(1) Frequency Domain Analysis

```c
int bode(class CPlot *plot, array double mag[],
          array double phase[],
          array double wout[],
          /* double w[]
            or
            double wmin, double wmax */);
```

- `plot`  Pointer to an existing object of class CPlot
- `mag`   Array of reference containing the magnitudes of the frequency response at the frequencies in array `wout`
- `phase` Array of reference containing the phases of the frequency response at the frequencies in array `wout`
- `wout`  Array of reference containing the output frequencies
- `w`     Array of reference containing the user-defined frequencies
- `wmin` and `wmax` Two double values specifying the frequency interval

**Example 1: Create the Bode Plot for a System**

Create the Bode plot for the following system.

\[
H(s) = \frac{1}{(s+1)(s+2)(s+3)}
\]

Program:

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

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

  CControl sys;
  CPlot plot;

  sys.model("zpk", NULL, p, k);
  sys.grid(1);
  sys.bode(&plot, NULL, NULL, NULL);

  return 0;
}
```
(2) Root Locus Design

```c
int rlocus(class CPlot *plot, array double &r,
            array double &kout,
            /* array double &k */);
```
- **Plot and plot the root locus of a system**
- **plot** Pointer to an existing object of class CPlot
- **r** Array of reference containing the roots corresponding to the gains in array **kout**
- **kout** Array of reference containing the output gains
- **k** Array of reference containing the user-defined gains

```c
int rlocfind(array double &k, array double &poles,
             /* array double complex p[: ] */);
```
- **Find the gains that make the closed-loop system poles move to the user-defined poles as close as possible**
- **k** Array of reference containing the output gains
- **poles** Array of reference containing the closed-loop system poles corresponding to the gains in array **k**
- **p** Array of reference containing the user-specified poles

```c
int sgrid(int flag, /* array double z[: ], array double w[: ] */);
```
- **Generate an s-plane grid for constant damping factors and natural frequencies.**
- **flag** Integer to turn on/off the sgrid.
- **z** Computational array containing damping factors at which the grid lines are plotted.
- **w** Computational array containing natural frequencies at which the grid lines are plotted.

```c
CControl *feedback(CControl *sys2, /* array int feedin[:], array int feedout[:], int sign */);
```
- **Feedback connection of two LTI models.**
- **sys2** Pointer to the class **CControl** representing the system to be connected in the feedback path
- **feedin** One-dimensional computational array containing indices specifying which inputs are involved in the feedback loop.
  For SISO systems, the default value is \{1\}.
- **feedout** One-dimensional computational array containing indices specifying which outputs are involved in the feedback loop.
  For SISO systems, the default value is \{1\}.
- **sign** Integer indicating every element of input is a positive feedback or a negative one. The default value is -1, denoting a negative feedback.
Example 2: Create the Root Locus for a System

Create the root locus for the following system. The grid line is plotted at $\zeta = 0.4$.

$$H(s) = \frac{1}{(s + 1)(s + 2)(s + 3)}$$

Program:

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

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

    array double zeta[1] = {0.4};
    array double omega[1] = {5};
    CControl sys;
    CPlot plot;

    sys.model("zpk", NULL, p, k);
    sys.sgrid(1, zeta, omega);
    sys.rlocus(&plot, NULL, NULL);

    return 0;
}
```

Output:

![Root Locus Graph](image)
Example 3: Find the Gain Corresponding to the Pole Selected from the Root Locus

For the system below, find the gain corresponding to the pole, \(-0.8 + j1.84\), which locates at the intersection of the root locus and the grid \((\zeta = 0.4)\) as shown above.

\[
H(s) = \frac{1}{(s+1)(s+2)(s+3)}
\]

Program:

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

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

    // selected pole from the root locus
    array double complex spole[1] = {complex(-0.8, 1.84)};

    // output closed-loop poles
    array double complex clpole[3];

    // output gain
    array double gain[1];

    CControl sys;

    sys.model("zpk", NULL, p, k);
    sys.rlocfind(gain, clpole, spole);
    printf("gain: %f\n", gain);
    printf("closed-loop poles: %f\n", clpole);
    return 0;
}
```

Output:

gain: 11.660785

closed-loop poles: complex(-4.414452,0.000000) complex(-0.792774,1.836351)
complex(-0.792774,-1.836351)
Example 4: Find the Step Response of a Closed-Loop System

Find the step response for a unity feedback system with a plant as shown below.

\[ H(s) = \frac{11.66}{(s + 1)(s + 2)(s + 3)} \]

Program:

```c
#include <control.h>

int main() {
    double k = 11.66;
    array double complex p[3] = {complex(-1, 0),
                                complex(-2, 0),
                                complex(-3, 0)};
    array double num[1] = {1};
    array double den[1] = {1};
    double tf = 20;
    CControl sys1, sys2, *sys3;
    CPlot plot;
    sys1.model("zpk", NULL, p, k);
    sys2.model("tf", num, den);
    sys3 = sys1.feedback(&sys2);
    sys3->grid(1);
    sys3->step(&plot, NULL, NULL, NULL, tf);
    return 0;
}
```

Output: