lesson2

Contents

LESSON 2

Related Assignments:
Preread + participation activities: 3.1-9; 4.1,3, read about linspace in 4.8; 5.5
After class challenge activities: 3.1-5 first, then 3.6-9
Also: hw2c

```
% reduce needless whitespace
format compact
% reduce irritations
more off
% start a diary
%diary lesson2.txt
% for me only
%echo on
```

THE PROBLEM WE WANT TO SOLVE

We want to find the frequencies (tones) of a string with one end rigidly attached and the other end flexibly attached.
It can be shown that all valid frequencies omega must satisfy the equation

```
     - k omega = tan(omega)
```

Here k is a constant depending on the string properties.
Our problem is to figure out what those valid frequencies are.

PLOT TO UNDERSTAND THE PROBLEM BETTER

Somehow we must find the solution(s) to the equation

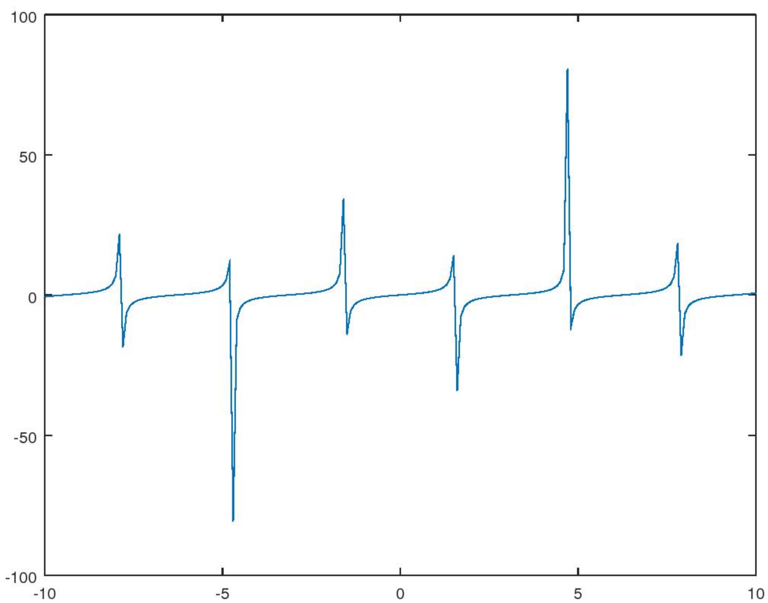
```
     - k omega = tan(omega)
```

That is not that straightforward. So maybe we should first examine the functions in the right and left hand sides by plotting them.

Plot tan(omega) for -10 < omega < 10

```
% generate 201 omega values between -10 and 10
omega=[-10:0.1:10];

% this makes omega a line of numbers:
%omega

% another way to do the same thing:
%omega=linspace(-10,10,201);
```

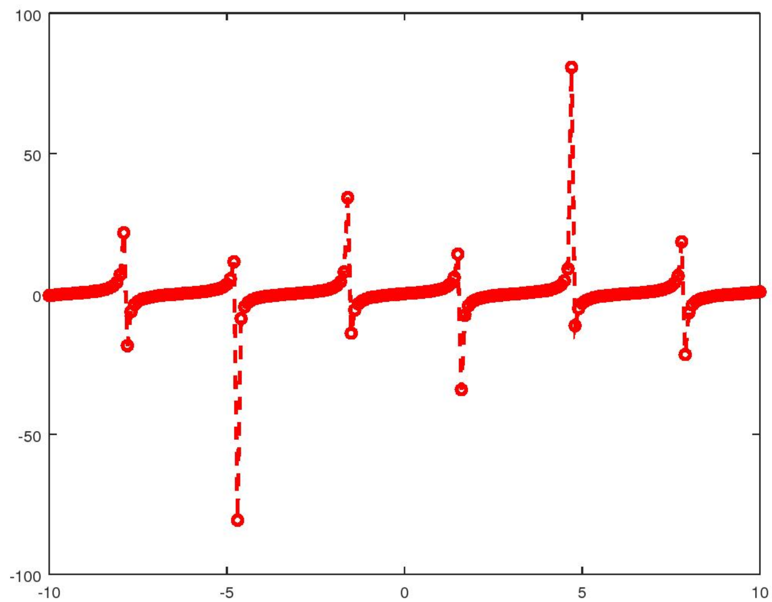
```
% it is preferable to have omega as a column, use a quote
    to do that
omega=omega';
%omega

% create f = tan(omega) values
f=tan(omega);
%f

% plot (unmaximize this window to have the plot visible)
plot(omega,f)
```



Try improving the plot

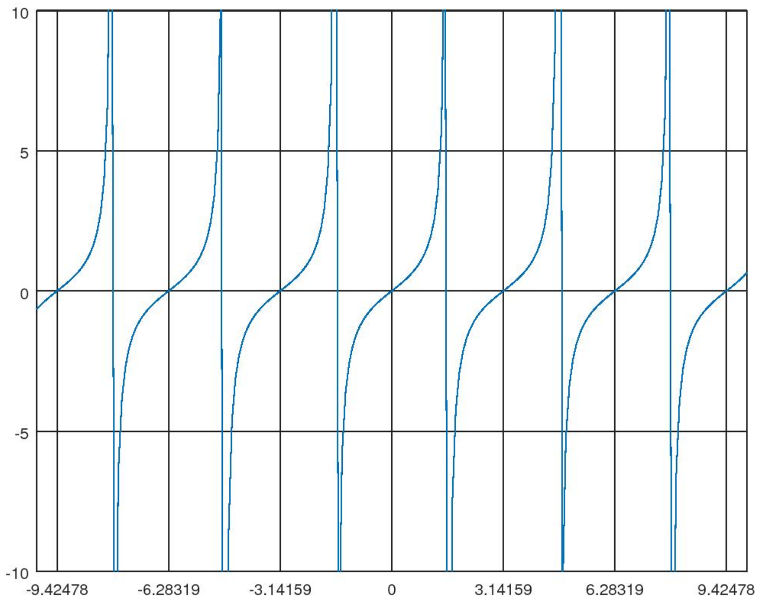
```
% to find out how to modify the plot
%help plot
% (also google 'matlab chart line properties')

% --: dashed line, o: use circle symbols, r: use a red
    line
```

```
plot(omega, f, '--or', 'LineWidth', 2)
```



Try, try, again

```
% plot straightforwardly
plot(omega, f)

% but reduce the vertical extent of the plot
axis([-10 10 -10 10])

% and add a grid
grid on

% set the tick marks at multiples of pi
%set(gca)
set(gca, 'xtick',[-3*pi:pi:3*pi])

% put the x-axis at y=0
set(gca, 'xaxislocation', 'origin')
```



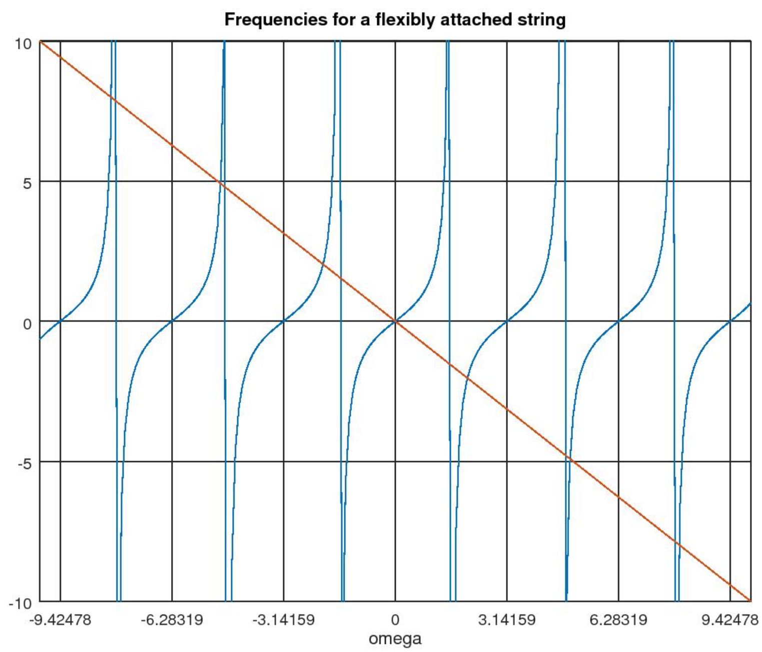
Now plot *both* -k omega and tan(omega)

Where those functions intersect, we have valid frequencies.

```
% take k = 1 for now
k=1

% add -k omega to f as a second column
f=[f -k*omega];
%f

% plot both now
plot(omega, f)

% reduce the vertical extent of the plot
axis([-10 10 -10 10])

% add a grid
grid on

% add a label on the x-axis
xlabel('omega')
```

```
% add a title
title('Frequencies for a flexibly attached string')

% set the tick marks at multiples of pi
set(gca,'xtick',[-3*pi:pi:3*pi])

% put the x-axis at y=0
set(gca,'xaxislocation','origin')
```

```
k =   1
```



FINDING ACCURATE VALUES FOR THE FREQUENCIES

To keep it simple, let's keep k=1 for now and find the lowest frequency first.

Finding the value of the lowest frequency

We now want to find the lowest positive frequency omega1 so that:

```
-k omega1 = tan(omega1)
```

We define the difference between right and left hand sides to be the error in the equation:

```
error = tan(omega) + k omega
```

This error must be zero for the correct omega1.

Create a function for the error

We will call the function 'freqEq1' and store it in a file named 'freqEq1.m'. It contains:

```
function error = freqEq1(omega)

% This function returns the error in the equation
% satisfied by the frequencies of a string with one end
% flexibly attached.  The scaled attachment flexibility k
% is assumed to be 1.
%
% Input:
%    omega: the frequency to test
% Output:
%    error: zero if omega is a correct frequency (tone)
%           of the string, nonzero if it is not.
%
% Advanced analysis taught in Analysis in Mechanical
% Engineering II shows that the equation the frequencies
% must satisfy is:
%                  - k omega = tan(omega)
% So if the frequency is not right, the error in the
% equation (difference between the right and left hand
% sides) is:
%                 error = tan(omega) + k omega

% Note that omega is in radians and do not forget the
    semi-colon
error = tan(omega) + omega;

end
```

Play a bit with the function

```
% see whether matlab can see the function
%help freqEq1
```

```
% for omega=0 the error is zero but then there is no
    sound!
freqEq1(0)

% for omega=1 the error is not zero, so omega=1 is _not_
% a frequency of vibration of this string
freqEq1(1)
% looking at the graph, 2 seems to be close to the
    correct lowest frequency
%set(gca,'xtick',[-10:2:10])
freqEq1(2)

% how about 1.9 or 2.1?
freqEq1(1.9)
freqEq1(2.1)
```

```
ans = 0
ans =  2.5574
ans = -0.18504
ans = -1.0271
ans =  0.39015
```

Let matlab find the value for us

Matlab finds zeros ('roots') of functions using the 'fzero' library function.

```
% Get a clue how to use fzero first
%help fzero

% tell fzero to start searching for a zero in freqEq1
    near omega = 2
fzero('freqEq1',2)

% suppose we start at .5 pi
fzero('freqEq1',.5*pi)
% Oops.  In fact we could have ended up _anywhere_.

% The _safe_ way is to tell fzero to search in a small
    interval
% that contains only the root we want, like from 1.9 to
    2.1:
omega1=fzero('freqEq1',[1.9 2.1])
```

```
ans =  2.0288
ans =  1.5708
omega1 =  2.0288
```

Find many more frequencies

If you want to find more frequencies, it would be simplest to start fzero at odd values of pi/2. But that does not work because the tangent is infinite there. But suppose we multiply the equation

```
    0 = tan(omega) + k omega
```

by cos(omega):

```
    0 = sin(omega) + k cos(omega)
```

Then there is no longer a singularity at any omega.
So we define a new function:

```
function error = freqEq1Mod(omega)

% This function returns the error in the equation
% satisfied by the frequencies of a string with one end
% flexibly attached.  The scaled attachment flexibility k
% is assumed to be 1.
%
% Input:
%    omega: the frequency to test
% Output:
%    error: zero if omega is a correct frequency (tone)
%           of the string, nonzero if it is not.
%
% Advanced analysis taught in Analysis in Mechanical
% Engineering II shows that the equation the frequencies
% must satisfy is:
%                  - k omega = tan(omega)
% However, the tan is infinite at any odd amount of pi/2,
% and that is a numerical problem.  So we multiply both
% sides by the cosine:
%              - k omega cos(omega) = sin(omega)
% Then if the frequency is not right, the error in the
% equation (difference between the right and left hand
% sides) is:
%            error = sin(omega) + k omega cos(omega)

% Note that omega is in radians and do not forget the
   semi-colon
error = sin(omega) + omega*cos(omega);

end
```

```
% let's try it out
omega1=fzero('freqEq1Mod',0.5*pi)
% yes, that produced the correct root now

% seems to work OK:
omega2=fzero('freqEq1Mod',1.5*pi)

% try the next one
omega3=fzero('freqEq1Mod',2.5*pi)
% the last is just a little bit bigger than 2.5*pi
2.5*pi

% at some point, the frequencies will get so close to the
% odd multiple of pi/2 that we can ignore the difference.
```

```
omega1 =   2.0288
omega2 =   4.9132
omega3 =   7.9787
ans =   7.8540
```

HOW ABOUT IF K IS NOT 1??

Surely we cannot create a new function for every possible value of k.
So we must create a function that accepts k as an input argument. Then we can use that function for *any* k we want:

```
function error = freqEq(omega,k)

% Function used to find the natural frequencies of a
% string that has one end rigidly attached to the musical
% instrument but the other end attached to a flexible
% strip.
%
% Input:
%    omega: The natural frequency in radians
%    k:     The bending flexibility of the strip
%    Both are suitably nondimensionalized in a way not
%    important here.
%
% Output:
%    error: If error is zero, then the frequency is a
%           valid one for that value of k.  Note that a
%           string can vibrate with infinitely many
%           frequencies (theoretically at least)
%
```

```
% Advanced analysis taught in Analysis in Mechanical
% Engineering II shows that the equation the frequencies
% must satisfy is:
%                 - k omega = tan(omega)
% However, the tan is infinite at any odd amount of pi/2,
% and that is a numerical problem.  So we multiply both
% sides by the cosine:
%             - k omega cos(omega) = sin(omega)
% Then if the frequency is not right, the error in the
% equation (difference between the right and left hand
% sides) is:
%           error = sin(omega) + k omega cos(omega)

% Note that omega is in radians and do not forget the
    semi-colon
error = sin(omega) + k*omega*cos(omega);

end
```

But how do we tell fzero what k to use??

There is no way to tell fzero to use a second input argument in a function. Instead we must tell matlab itself to provide fzero a new function that has the desired value of k in it.

The convenient way to do that is to tell matlab to create an anonymous (nameless) function (x) of x that for given x returns freqEq(x,k), with k the value we want. That can be done as '@(x) freqEq(x,k)'. (The "@" is *not* a function name. It tells matlab to create a "handle" to that function for fzero to get hold of it.)

```
% let's first try it for the current value k = 1
omega1=fzero(@(x) freqEq(x,k),0.5*pi)
omega2=fzero(@(x) freqEq(x,k),1.5*pi)
omega3=fzero(@(x) freqEq(x,k),2.5*pi)

% how about another value of k now?
k=2;

% If you want others to use this m-file, and select their
     own value
% of k, uncomment the next line by removing the %:
%k=input('Please enter a value for k, like 2: ')
% However, this will prevent publishing on at least
    Octave.

% notify about thenew k-value
disp(['New k-value: ', num2str(k)])
```

```
% compute the new frequencies
omega1=fzero(@(x) freqEq(x,k),0.5*pi)
omega2=fzero(@(x) freqEq(x,k),1.5*pi)
omega3=fzero(@(x) freqEq(x,k),2.5*pi)
```

```
omega1 =  2.0288
omega2 =  4.9132
omega3 =  7.9787
New k-value: 2
omega1 =  1.8366
omega2 =  4.8158
omega3 =  7.9171
```

PRINT OUT THE FREQUENCIES NICELY

The 'fprintf' function allows you to print out numbers in your own way.
Function fprintf uses the following symbols:
%i: integer (also %d)
%f: floating point number
%[PrintPositions][.DigitsBehindPoint]f
%e: exponential notation
%g: either %f or %e

```
% the first %f gets replaced by k, the second by omega
fprintf('The lowest frequency of vibration for k = %f is:
    %f\n',k,omega1)
% the \n ends the line (this is *not* automatic)

% to print out k as an integer, use %i instead of %f
fprintf('The lowest frequency of vibration for k = %i is:
    %f\n',k,omega1)

% note that %f performs rounding
fprintf('The lowest frequency of vibration for k = %i is:
    %.3f\n',k,omega1)
fprintf('The lowest frequency of vibration for k = %i is:
    %6.3f\n',k,omega1)

% to just print the number, without 'omega1=', use disp
disp(omega1)
```

```
The lowest frequency of vibration for k = 2.000000 is:
    1.836597
The lowest frequency of vibration for k = 2 is: 1.836597
```

```
The lowest frequency of vibration for k = 2 is: 1.837
The lowest frequency of vibration for k = 2 is:  1.837
 1.8366
```

ADDITIONAL REMARKS

To find the smallest or largest value of a function instead of a zero value, you could find a zero for the derivative. Alternatively, you can directly search for a minimum by using function 'fminbnd' instead of 'fzero'. To search for a maximum, search for a minimum of minus the function.
If you have more than one variable, things get messier. Try 'fzero' or 'fminunc'.

End lesson 2