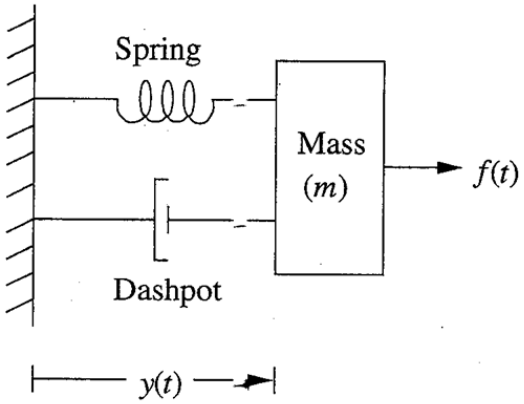
ELEG 305 SIGNALS AND SYSTEMS
COMPUTER ASSIGNMENT #2

Second-Order Systems and Bode Plots
Due Tuesday May 14
(50 points)

The objectives are to (1) get an understanding of the behavior of systems described by second-order differential equations and (2) learn how to generate Bode plots.

Please read O&W, pp. 448-460, on first- and second-order systems and Bode plots, and look at the slides on "First- and Second-Order Systems" that have been uploaded to Canvas. Hand-in your computer code, as well as all of the requested plots and answers to all of the questions.



A Second-Order Shock Absorber

The second-order, linear, constant-coefficient, differential equation of the form

$$\frac{d^2y(t)}{dt^2} + 2\zeta\omega_n\frac{dy(t)}{dt} + \omega_n^2 y(t) = \omega_n^2 x(t) \qquad (1)$$

can be used to model many physical systems. Two examples are RLC electrical circuits and mechanical systems like the shock absorber illustrated in the figure above, which consists of a spring and a viscous dashpot. The form of (1) is very useful because ω_n and ζ are directly related to important properties of these systems; ω_n is called the *undamped natural frequency* and ζ is called the *damping ratio*.

For the shock-absorber system illustrated in the figure, the signal $f(t)$ is the external force applied to the mass m, and $y(t)$ is the horizontal displacement from the equilibrium point at which the spring exerts no restorative force (the effects of gravity will be ignored). The restorative force is given by

$$F_{\text{spring}} = -ky$$

and the force exerted by the dashpot is given by

$$F_{\text{dash}} = -v\frac{dy}{dt}$$

The parameter k is called the spring constant and ν is a measure of the viscosity of the dashpot. Summing the forces acting on the mass connected to the shock absorber and, for convenience, substituting *f(t) = kx(t)* for the external force, we obtain

$$\frac{d^2y(t)}{dt^2} + \left(\frac{\nu}{m}\right)\frac{dy(t)}{dt} + \left(\frac{k}{m}\right)y(t) = \left(\frac{k}{m}\right)x(t) \qquad (2)$$

Comparing (1) and (2), we can see that

$$\omega_n = \sqrt{\frac{k}{m}} \quad \text{and} \quad \zeta = \frac{\nu}{2\sqrt{km}}$$

As you will see, the shock absorber acts as a *lowpass filter* on the forcing term $x(t)$, damping high-frequency fluctuations due to the road surface and allowing for a smoother ride. However, if the spring constant and the viscosity of the dashpot are not chosen carefully, the shock absorber might respond too slowly to bumps in the road, leading to a loss of control of the vehicle.

In this assignment, you will become familiar with how the values of k and ν, and hence the values of ω_n and ζ, affect the performance of the shock absorber. Since shock absorbers are usually chosen with a specific vehicle mass in mind, assume for the rest of this exercise that $m = 1$. In this case, the parameters ω_n and ζ are determined completely by the shock-absorber coefficients k and ν.

Work to be Done and Questions to Answer

(a) Consider the LTI system that satisfies (2).

i) *Analytically* determine the frequency response of this system. Also, answer the following:

ii) What is the value of $20\log_{10}|H(j\omega)|$ at $\omega = 0$?

iii) To what value does $20\log_{10}|H(j\omega)|/\log_{10}\omega$ converge for large values of ω?

The values from (ii) and (iii) are the asymptotes of the Bode plot of $|H(j\omega)|$, and will be useful for verifying and analyzing the plots you will obtain next.

Let $\zeta = 1/\sqrt{2}$. In the following, we will determine, through computer computation, how the frequency and step responses of the shock absorber change with the value of ω_n.

(b) For $k = 1$, determine the coefficients of the numerator and denominator polynomials of the frequency response and store these values in arrays b and a, respectively. Use the function `freqs(b,a,ω)` to determine $H(j\omega)$ at the frequencies contained in the array `ω = logspace(-2,2,100) (NOTE: The Python default for the third parameter is 50; change this in your program to 100).` Store these samples of $H(j\omega)$ in the array `H1`. Note that array $\boldsymbol{\omega}$ contains evenly spaced samples of $\log_{10}\omega$ rather than evenly spaced samples of ω. This is because you will eventually plot $20\log_{10}|H(j\omega)|$ versus $\log_{10}\omega$.

(c) For $k = 0.09$ and $k = 4$, determine $H(j\omega)$ at the frequency samples in $\boldsymbol{\omega}$ and store these samples in `H2` and `H3`, respectively.

(d) Use `loglog(ω, abs(H1)**20,'ro',basex=10,basey=10)` to create a Bode plot, where `abs(H1)` takes the absolute value of the values of the complex frequency

response. The exponential term `**20` is used because the Bode plot requires a plot of $20 \log_{10} |H(j\omega)|$; multiplying by a factor 20 is equivalent to adding an exponent of 20 inside the log function. (The parameter `'ro'` means the plot is made with red circles.)

i) Generate Bode plots for `H1, H2,` and `H3,`and plot them on the same set of axes.

The cutoff frequency of the shock-absorbing system can be loosely defined as the "knee" of the Bode plot of $|H(j\omega)|$. For frequencies higher than the cutoff frequency, the magnitude of the frequency response decreases rapidly with increasing ω. In addition to creating the plots, answer the following:

ii) Based on your plots, is ω_n roughly equal to the cutoff frequency of each shock absorber?
iii) Do the asymptotes of your plots agree with the values determined in (a)?

(e) For the three shock-absorbing systems given by $k = 0.09, 1, \text{and } 4$, use `step2((b,a),T=t)` to determine the *step response* of each system at the time samples contained in the array `t=linspace(0,30)`. Note that for different k parameters, the arrays `b` and `a` are different.

i) Plot the three step responses on a single set of axes.
ii) How does the rise-time, i.e., the time for the step response to first reach its steady-state value, change as a function of ω_n?
iii) How is the rise-time related to the filter bandwidth, i.e., the cutoff frequency?
iv) Would you expect a shock absorber with a small bandwidth to react quickly or slowly to rapid changes in the external force $x(t)$?

PYTHON NOTES

General Comment: There were numerous difficulties last year using/linking the libraries required for the computer assignments. **For both Python 2.7 and Python 3.6**, in order to use functions from an external library, you must download and install the library, and add the command **"`from X import *`"**, where `X` is replaced by the name of the library.

Several students reported errors such as **"`ImportError: No module named scipy`"**. This was usually due to not being able to link the library to Python. This is less of a problem in Mac OS and Linux because you can simply use the following command: **"`pip install scipy`"**. For Windows, however, there is no simple way to do this; in this case, if you continue to have difficulties, you should install Anaconda, which offers both Python 2.7 and Python 3.6. In case you have more than one Python version in MacOS and Linux, it is good practice to run 'pip install scipy' for the version you would like to work with: "python2.7 -m pip install scipy".

Function Explanations:

1. `freqs(b,a,worN)`

For the command `freqs`, the scipy library is required. To add the library to your program, you should add the following command:

```
from scipy.signal import *
```

The function `freqs(b,a,worN)` returns the filter frequency response,

$$H(j\omega) = \frac{\mathrm{b}[0] * (\mathrm{j}\omega)^{\mathrm{nb}-1} + \mathrm{b}[1] * (\mathrm{j}\omega)^{\mathrm{nb}-2} + \ldots + \mathrm{b}[\mathrm{nb}-1]}{\mathrm{a}[0] * (\mathrm{j}\omega)^{\mathrm{na}-1} + \mathrm{a}[1] * (\mathrm{j}\omega)^{\mathrm{na}-2} + \ldots + \mathrm{a}[\mathrm{na}-1]}$$

where `b` is an array of the coefficients of the *numerator* polynomial {b[0]… b[nb-1]} and `a` is an array of the coefficients of the *denominator* polynomial {a[0]… a[na-1]}. The parameter nb is the dimension of array `b`, and na is the dimension of array `a`. This function returns two parameters, so be careful to get the one you need. If you only need the second one, you can use "`_,H1 = freqs(…..)`".

Other Parameters

`worN` : This parameter is optional, and can be an integer or an array, or you can simply ignore it. If it is an integer, then the frequency response that is returned is computed at that integer number of frequency points (actual frequencies are chosen by the computer). If it is an array, then the response that is returned is computed at each frequency point of the array `worN`. *For this assignment, you must fill in an array (see Part b in the "Work to be Done" section).*

Returns – The function returns two quantities: `w` and `h`

`w` : N-dimensional array (ndarray) containing the frequencies where h was computed.

`h` : N-dimensional array (ndarray) containing the frequency response.

---

2. `linspace(start,stop,num=50,endpoint=True,retstep=False,dtype=None)`

`logspace(start,stop,num=50,endpoint=True,base=10.0,dtype=None, axis=0)`

For the commands `linspace` and `logspace`, the numpy (or pylab) library is required. To add the library to your program, you should add the following command:

```
from numpy import *
```

or

```
from pylab import *
```

a) The function `linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)` returns `num` evenly spaced values over the interval [`start,stop`].

Parameters

`start` : A scalar containing the starting value.
`stop` : A scalar containing the final value.
`num` : This parameter represents the number of values to generate, and is optional; it is either an integer value, or it can be ignored. The default value is 50.
`endpoint` : Chooses if the stop point is included in the vector. Default is True. *For this assignment, you do not need to use this parameter.*
`retstep` : If True, return (samples, step), where step is the spacing between samples. Default is false. *For this assignment, you do not need to use this parameter.*
`dtype` : The type of the output array. If dtype is not given, infer the data type from the other input arguments. *For this assignment, you do not need to use this parameter.*

Returns

`sample` : An N-dimensional floating-point array (ndarray) containing `num` equally spaced samples in the closed interval "[`start`, `stop`]" .
`step` : If retstep is false, this return value will not be present. This value is not necessary, so it is fine if retstep is not used.

b) The function `logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None, axis=0)` returns values spaced evenly ***on a log scale***. To be specific, in linear space, the returned sequence starts at **"`base **start`"** (`base` to the power `start`) and ends with **"`base**stop`"** (`base` to the power `stop`). The symbol ** is the power operator in Python.

Parameters

`start` : A floating-point scalar containing the exponent for the starting value. The sequence starts at "`base**start`".
`stop` : A floating-point scalar containing the exponent for the final value. The sequence ends at "`base**stop`".
`num` : This parameter represents the number of values to generate, and is optional; it is either an integer value, or it can be ignored. The default value is 50.
`endpoint` : Chooses if the stop point is included in the vector. Default is True. *For this assignment, you do not need to use this parameter.*

`base` : The base of the log space. We will use the default value (log base 10).
`dtype` : The type of the output array. If dtype is not given, infer the data type from the other input arguments. *For this assignment, you do not need to use this parameter.*
`axis` : The axis in the result to store the samples. Relevant only if start or stop are array-like. We are using scalar start and stop. *For this assignment, you do not need to use this parameter.*

Returns

`sample` : An N-dimensional floating-point array (ndarray) containing `num` equally spaced samples on a log scale.

---

3. `loglog(x,y,'ro',basex,basey)`

For the command `loglog`, the pylab library is required. To add the library to your program, you should add the following command:

```
from pylab import *
```

The function `loglog(x,y,'ro',basex,basey)` creates a plot using a *log scale on both axes*. It supports all the keyword arguments of function `plot()`. The input parameters `x and y` are, respectively, the x-coordinate array and the y-coordinate array (in linear space), and `basex` and `basey` indicate the bases of the logarithms. The parameter `'ro'` means the plot is made with red circles. This link contains more details about the arguments for plot(): http://matplotlib.org/users/pyplot_tutorial.html

It is important to use 'ro' instead of `ro`, otherwise the interpreter will issue errors.

---

4. `step2(system,X0,T,N,kwargs)`

For the command `step2`, the scipy library is required. To add the library to your program, you should add the following command:

```
from scipy.signal import *
```

The function `step2(system,X0,T,N,kwargs)` returns the step response of a system.
This function returns two parameters, so be careful to get the one you need.
In order to use the vector of coefficients of the linear equation, use "(numerator, denominator)" as input with the parenthesis. Since this function has optional parameters, it is important to tell the function which parameter you are using as input; instead of step2(xxxxx, t), use step2(xxxxx,T=t), where t is your time axis.

Parameters

`system` : A dual `(b,a)` describing the system; arrays `b` and `a` represent the numerator and denominator polynomial coefficients, respectively, of the system's frequency response.
`X0` : This array provides the initial state-vector, and is optional. The default is zero. *For this assignment, you do not need to use this parameter.*

`T` : This is an N-dimensional array that provides the values of time for which the step response is computed. *For this assignment,* `T` *is generated using* `linspace` *(see Part e of the "Work to be Done" section).*
N : Number of time points to compute if T is not given. Since we have T, this is not used. *For this assignment, you do not need to use this parameter.*
`kwargs`: With this we can send more arguments for the inner functions called by step2. *For this assignment, you do not need to use this parameter.*

Returns

T : Output time points. You can ignore this output if you wish.
`yout` : An N-dimensional array (ndarray) containing the step response of the system at the time values contained in `T`.