UNIVERSITY COLLEGE DUBLIN

SCHOOL OF ELECTRICAL & ELECTRONIC ENG.

EEEN30110 SIGNALS AND SYSTEMS

Experiment 3SS4

Z-TRANSFORM

1. Objective:

To investigate the Z-transform and discrete-time system analysis.

2. Background Information:

See information in handout for laboratory 3SS1-3SS3 and course notes.

Recall that Matlab can store a rational polynomial by storing the numerator and denominator polynomials separately in Matlab notation. There is a more sophisticated way that Matlab can represent a rational polynomial, namely as a *system variable*. As discussed previously this is achieved by the command **tf**. In this laboratory we will seek to create a system variable corresponding to a *discrete-time* system. So for example, suppose we wish to store the rational polynomial $\frac{z+0.2}{z^3+0.9z^2+0.4z+0.1}$. We can store the numerator and denominator polynomials separately as:

```
>> N = [1 0.2]
>> D = [1 0.9 0.4 0.1]
```

or carry on to create the system variable:

```
>> Sys_example = tf(N,D,-1)

Transfer function:
        z + 0.2
  --------------------------
z^3 + 0.9 z^2 + 0.4 z + 0.1

Sampling time: unspecified
```

here the third argument is set equal to the sampling period if this is known or -1 if, as in this case, it is not. The denominator polynomial D is called the *characteristic polynomial* of the system. The roots of this polynomial are called the *system poles*. We can determine the system poles by entering either of the following commands:

```
>> roots(D)
```

or

```
>> pole(Sys_example)
```

ans =

-0.5000
-0.2000 + 0.4000i
-0.2000 - 0.4000i

In this case two of the poles are complex and are therefore conjugates of one another and the third is real. All three poles have modulus less than one.

The roots of the numerator polynomial N are called the *system zeros*. We can determine the system zeros by entering either of the following commands:

>> roots(N)

or

>> zero(Sys_example)

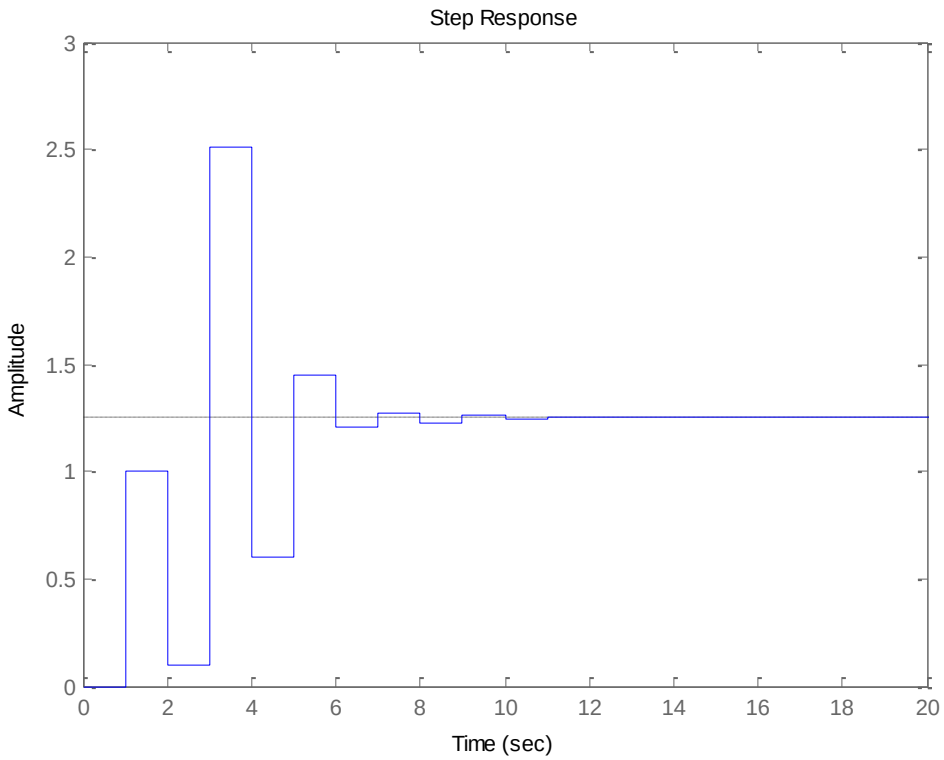
ans =

-0.2

If we wish to find the *forced component* of the discrete-time unit step response of the system whose transfer function is $\frac{z+0.2}{z^3+0.9z^2+0.4z+0.1}$ we may enter the following command:

>> step(Sys_Example)

Matlab opens a figure window and plots the forced step response:



If a figure window is already open then Matlab overwrites it (unless the hold function is active). It should be noted that Matlab effectively makes an error here in its declaration that the units of time are seconds. In this case where we do not know the sampling period the units should really be samples.

Quantitative features of the step response can be obtained by right clicking on the background space of the figure and choosing **Characteristics** as usual. Alternatively the command:

```
>> stepinfo(Sys_Example)

ans =

      RiseTime: 2
  SettlingTime: 7
   SettlingMin: 0.6010
   SettlingMax: 2.5100
     Overshoot: 100.8000
    Undershoot: 0
          Peak: 2.5100
      PeakTime: 3
```

indicates that the 2% settling time is 7 samples or 7 discrete-time steps. The overshoot is a little over 100%

If the system has transfer function $\frac{z+0.2}{z^3+0.9z^2+0.4z+0.1}$ then, since the discrete-time unit step has Z-transform $\frac{1}{1-z^{-1}}=\frac{z}{z-1}$ (which we may store as a system variable using the code Step_Input = tf([1 0],[1 -1],-1) for example), it follows that the Z-transform of the output is the product of these two rational polynomials. To multiply two rational polynomials in *z* that have been stored as discrete-time system variables one may use the Matlab command **series**.

One may find the inverse Z-transform of a rational polynomial in z either numerically by creating an associated discrete-time system variable as above and using the command **impulse**, or analytically, by using the **residue** command to find the partial fraction expansion and inverting term-by-term. The problem of course is that in lectures I work in terms of z^{-1} and so compute the partial fraction expansion in these terms, whereas Matlab works in the more reasonable terms of z. Accordingly a typical term in the partial fraction expansion computed by Matlab will have the form: $\frac{A}{(z-\mu)^k}$, so you will have to adjust the inversion formula in the notes to suit this alternative formulation. To this end note that: $\frac{A}{(z-\mu)^k}=\frac{A(-1)^k}{\mu^k}\frac{1}{\left(z^{-1}-\frac{1}{\mu}\right)^k}z^{-k}$.

3. System Problems

1. A discrete-time system has an input $x(n)$ and output $y(n)$. It is described by the linear, constant-coefficient difference equation:

$$y(n)-0.5y(n-1)-0.25y(n-2)=x(n)+2x(n-1)+x(n-2)$$

Find the transfer function $H(z)$ of the system. Determine the unit step response of the system and find the percentage overshoot, the 2% settling time and the steady-state value.

Grading: Transfer function = 1, Unit step response 1, PO = 1, ts(2%) = 1, SSV = 1.

2. Assume zero initial conditions. Given that the input to the system is a discrete-time sinusoid commencing at $n=0$, i.e. $x(n)=\cos(\Omega_0 n)u(n)$, find the steady-state response of the system for the four special frequencies: $\Omega_0 = 0, \frac{\pi}{7}, \frac{\pi}{2}, \pi$. Explain in particular why the latter is zero.

Grading: SS Response to each frequency = 4, Explanation = 1.

3. Plot the frequency response $\left|H\left(e^{j\Omega}\right)\right|$ vs. Ω for $-\pi \le \Omega \le \pi$. Hence discuss the filtering properties of the system.

Grading: Plot = 2.5, discussion = 1.

4. Assume initial conditions: $y(-1)=1$, $y(-2)=0$. Assume that the input signal is a discrete-time unit step. Find a formula for the Z-transform of the solution $y(n)$. Hence find a formula for the solution $y(n)$ itself.

Grading: Z-transform 2.5, formula 5