EME172 Discussion 2

Introduction to Ch Control System Toolkit

(1) Creating Linear Models

```
int model("tf", array double num[&], array double den[&]);
```

- Create transfer function models
- *num* Array of double containing the coefficients of the numerator
- *den* Array of double containing the coefficients of the denominator

```
int model("ss", array double &a, array double &b,
                array double &c, array double &d);
```

- Create state-space models
- *a* Array of reference containing the value of system matrix
- *b* Array of reference containing the value of input matrix
- *c* Array of reference containing the value of output matrix
- *d* Array of reference containing the value of transmission matrix

```
int model("zpk", array double complex z[&],
                 array double complex p[&],
                 double k);
```

- Create zero/pole/gain models
- *z* Array of double containing the zeros of the system
- *p* Array of double containing the poles of the system
- *k* Double type value containing the gain of the system

Example 1: Transfer Function Models

$$H(s) = \frac{s}{s^2 + 2s + 10}$$

Program:

```
/* example1.ch */
#include <control.h>

int main() {
    array double num[2] = {1, 0}, den[3] = {1, 2, 10};
    CControl sys;

    sys.model("tf", num, den);
    sys.printtf();

    return 0;
}
```

Output:

```
Transfer function parameters:
Numerator: 1.000000*s+
Denominator: 1.000000*s*s+2.000000*s+10.000000
```

Example 2: State-Space Models

$$A = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$D = 0$$

Program:

```
/* example2.ch */
#include <control.h>

int main() {
    array double A[2][2] = {0, 1, -5, -2},
                 B[2][1] = {0, 3},
                 C[2] = {1, 0},
                 D[1] = {0};

    CControl sys;

    sys.model("ss", A, B, C, D);
    sys.printss();

    return 0;
}
```

Output:

```
A =
  0.000000   1.000000
 -5.000000  -2.000000
B =
  0.000000
  3.000000
C =
  1.000000   0.000000
D =
  0.000000
```

Example 3: Zero-Pole-Gain Models

$$H(s) = -2\frac{s}{(s-2)(s^2-2s+2)}$$

Program:

```
/* example3.ch */
#include <control.h>

int main() {
    array double complex z[1] = {0};
    array double complex p[3] = {2, complex(1, 1), complex(1, -1)};
    double k = -2;
    CControl sys;

    sys.model("zpk", z, p, k);
    sys.printtf();

    return 0;
}
```

Output:

```
Transfer function parameters:
Numerator: -2.000000*s+
Denominator: 1.000000*s*s*s-4.000000*s*s+6.000000*s-4.000000
```

(2) Time Domain Analysis

```
int step(class CPlot *plot, array double &yout,
                            array double &tout,
                            array double &xout,
                            /* double tf */);
```

- Calculate and plot step response of a system
- *plot* Pointer to an existing object of class CPlot
- *yout* Array of reference containing the output of the step response
- *tout* Array of reference containing the time vector for the simulation
- *xout* Array of reference containing the state trajectories
- *tf* Double value specifying the final time of the simulation

```
int impluse(class CPlot *plot, array double &yout,
                               array double &tout,
                               array double &xout,
                               /* double tf */);
```

- Calculate and plot impulse response of a system
- *plot* Pointer to an existing object of class CPlot
- *yout* Array of reference containing the output impulse response
- *tout* Array of reference containing the time vector for the simulation.
- *xout* Array of reference containing the state trajectories.
- *tf* Double value specifying the final time of the simulation.

Example 4: Step Response for Transfer Function Model

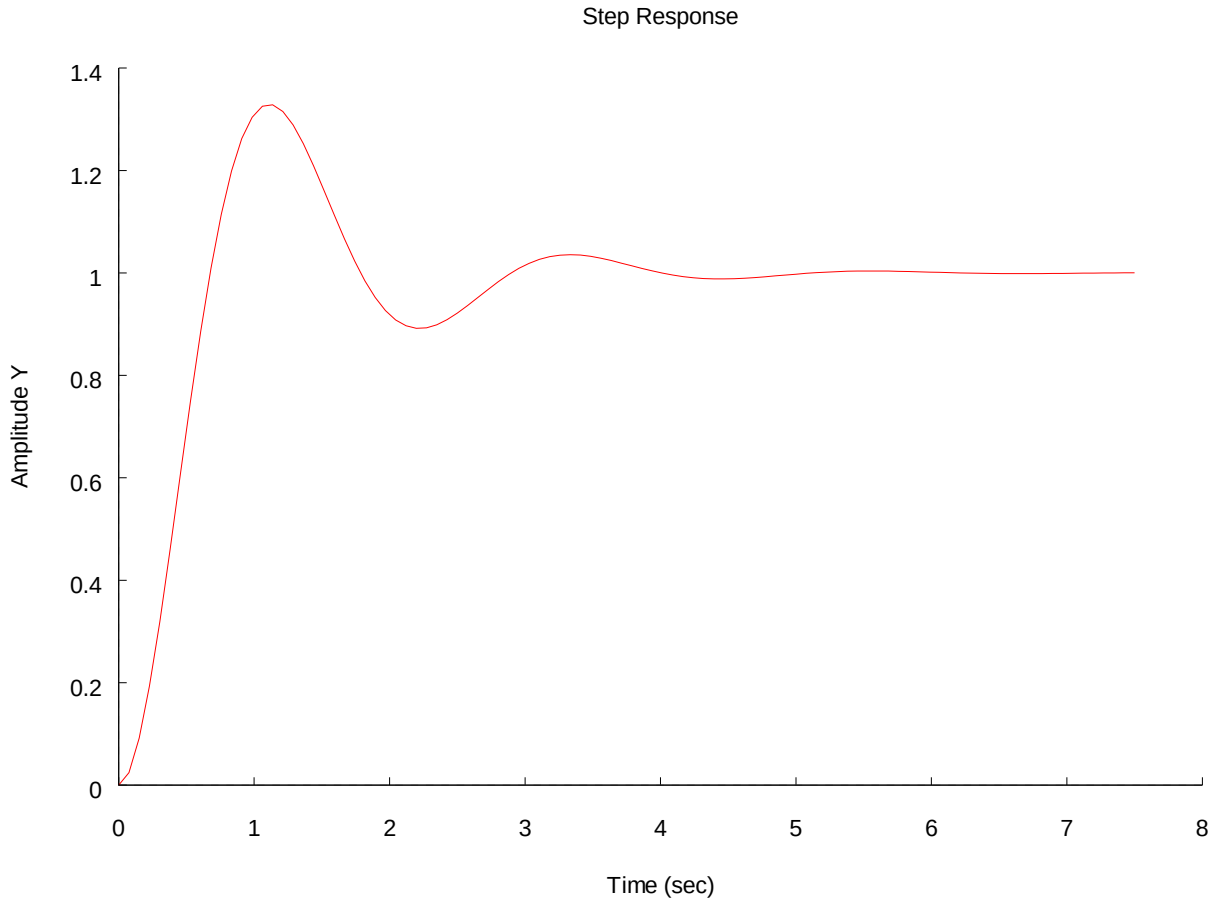
Plot the step response of the following second-order system.

$$H(s) = \frac{9}{s^2 + 2s + 9}$$

Program:

```
/* example4.ch */
#include <control.h>

int main() {
    array double num[1] = {9},
                 den[3] = {1, 2, 9};
    CControl sys;
    CPlot plot;

    sys.model("tf", num, den);
    sys.step(&plot, NULL, NULL, NULL);

    return 0;
}
```

Output:


Step Response
Amplitude Y
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6
7
8

Example 5: Step Response for State-Space Model

Plot the step response of the following system.

$$A = \begin{bmatrix} -2 & -9 \\ 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

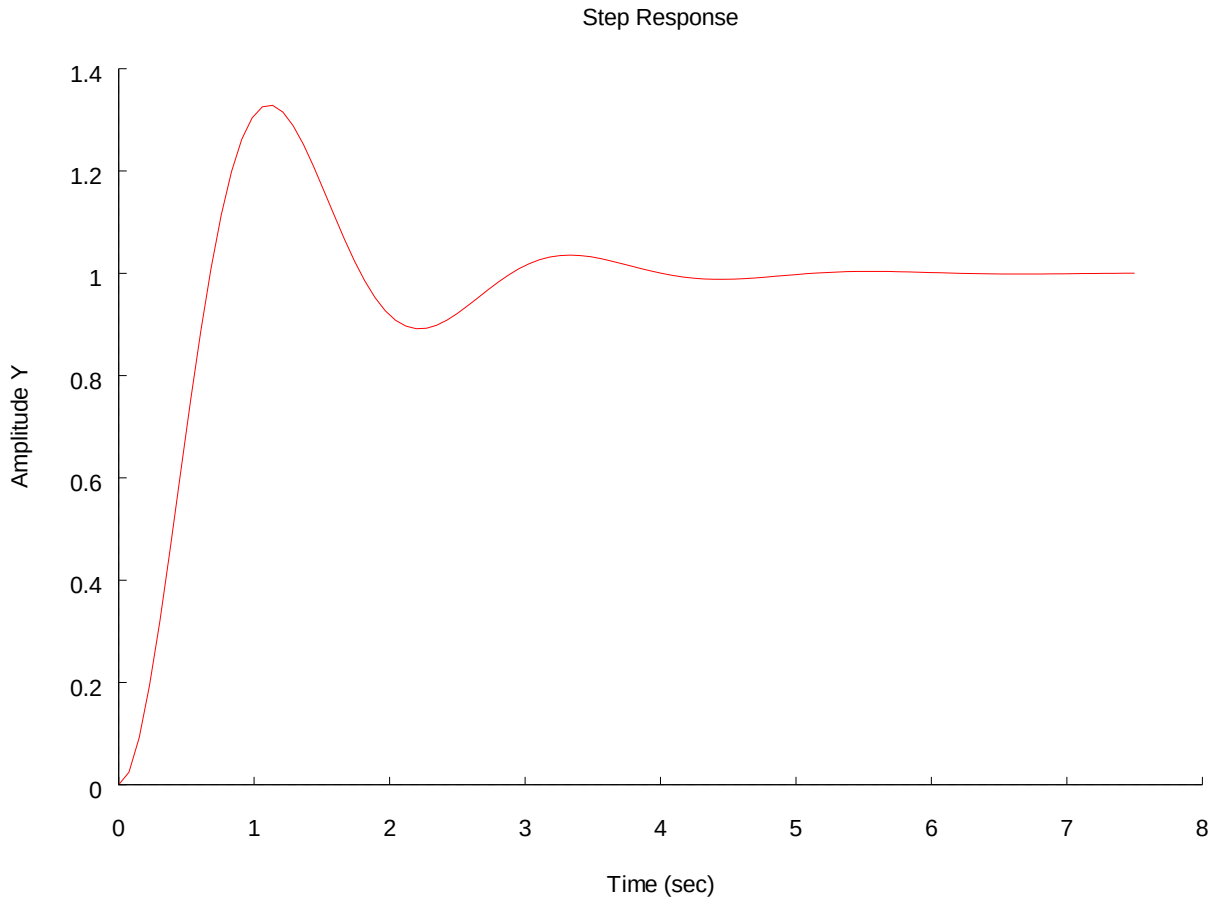
$$C = \begin{bmatrix} 0 & 9 \end{bmatrix}$$

$$D = 0$$

Program:

```
/* example5.ch */
#include <control.h>

int main() {
    array double A[2][2] = {-2, -9, 1, 0},
                 B[2][1] = {1, 0},
                 C[2] = {0, 9},
                 D[1] = {0};

    CControl sys;
    CPlot plot;

    sys.model("ss", A, B, C, D);
    sys.step(&plot, NULL, NULL, NULL);

    return 0;
}
```

Output:


Step Response
Amplitude Y
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6
7
8

Example 6: Step Response for Zero-Pole-Gain Model

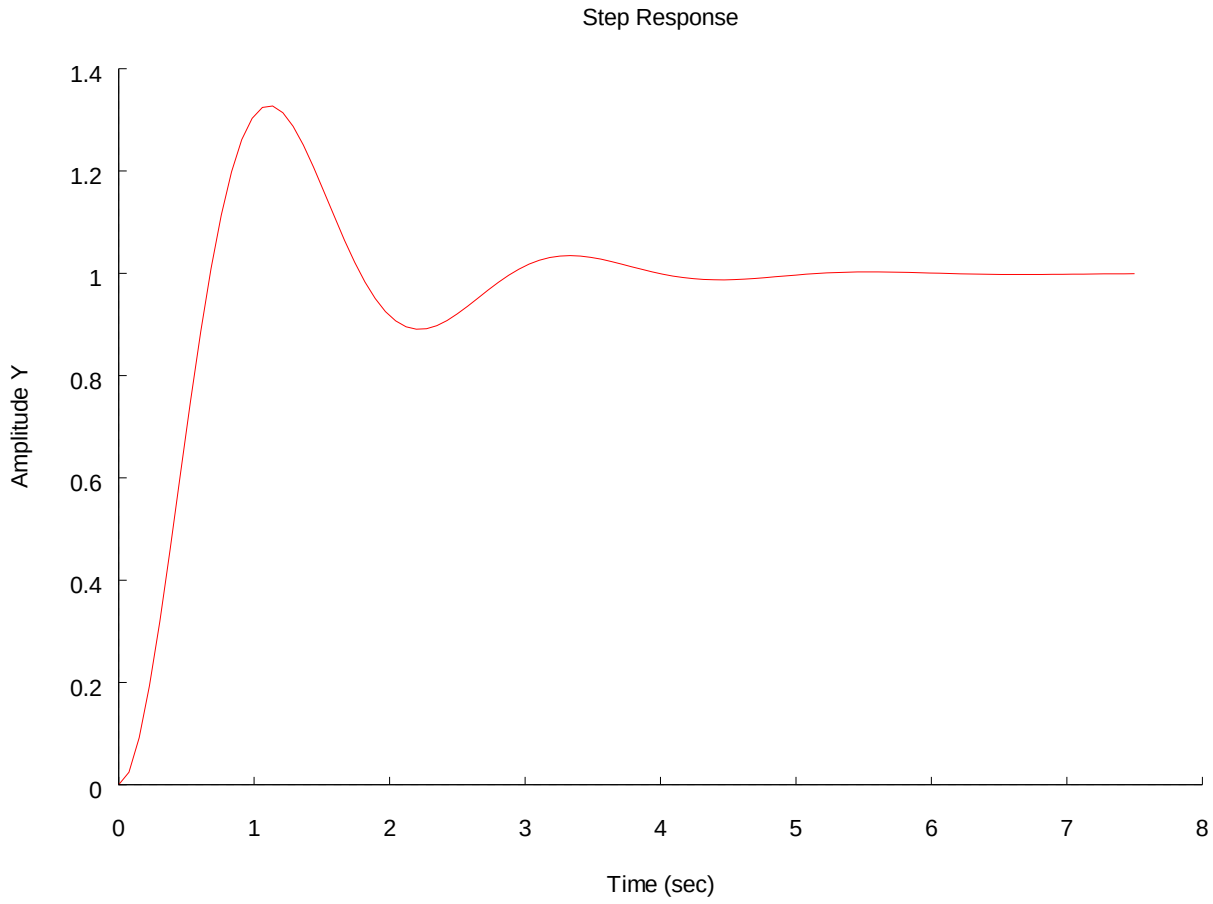
Plot the step response of the following system.

$$H(s) = \frac{9}{[s-(-1+2.83i)][s-(-1-2.83i)]}$$

Program:

```
/* example6.ch */
#include <control.h>

int main() {
    array double complex p[2] = {complex(-1, 2.83), complex(-1, -2.83)};
    double k = 9;
    CControl sys;
    CPlot plot;

    sys.model("zpk", NULL, p, k);
    sys.step(&plot, NULL, NULL, NULL);

    return 0;
}
```

Output:


Step Response
Amplitude Y
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
1
2
3
4
5
6
7
8

(3) System Gain and Dynamics

```
array double dcgain()[:][:];
```

- Find DC (low-frequency) gain

```
int pzmap(class CPlot *plot, array double complex &p,
                             array double complex &z);
```

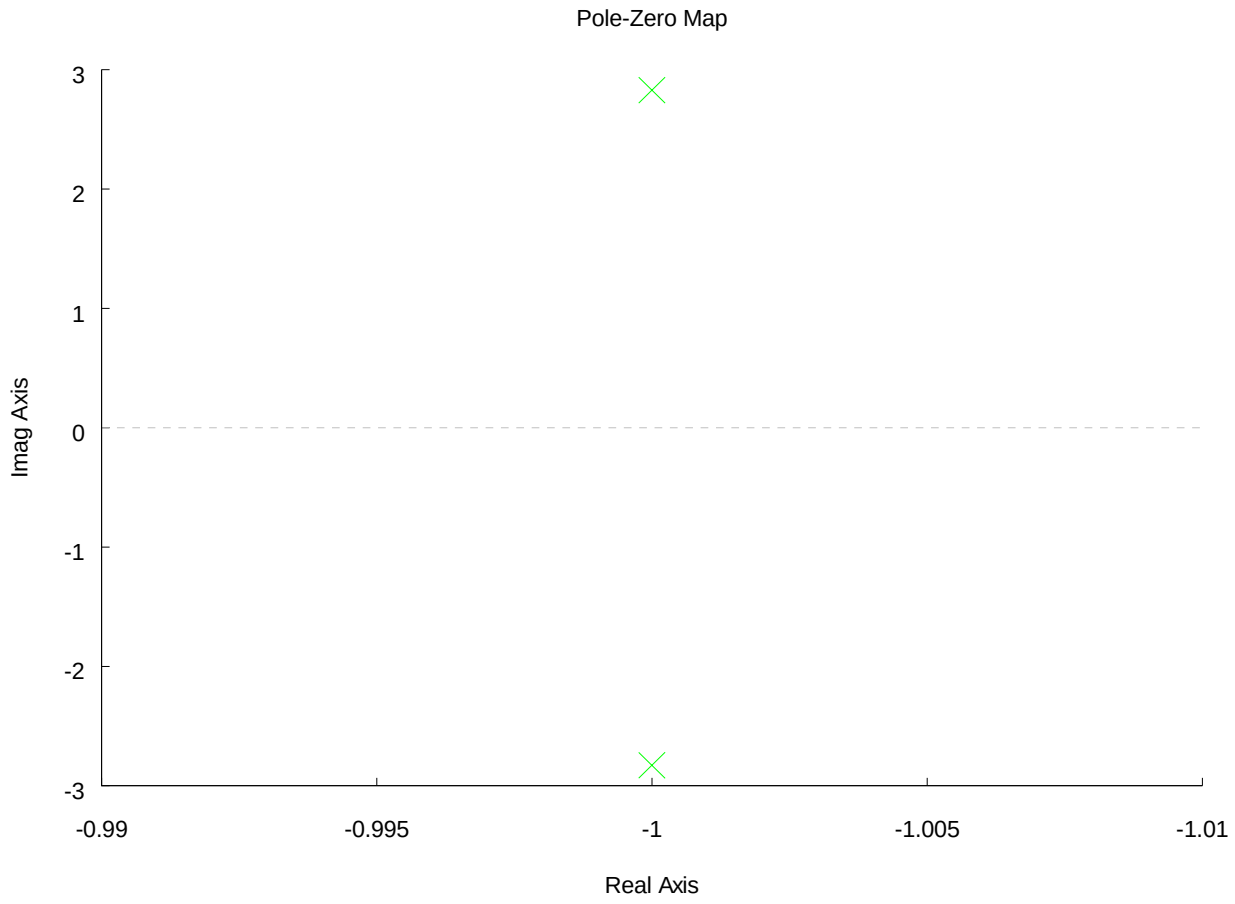
- Create the pole-zero map and calculate the pole(s) and zero(s) of a system.
- *plot* Pointer to an existing object of class CPlot
- *p* Array of reference containing the system poles
- *z* Array of reference containing the system zeros

Example 7: Find the DC gain, poles and zeros of the system in Example 4, and plot pole/zero map.

Program:

```
/* example7.ch */
#include <control.h>

int main() {
    array double num[1] = {9},
                 den[3] = {1, 2, 9},
                 k[1][1];
    int np, nz;
    CControl sys;
    CPlot plot;

    sys.model("tf", num, den);
    np = sys.size('p');
    nz = sys.size('z');

    array double complex p[np], z[nz];

    k = sys.dcgain();
    sys.pzmap(&plot, p, z);

    printf("k = %f\n", k);
    printf("p = %f\n", p);
    printf("z = %f\n", z);

    return 0;
}
```

Output:



k = 1.000000

p = complex(-1.000000,2.828427) complex(-1.000000,-2.828427)

z =