Bode Plots and System Identification

Phoebe Hoidn, Pierre–André Chevalier, Andreas Stahel

26th August 2016

1 Introduction

A spring mass system with an external force $f_0(t)$ may be described by the differential equation

$$m\,\ddot{y}(t) + \gamma\,\dot{y}(t) + k\,y(t) = f_0(t)$$

where m is the mass, γ the friction coefficient and k the spring constant. The equation can be standardized by dividing by m:

$$\ddot{y}(t) + \frac{\gamma}{m}\,\dot{y}(t) + \frac{k}{m}\,y(t) = \frac{1}{m} f_0(t)$$

With the notations $2\,\alpha = \frac{\gamma}{m}$, $\omega_0^2 = \frac{k}{m}$ and $f(t) = \frac{1}{m} f_0(t)$, the differential equation can be simplified to

$$\ddot{y}(t) + 2\,\alpha\,\dot{y}(t) + \omega_0^2\,y(t) = f(t) \tag{1}$$

Let us now apply the Laplace transformation to the equation, using the notation $Y(s) = \mathcal{L}[y(t)](s)$ and $F(s) = \mathcal{L}[f(t)](s)$ and assuming vanishing initial conditions $y(0) = \dot{y}(0) = 0$. We obtain:

$$s^2 Y(s) + 2\alpha s Y(s) + \omega_0^2 Y(s) = F(s)$$

Solving for $Y(s)$ leads to

$$Y(s) = \frac{1}{s^2 + 2\alpha s + \omega_0^2} \cdot F(s)$$

The transfer function $G(s)$ for this system is

$$G(s) = \frac{1}{s^2 + 2\alpha s + \omega_0^2}$$

Replacing the variable s by $i\omega$ we obtain the complex transfer function as function of the angular velocity ω.

$$\begin{aligned} G(i\,\omega) &= \frac{1}{(i\omega)^2 + 2\alpha i\omega + \omega_0^2} = \frac{1}{-\omega^2 + 2\alpha i\omega + \omega_0^2} \\ &= \frac{(\omega_0^2 - \omega^2) - 2\alpha i\omega}{(\omega_0^2 - \omega^2)^2 + (2\alpha\omega)^2} \\ &= \frac{\omega_0^2 - \omega^2}{(\omega_0^2 - \omega^2)^2 + (2\alpha\omega)^2} - i \cdot \frac{2\alpha\omega}{(\omega_0^2 - \omega^2)^2 + (2\alpha\omega)^2} \end{aligned}$$

The norm of this complex transfer function is

$$|G(i\omega)| = \frac{1}{\sqrt{(\omega_0^2 - \omega^2)^2 + (2\,\alpha\,\omega)^2}} = \frac{1}{\sqrt{(\omega^2 - \omega_0^2)^2 + 4\,\alpha^2\,\omega^2}}$$

If the external force is an oscillation of the form $f(t) = U\cos(\omega t)$, we find the amplitude of the response signal as a function of the frequency of the input by multiplying the norm above with the amplitude U of the force, that is:

$$A(\omega) = \frac{U}{\sqrt{(\omega^2 - \omega_0^2)^2 + 4\alpha^2\omega^2}} \tag{2}$$

For the same input, the phase $\phi(\omega)$ of the response in function of the frequency is given by

$$\phi(\omega) = \arctan\left(\frac{2\alpha\omega}{\omega^2 - \omega_0^2}\right)$$

The phase does not depend on the amplitude of the input. But if the input has already a phase ϕ_0, we have to add this phase to the given phase $\phi(\omega)$.

2 Some Plot Commands in *Octave* or MATLAB

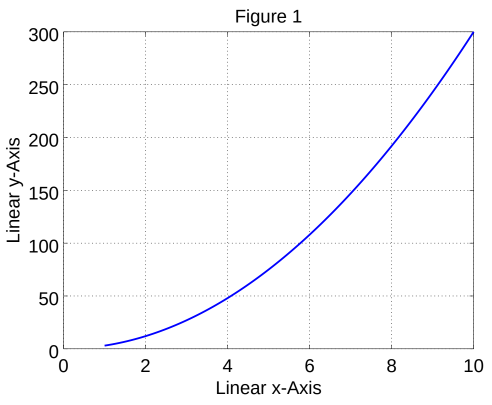
The purpose of this section is to familiarize you with a few plot commands in *Octave*/Matlab, such that you can solve the task in the subsequent sections. All examples and codes work with *Octave* and MATLAB. If you are already familiar with this topic you may skip this section.

2.1 Different scale systems

A curve can be plotted in different scale systems. The most important scales are: linear-linear, log-linear, linear-log and log-log. We show here how to plot a simple function in *Octave*/MATLAB and we give some comments about the different scale systems to be used.

For this purpose we begin to examine the simple quadratic function $y = 3\,x^2$. Plotting the graph of this function on the interval $[1, 10]$ and choosing the linear-linear scale system, we obtain clearly a piece of a parabola. The *Octave*/MATLAB function `plot()` solves the task. The code for this is given below and the Figure 1 shows the resulting graph.

```
x = linspace(1,10,100) ;
y = 3.0*x.^2 ;
plot(x,y)
xlabel('Linear x-Axis') ;
ylabel('Linear y-Axis') ;
title('Figure 1') ;
grid on
```

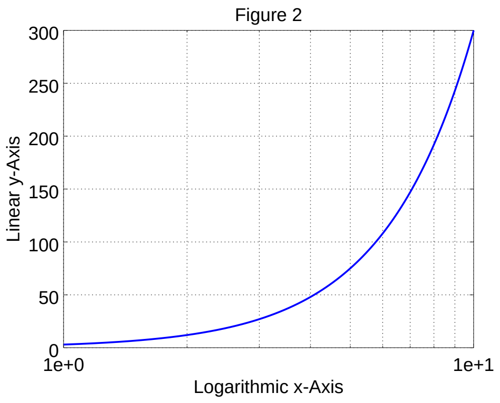


Let us now apply the logarithmic substitution $u = \log_{10}(x)$. We find

$$y = 3\,x^2 = 3 \cdot \left(10^{\log_{10}(x)}\right)^2 = 3 \cdot (10^u)^2 = 3 \cdot 10^{2u}$$

In the log-linear scale system, the given power function turns into a exponential function, see Figure 2. The log-linear plot is carried out by the command `semilogx()`.

```
semilogx(x,y)
xlabel('Logarithmic x-Axis') ;
ylabel('Linear y-Axis') ;
title('Figure 2') ;
grid on
```



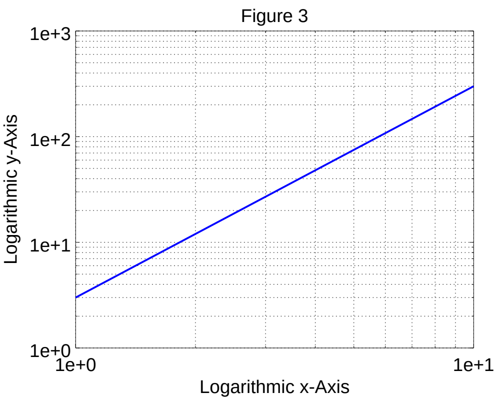
Similarly, it is possible to make a linear-log plot by using the command `semilogy()`. This plot is of no interest for our example.

For the log-log plot, we have first to apply the log on the both sides of the equation $y = 3x^2$, taking finally $u = \log_{10}(x)$ and $v = \log_{10}(y)$:

$$\begin{aligned} \log_{10}(y) &= \log_{10}\left(3\,x^2\right) = \log_{10}(3) + \log_{10}\left(x^2\right) \\ &= \log_{10}(3) + 2\,\log_{10}(x) \\ v &= \log_{10}(3) + 2\,u \end{aligned}$$

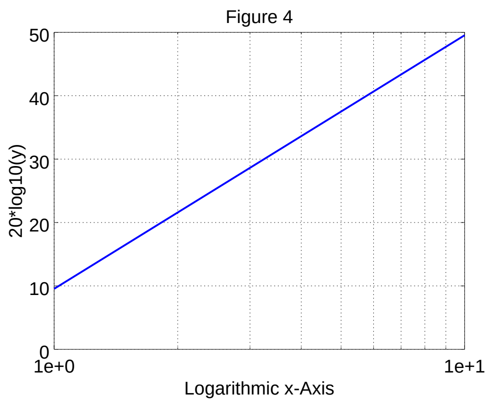
The new function v is a linear function of the new variable u and the corresponding graph is a straight line. The slope of the line (here 2) is given by the exponent in $y = 3\,x^2$ and the intersection of the line with the vertical axis gives the log of the coefficient (here 3). The log-log plot is produced by the *Octave* command `loglog()`, see Figure 3.

```
loglog(x,y)
xlabel('Logarithmic x-Axis') ;
ylabel('Logarithmic y-Axis') ;
title('Figure 3') ;
grid on
```



When amplitudes of signals are examine one often used a logoaritmic scale, i.e. the "Decibel" scale [dB]. If y is the amplitude, the corresponding value in [dB] is given by $v = 20\log_{10}|y|$, see Figure 4.

```
semilogx(x,20*log10(y))
xlabel('Logarithmic x-Axis') ;
ylabel('20*log10(y)') ;
title('Figure 4')
grid on
```



2.2 Basic Plots with *Octave*/MATLAB

This section gives some basic command to generate plots. If you are already familiar with MATLAB or *Octave* you can just glance over the presented commands.

The basic command `plot()` in *Octave* (and MATLAB) requires two arguments: a vector with the x–coordinates of the points and a second vector with the y–coordinates. The optional third argument is a **format string** that allows you to choose the color of the line and the style of the plot. For example, the format string `'r+'` will generate a red plot, the points are marked by + and with `legend('cos(x)')` the graph will be labeled by $\cos(x)$.

If no format string is given, *Octave* will connect the points by lines and choose a color. Consult the manual or `help plot` to get further information. There are many more options available to improve your graphics:

- With `grid on` (resp. `grid off`) you can turn a background grid on and off.
- With the command `title()` you can set a title for the graph.
- With the commands `xlabel()` and `ylabel()` you can choose labels for the two axes.
- With the command `legend()` you can put labels on the graphs.
- With the command `axis()` you can choose the domain to be displayed and the aspect ratio.
- With the command `text()` you can place an arbitrary text within the graphics.
- The command `clf` will clear all options and reset the graphics into default mode.

As an illustrative example consider the code below and the results in Figure 5.

```
x = -1:0.05:10; y1 = sin(x); y2 = cos(x);
clf
figure(1);  % use a first graphics window
plot(x,y2,'r+')
legend('cos(x)')

figure(2); % use a second graphics window
plot(x,y2,'r+',x,y1,'b+-', x,y1.*y2)
legend('cos(x)','sin(x)','sin(x)*cos(x)')
title('Some Plots');  grid on
xlabel('Distance'); ylabel('Amplitude');
```

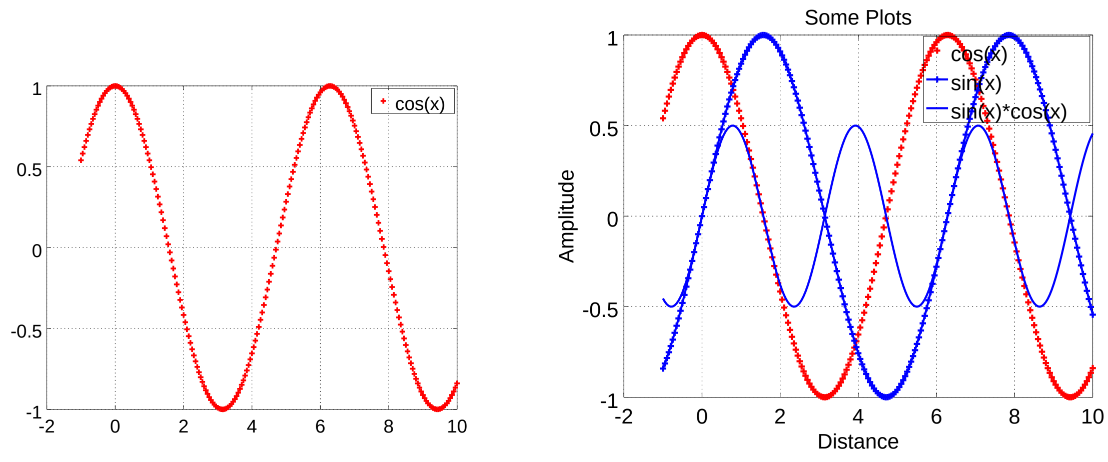


Figure 5: A basic plot of the cos–function and some combinations

With the command `print()` you can save the current print into a file in many different formats. See `help print` for details.

- To write the current graph in the `png` format (Portable Network Graphics) into a file `CosSin.png` simply use:

```
print('CosSinPlot.png')
```

- To generate a PDF file you can use (*Octave* only)

```
print('CosSinPlot.pdf','-dpdfwrite')
```

2.3 Logarithmic Plots

Octave provides the three commands `semilogx()`, `semilogy()` and `loglog()` to generate logarithmic plots. If a transfer function is given on the domain $1 = 10^0 \leq \omega \leq 10^4$ by

$$G(i\,\omega) = \frac{i\,\omega}{(100 - i\,\omega)\,(\omega^2 - 10^4 + i\,30\,\omega)^2}$$

```
om = logspace(0,4,201);
g  = om./((100-i*om).*((om.^2-1e4+i*30*om).^2));
```

The results of the commands `plot(om,abs(t))` and `plot(om,angle(t))` are certainly not useful, since a linear scale for the frequency axis is used. The result of

```
semilogx(om,abs(g))
```

shows the resonance behavior and with

```
loglog(om,abs(g))
```

we even see the shape of the Bode plot. But we absolutely have to use a Decibel scale for the amplitude to obtain a satisfying result. This is coded in *Octave*/`MATLAB` and the result is shown in Figure 6.

```
clf
semilogx(om,20*log10(abs(g)))
grid on
title('A Bode Plot')
xlabel('frequency [rad/sec]')
ylabel('amplitude [dB]')
```

A phase plot is generated by

```
semilogx(om,angle(g))
```

or by

```
semilogx(om,angle(g)*180/pi)
xlabel('frequency [rad/sec]')
ylabel('phase [degree]')
```

if you insist on degree instead of radian.

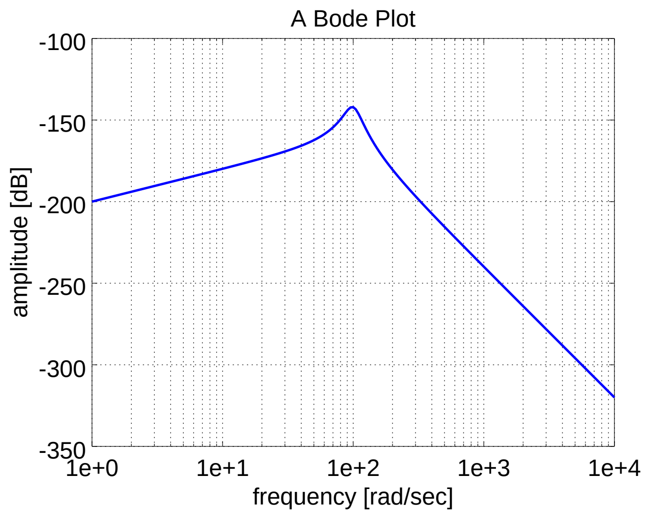


Figure 6: A first Bode plot

3 From Measurements to Plots

An electrical LRC circuit can be modeled by the differential equation

$$L\,\ddot{q}(t) + R\,\dot{q}(t) + \frac{1}{C}\,q(t) = u(t)$$

The function $q(t)$ decribes the electrical charge on the capacity and $u(t)$ is the external voltage source. Mathematically, this problem is identical to the above mechanical problem. The discussion above can easily be adapted for the parameters L, R and C in place of m, γ and k. As an example, the voltage source could be $u(t) = A\cos(\omega t)$. With a slight adaptation, the same equation can be formulated for the current $i(t)$. Mathematically there is no difference.

Such an electrical circuit was examinated in a lab. The amplitude of the output signal (for current) and the phase were measured in dependance of the frequency of the input (sinusoidal voltage source). The measured data were stored in a file named `Transfer1.txt` . The file contains 3 columns of data. The first column lists the frequency [in Hz], the second column the measured amplitude and the third column gives the phase angle [in degrees].

3.1 Reading the data from the file

Now we want to read the data from this file into *Octave* and then display the results. The code below applies the following steps:

- load the data stored in the file `Transfer1.txt` .
- Store the information on frequency, amplitude and angle in separate variables.

```
data = load('Transfer1.txt');
freq = data(:,1); amp = data(:,2); angle = data(:,3);
```

3.2 Generating the plots

You are expected to write code for the following tasks:

1. Generate a plot of the amplitude as function of the frequency.
2. Generate a plot of the angle as function of the frequency.
3. Generate the Bode plot for the amplitudes.

Each plot should have the correct labels on the axis a title and all three plot should be visible on the screen. Hints: `xlabel()`, `ylabel()`, `title(), figure()`.

4 System Identification with the Help of Linear Regression

In this section we use linear regression to determine the parameters of a system. This is a necessary step to determine the quality factor (Q factor) of a resonator. Equation (2) implies that

$$\frac{1}{(A(\omega))^2} = \frac{1}{U^2}\,\left((\omega^2-\omega_0^2)^2 + 4\,\alpha^2\,\omega^2\right)$$

If we consider this as a function of $u = \omega^2$ we find

$$\begin{aligned} g(u) &= \frac{1}{U^2}\,\left((u-\omega_0^2)^2 + 4\,\alpha^2\,u\right) \\ &= \frac{1}{U^2}\,\left(u^2 + (4\,\alpha^2 - 2\,\omega_0^2)\,u + \omega_0^4\right) \;=\; a_2\,u^2 + a_1\,u + a_0 \end{aligned}$$

where

$$a_2 = \frac{1}{U^2} \quad , \quad a_1 = \frac{4\,\alpha^2 - 2\,\omega_0^2}{U^2} \quad \text{and} \quad a_0 = \frac{\omega_0^4}{U^2}$$

Thus we have a polynomial of degree 2. This is confirmed by the graph generated by

```
plot(freq.^2,1./(amp.^2))
```

We can use linear regression to determine the optimal values of the three parameters a_0, a_1 and a_2. With the help of these we can compute the values of U, ω_0 and α.

To apply linear regression to our above data we first have to convert the frequency to angular velocity and compute $u = \omega^2$.

```
omega = freq*2*pi;   u = omega.^2;
```

Then we have to construct the matrix

$$\mathbf{F} = \begin{bmatrix} 1 & u_1 & u_1^2 \\ 1 & u_2 & u_2^2 \\ 1 & u_3 & u_3^2 \\ & \vdots & \\ 1 & u_n & u_n^2 \end{bmatrix}$$

```
n  = length(omega);
F  = ones(n,3);      F(:,2) = u;     F(:,3) = u.^2;
```

Then we use the command `LinearRegression()` to determine the optimal parameter values and their variances.

```
[a,yVar,residual,aVar] = LinearRegression(F,1./(amp.^2));
parameters   = a'
StandardDev  = sqrt(aVar)'
```

Then we compute the parabola generated by the linear regression and the measured parabola in the same figure.

```
y = F*a;
plot(u,1./(amp.^2),u,y)
```

The result in Figure 7 is obviously of a very poor quality.[1]

This is caused by the illconditioning of the problem. The matrix

$$\mathbf{F}^T \cdot \mathbf{F} \approx \begin{bmatrix} 1.01\cdot 10^2 & 5.24\cdot 10^{10} & 4.51\cdot 10^{19} \\ 5.24\cdot 10^{10} & 4.51\cdot 10^{19} & 4.62\cdot 10^{28} \\ 4.51\cdot 10^{19} & 4.62\cdot 10^{28} & 5.15\cdot 10^{37} \end{bmatrix}$$

has entries of vastly different magnitudes. This is caused by the entries of 1 and ω^4 in the matrix $\mathbf{F}$. If we rescale the frequency axis such that only small values occur, then the problem will disappear. Thus we have to add a line to the code above.

[1]With recent versions of *Octave* and `MATLAB` the result is not obviously wrong anymore, but the problem persists. To obtain obviously wrong results use the matrix $\mathbf{F}^T \cdot \mathbf{F}$, instead of the QR factorization, to solve the linear regression problem.

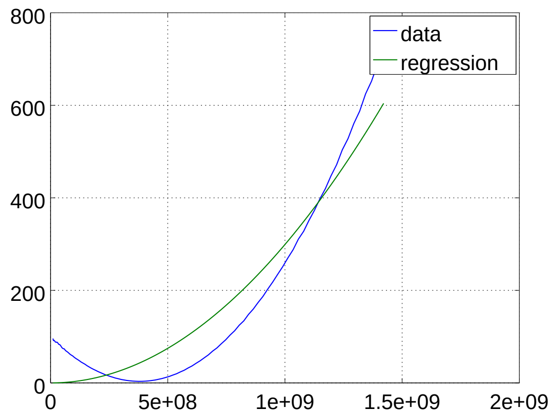


Figure 7: Measured parabola data and obviously wrong reconstruction by generated by linear regression

```
scalefactor = 1000;  %% factor=1 will fail , due to ill conditioning
freq = freq/scalefactor;
```

Taking the shorter notation β for the scale factor, we can express the polynomial function $g(\omega^2)$ for the scaled variable $\frac{\omega}{\beta}$:

$$g(\omega^2) = a_2\,\omega^4 + a_1\,\omega^2 + a_0 = a_2\,\beta^4\left(\frac{\omega}{\beta}\right)^4 + a_1\,\beta^2\left(\frac{\omega}{\beta}\right)^2 + a_0$$

The coefficients of the scaled polynomial are not the same as in the original, except a_0. The call of `LinearRegression()` will compute the optimal values for $a_2\,\beta^4$ and $a_1\,\beta^2$ and thus we have finally to adapt the results by rescaling the coefficients:

```
[a(1) a(2)*scalefactor^2 a(3)*scalefactor^4]
[sqrt(aVar(1)) sqrt(aVar(2))/scalefactor^2 sqrt(aVar(3))/scalefactor^4]
```

The results obtained by the above code allow to identify the parameters α and ω_0^2 in Equation (1) and thus if one out of m, μ or k is known in the original problem, the others can be determined.