This tutorial will show how to use Matlab to generate coefficients for simple FIR filters.

It is a requirement to have the signal processing toolbox which may or may not be included with some licenses of Matlab.

For the purpose of this example I will be designing a filter with the following specifications:
- 8 KHz Sampling Frequency
- Passband from 0 to 1 KHz
- Stopband from 2 to 4 KHz
- 1 dB Passband Ripple
- Approximately 0 dB passband attenuation
- Approximately -40 dB stopband attenuation

Let’s Begin!

We are going to be using a graphical tool from the signal processing toolbox called “filterbuilder.”

Open Matlab and type the following command:

```
>> filterbuilder
```

You should see the following GUI:
Here we select the type of filter that we would like to design. For the purpose of this example I will go ahead and choose Lowpass. Press OK once you have selected your filter type.

You will be shown another window. For a lowpass filter it should look like the following:

The following changes need to be made:

- **Frequency units**: Since we are processing sound, I set this value to Hz.
- **Input Fs**: This is the sample rate. For this example I use the value 8000.
- **Fpass**: This is the right edge of your passband. I use 1000 for this example.
- **Fstop**: This is the left edge of your stopband. I use 2000 for this example.
- **Apass**: Passband Attenuation – This is the tolerable amount of ripple centered about 0 allowed in the passband. For this example I use 1 dB.
- **Astop**: Stopband Attenuation – This is the upper limit of the stop band attenuation.
After all of the listed changes, the window should now look like the following:

![Lowpass Design window](image)

Ensure you click “Apply” in the bottom right corner after you make these changes in order to update the filter.

In order to check that we have the correct filter, we can now view its frequency response. Click the button in the top right of this window that says “View Filter Response”
My filter looks like this:

![Magnitude Response (dB)](image)

We see that this filter meets all of the required specifications.

Now, we need to get our filter coefficients.

When you hit “Apply” previously you should have gotten the following message in the Matlab terminal:

```
>> The variable 'H1p' has been exported to the command window.
```

This created an object saved as H1p (if you did not rename it).

To extract our coefficients we could run the following code:

```
>> myFilterCoeffs = H1p.Numerator;
```
Now if we check what the value of myFilterCoeffs...

```matlab
myFilterCoeffs =

Columns 1 through 7
0.0024  0.0260  0.0447  0.0233 -0.0485 -0.1096 -0.0711

Columns 8 through 13
0.0846  0.2641  0.3427  0.2809  0.1490  0.0435
```

There are our 13 coefficients!

Now let's test this out!

Let's create two sinusoids and filter them with our shiny new filter!

The following code creates 10000 samples of 2 sinusoids with frequencies of 500 and 3000 at a sample rate of 8000.

```matlab
>> t = 1:10000;
>> data1 = sin(2*pi*t*500/8000);
>> data2 = sin(2*pi*t*3000/8000);
```

Convince yourself that these are actually sinusoids by plotting them.

Here I plot 100 samples of our 500Hz sinusoid. I use the following code:
Note: You don't want to plot all 10000 samples because you can't see anything at that point!

```matlab
>> plot(data1(5000:5100))
```
We see the variables in our workspace have upper and lower limits of 1 and -1 respectively. This is exactly what we expect considering these are plain sinusoids.

```
data1       <1x10000 double>   -1   1
data2       <1x10000 double>   -1   1
```

Using the filter command we can pass our data and our filter object called Hlp (unless you renamed it) and obtain filtered data.

So after running the following code:

```
>> data1filt = filter(Hlp, data1);
>> data2filt = filter(Hlp, data2);
```

We see the following in our workspace:

```
data1filt   <1x10000 double>  -0.9693  0.9693
data2filt   <1x10000 double>  -0.0286  0.0795
```

We see that our 500Hz sinusoid is attenuated only slightly but the 3000 Hz sinusoid is attenuated dramatically! This is exactly how our filter should behave.

I hope this helps!