles of T(s).
```

Output:

```
ans =

    'T(s)'


T =

                                8
  ----------------------------------------------------
  s^8 - 2 s^7 - s^6 + 2 s^5 + 4 s^4 - 8 s^3 + 4 s^2 + 8

Continuous-time transfer function.


poles =

   1.7162 + 0.2739i
   1.7162 - 0.2739i
   0.8633 + 0.9912i
   0.8633 - 0.9912i
  -1.2233 + 0.8238i
  -1.2233 - 0.8238i
  -0.3562 + 0.7602i
  -0.3562 - 0.7602i
```

Task 2.c:

Editor - C:\Users\DELL\Untitled.m

Untitled.m

```matlab
numg=18; % Define numerator of G(s).
deng=conv ([1 0],[1 1 -7 -7 -18 0]); % Define denominator of G (s).
G=tf(numg,deng); % Create G(s) object.
'T(s)' % Display label.
T=feedback(G, 1) % Calculate closed-loop T(s)
% object.
% Negative feedback is default
% when there is no sign parameter.
poles=pole(T) % Find poles of T(s).
```

Output:

```
>> Untitled

ans =

    'T(s)'


T =

                    18
  ---------------------------------------
  s^6 + s^5 - 7 s^4 - 7 s^3 - 18 s^2 + 18

Continuous-time transfer function.


poles =

   2.8988 + 0.0000i
  -3.0834 + 0.0000i
  -0.3135 + 1.5437i
  -0.3135 - 1.5437i
  -1.0000 + 0.0000i
   0.8116 + 0.0000i
```

Task 3:

```
K=[1:1:2000]; % Define range of K from 1 to 2000
% in steps of 1.
for n=1:1:length(K); % Set up length of DO LOOP to equal
% number of K values to be tested.
dent=[1 18 77 K(n)]; % Define the denominator of T(s)
% for the nth value of K.
poles=roots(dent); % Find the poles for the nth value
% of K.
r=real(poles); % Form a vector containing the real
% parts of the poles for K(n).
if max(r) >=0, % Test poles found for the nth
% value of K for a real value    0.
poles % Display first pole values where
% there is a real part    0.
%804 Appendix B: MATLAB Tutorial
K=K(n) % Display corresponding value of K.
break % Stop loop if rhp poles are found.
end % End if.
end % End for.
```

Output:

```
poles =

 -18.0025 + 0.0000i
   0.0012 + 8.7775i
   0.0012 - 8.7775i


K =

        1387
```

Editor - C:\Users\DELL\Untitled.m

Untitled.m

```
clc
numg=1385; % Define numerator of G(s).
deng=conv ([1 0],[1 18 77 numg]); % Define denominator of G (s).
G=tf(numg,deng); % Create G(s) object.
'T(s)' % Display label.
T=feedback(G, 1)
  % Calculate closed-loop T(s)
% object.
% Negative feedback is default
% when there is no sign parameter.
poles=pole(T) % F
```

```
ans =

    'T(s)'


T =

                     1385
  ------------------------------------
  s^4 + 18 s^3 + 77 s^2 + 1385 s + 1385

Continuous-time transfer function.


poles =

 -17.7999 + 0.0000i
   0.4234 + 8.6108i
   0.4234 - 8.6108i
  -1.0469 + 0.0000i
```

Discussion :

In this lab we mainly focused on calculating parametors of closed loop system and according to given equations. We calculated pole locations of systems assuming system types and types of feed-back. And all by MATLAB coding. We coded to calculate pole locations of general transfer function and also closed loop system transfer functions. We calculated ~~stabily~~ stability of unity feedback system and steady state errors. We also calculated steady-state error constants.

Conclusion :

We learned to calculate pole locations of various types of and feedback system's transfer function. We learned to evaluate steady state error and steady-state error constants. Overall though this pandemic situation is already harsh on us, but we conducted this lab with the help of our course teacher.