**MATLAB Signal Processing 101: Your First Filter Design Adventure**

**Introduction**

Hey aspiring engineers and signal processing enthusiasts! 🚀 Ever wondered what it feels like to bring those textbook theories to life? Well, get ready to embark on your very first filter design journey using MATLAB. No, we're not diving into advanced masterclasses just yet; this is all about getting your hands dirty for the first time and turning those theoretical concepts into tangible, real-world designs.

Picture this: you've soaked in the classroom wisdom, nodded along to the theories, and now, it's your chance to put that knowledge into action. I'm here to be your guide as we navigate the maze of filter design, making the seemingly complex world of MATLAB feel like a playground rather than a puzzle.

So, if you're ready to break the seal on your MATLAB adventure, if you want to transform those theoretical lessons into practical skills, then grab your coding spirit, unleash your inner engineer, put on your thinking cap and let's dive into the world of hands-on filter design. It's not about mastering everything or solving a real world challenge just yet! Chill! it's about giving you that first taste of practical magic. Let's make MATLAB filter design your playground! 💻🔧✨

So lets get filtering. 😂 Here is the walkthrough.

* Welcome to MATLAB
* Meet the Playground - MATLAB Interface
* Saying Hello with Code
* Data Generation Magic
* Adding a Dash of Noise
* Let's Mix Things Up - Combining Data
* Visualize the Symphony - Data Plotting
* Enter the Filter Designer's Lair
* Filtering Out the Noise
* Admire the clean data and your new skillset
* The grand finale and beyond
* See you next time!

At this moment if you want to code along you should get your laptop ready and a comfortable seat! Otherwise just have fun learning.

**Welcome to MATLAB**

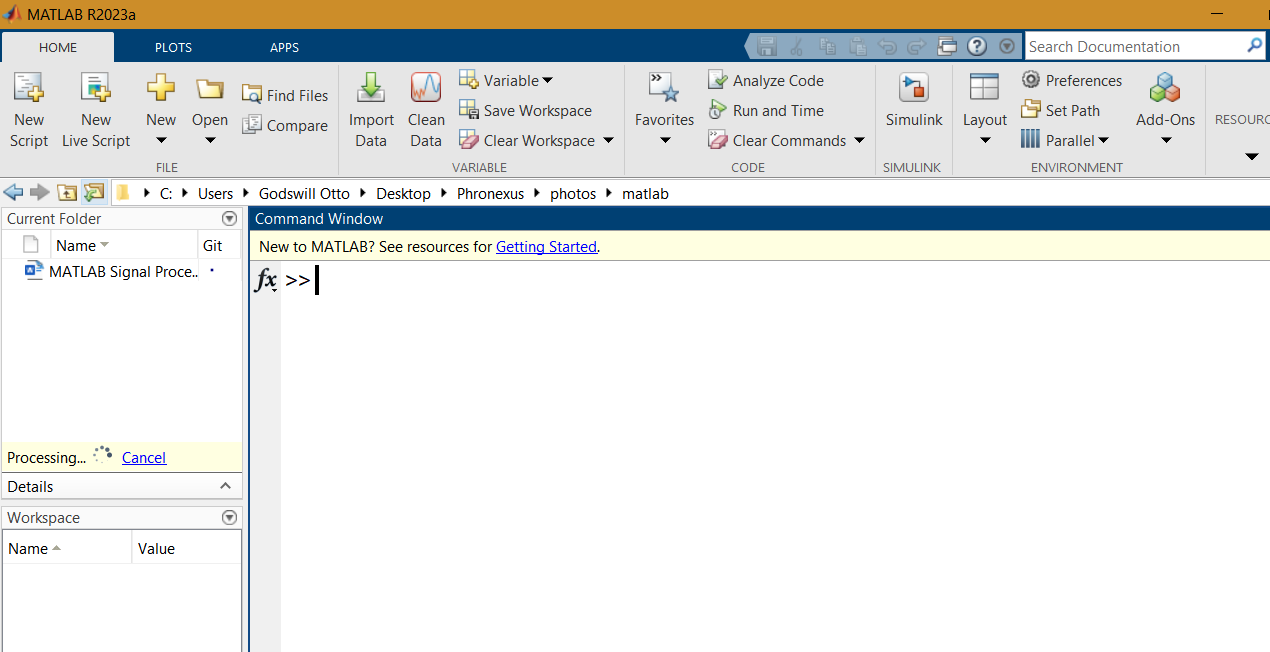
MATLAB (Matrix laboratory) is the trusted ally of engineers and scientists. It is a versatile tool that seamlessly integrates theory and practice, empowering you to bridge concepts with real-world applications. It has an intuitive interface and unmatched prowess in handling matrices and vectors. Its extensive array of functions for signal processing makes it a standout choice. From data generation to noise manipulation, MATLAB ensures this journey into signal processing is smooth, efficient, and, most importantly, user-friendly!

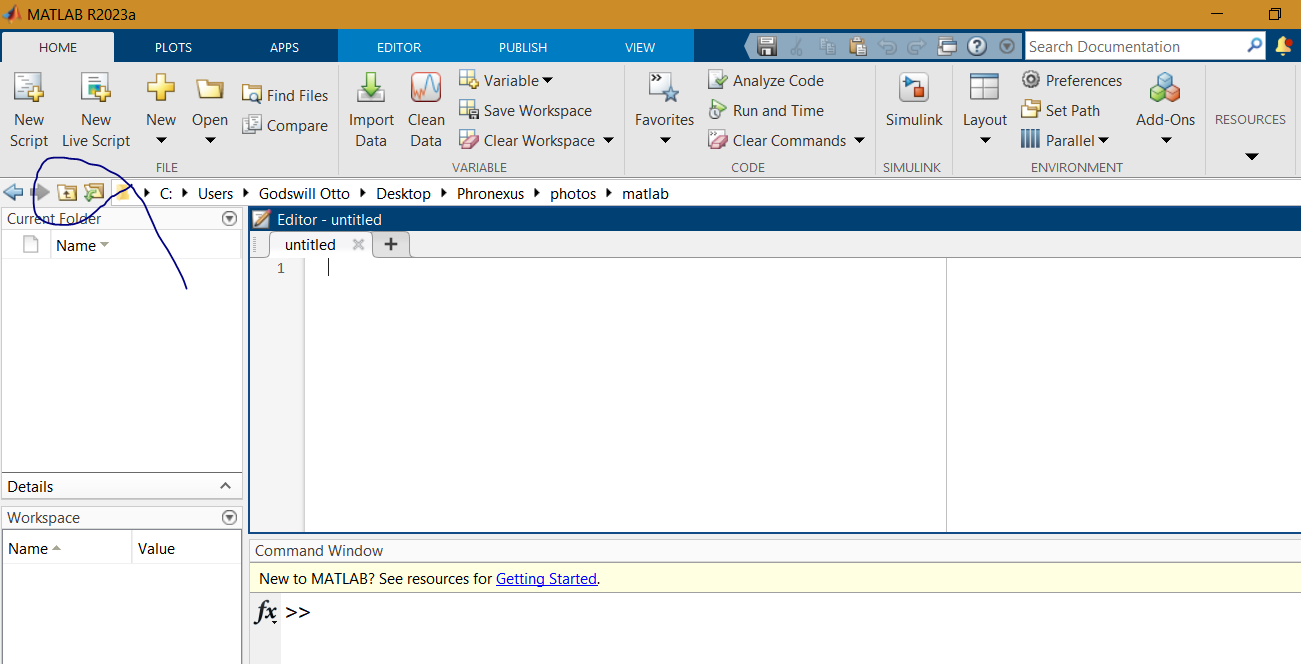
To install MATLAB, you can follow this video or read this article.

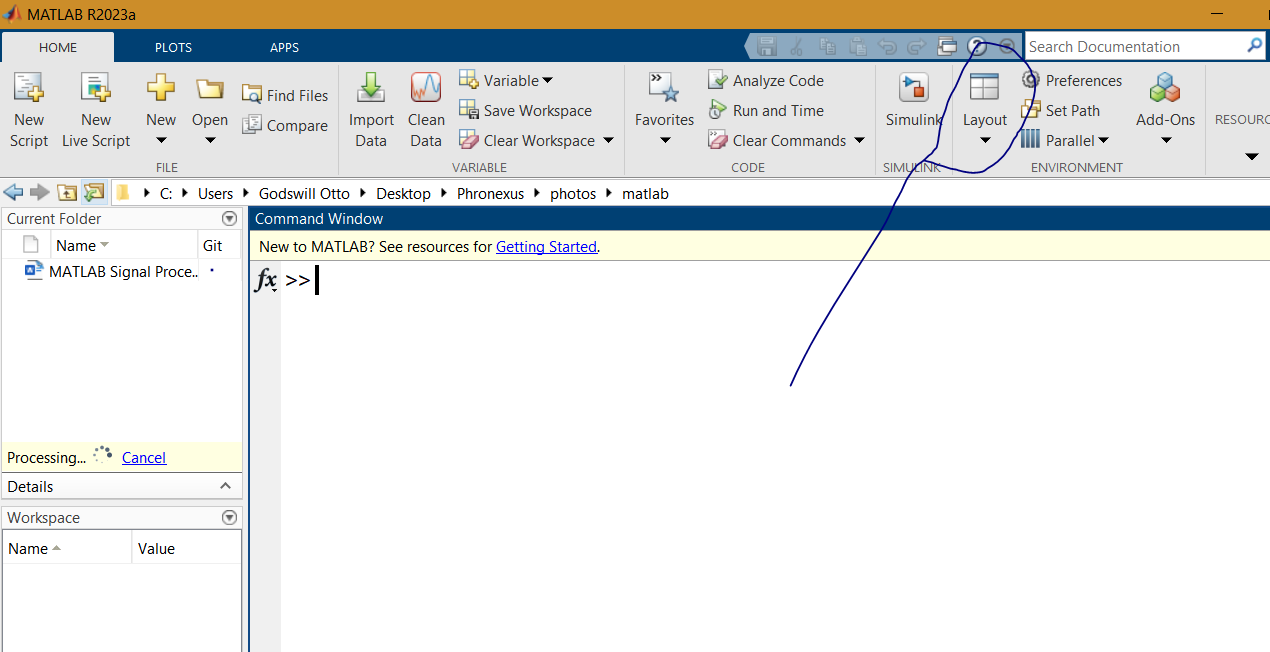
However, the good thing is if you are a student in an engineering school and you would require the course in your study, your school probably already provided a MATLAB access via your school’s portal. Also check out for that!

**Meet the Playground - MATLAB Interface**

The matlab environment looks like this.



Use this button below to navigate to the directory you want to work on. I suggest you create that directory beforehand.

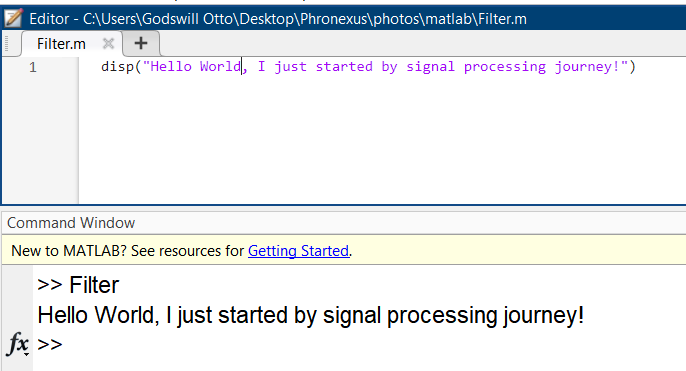
If in case your layout is not exactly like mine or you prefer it some other way use the layout button to choose the one you prefer. You want to have you command window and your work space showing as well. Optionally it is occasionally good to have your directory showing so it is easy to see files you want to work with. Then select default or whichever you like then tick the current folder and workspace options

You can quickly experiment with the command window but when writing codes, you want to use the script. So, click new script lets dive into saying hello world.

**Saying Hello with Code**

Now, let the fun Begin. Dive into the coding sea with our "Hello, World!" program. Fear not if you're not a coding maestro yet – we're keeping it simple

The code for hello world is below

 Disp(“Hello World, I just started by signal processing journey!”)

Hopefully it worked for you!

Another example

t = [0: 1/100 : 1];

disp(t)

Here, we have just created an array from 0 to 1 with steps of 1/100 (0.01) and saved to a variable t.

In signals you can think of it as time variable sampled at 100Hz and each time stamp (n) is increased by 0.01. Note the “;” is there to stop the varaibles from spilling or litering the command window, use it frequently unless you want to see the output of that line of code.

Easy peasy ya?

**Data Generation Magic**

Now that you've exchanged greetings, let's conjure some magic – data magic, to be exact. We're about to unravel the basics of generating clean data. Picture conducting a symphony, but this time, your orchestra is comprised of functions and numbers. Our chosen instrument? The unassuming sine wave. Simple, elegant, and the backbone of signal processing.

Let’s generate a simple wave with a given frequency of say 900Hz. Now, for the sake of filtering we would join this wave with another of higher frequency say 1200 Hz. To avoid distortion, theoretically have your sample frequency >= 2 times the max frequency so put sample frequency at 3000 Hz.

Let’s use the sine function with an amplitude of 4.

% sample frequency 3000 Hz, this means a max freq of 1500Hz can be used

% without aliasing distortion

Fs = 3000;

% Sample time

T = 1/Fs;

f1 = 900;

% generate data

t = [0:T:1-T];

sig = 4 \* sin(2\*pi \*f1 \*t);

explanation of code above

A sample frequency is defined as 3000 Hz, then the sample time is calculated. The data on the sample time is calculated and the function is inputed.

**Adding a Dash of Noise**

Now, let’s add some noise to the data.

Let’s create another wave of different frequency (1200 Hz) which we would filter out in this tutorial.

f2 = 1200;

noise = 12 \*sin(2\*pi\* f2 \*t) + 0.2 \* randn(size(t));

**Let's Mix Things Up - Combining Data**

Now, let's play with our data. Mixing clean and noisy data gives us a taste of real-world complexity. Here's how you can blend these signals

dirtyData = cleanData + noise;

**Visualize the Symphony - Data Plotting**

So, you want to dive into the world of plotting in MATLAB? Easy peasy! First off, fire up the plot using the figure command and unleash the plot function for some visual magic. Take a look at this snippet:

plot(t, cleanData);

xlabel('Time ');

ylabel('Amplitude');

title('Time Domain plot of clean signal');

figure;

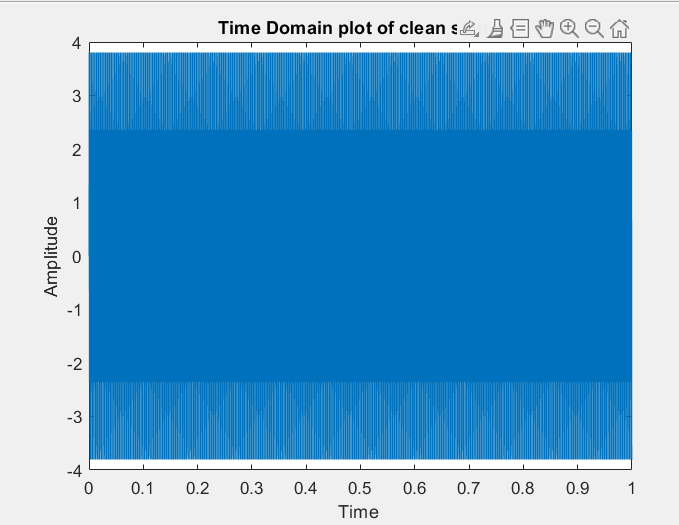
plot(t, dirtyData);

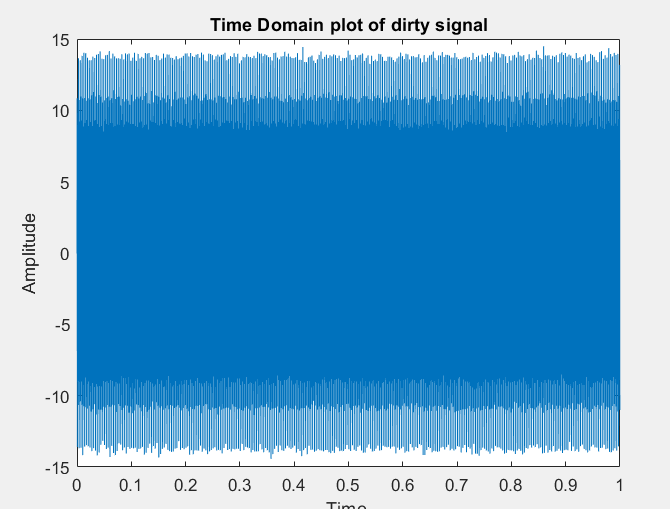
xlabel('Time ');

ylabel('Amplitude');

title('Time Domain plot of dirty signal');

Boom! You've got yourself some sleek time domain plots.





But wait, there's more!

Ever thought about seeing your signal in the frequency domain? MATLAB's got your back with the Fourier transform.

To convert the signal to the frequency domain use the fourier transform. Matlab provides as alternative to DFT in a matlab function fft (fast fourrier transform). Then use the absolute function to get the absolute value of the magnitude

Check out this snippet::

dirtyDataFd = abs(fft(dirtyData));

cleanDataFd = abs(fft(cleanData));

figure;

plot(dirtyDataFd);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Magnitude Spectrum of dirty Signal');

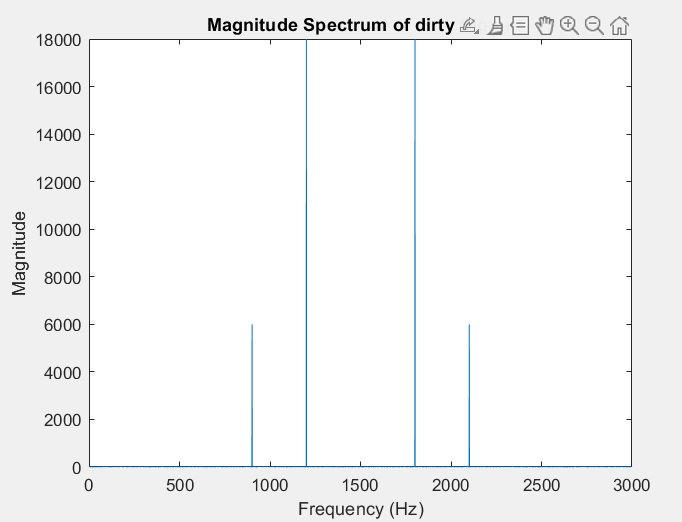
figure;

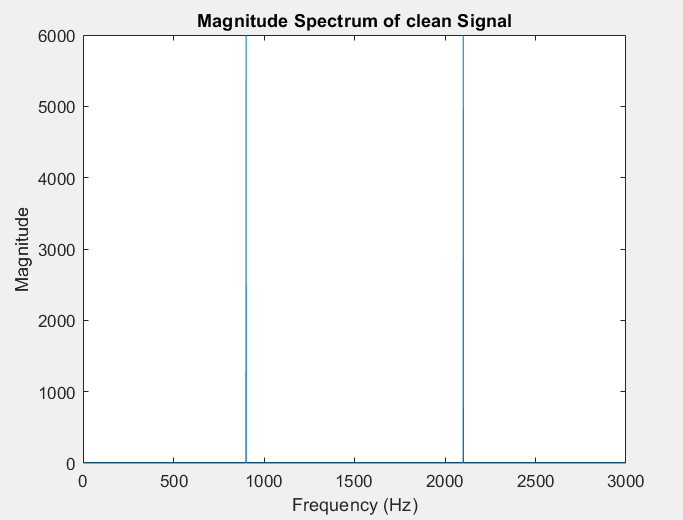
plot(cleanDataFd);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Magnitude Spectrum of clean Signal');

Here is the output



Voilà! You just unleashed the power of magnitude frequency plots. See the difference? The clean signal has one frequency chillin', while the dirty one's got a duo. Now, our mission? Filter out that interfering frequency. Ready for the next adventure?

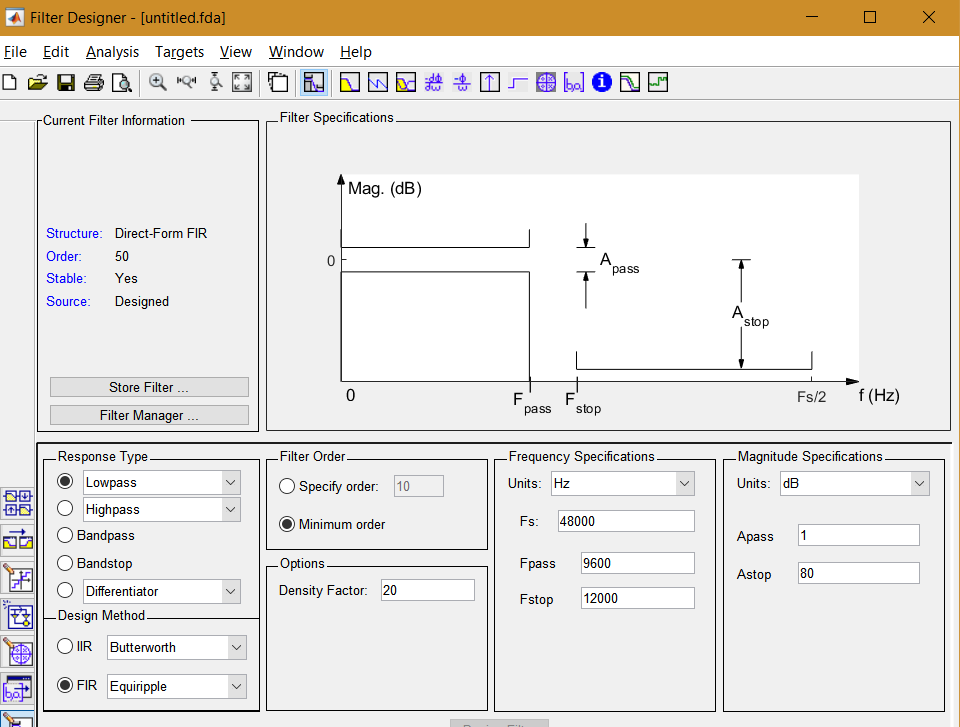
**Enter the Filter Designer's Lair**

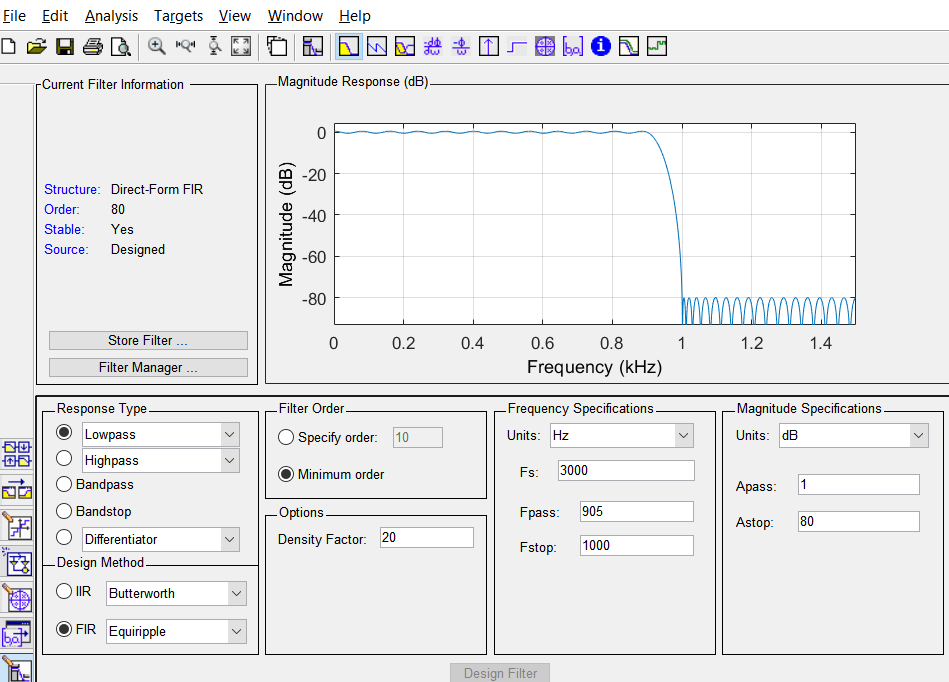
To conquer the higher frequency you need a .. (you probably already guessed it!) ya a Low Pass Filter. Writing that I feel like sokka from the airbending movie. 😂

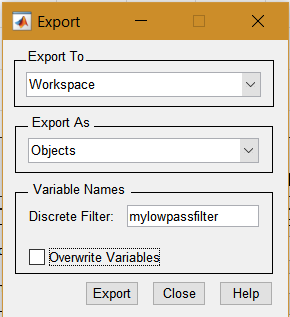
Matlab provides a tool called filter designer. Lets explore it! Summon the filter designer by typing "filterDesigner" in your command window. A new window unfolds, brimming with possibilities—close the tip of the day, we've got work to do.

Now, they are a lot of useful features but let's keep it simple. We're crafting a low pass filter with a band stop above 900 and below 1200. Here's the playbook:

* Select "Lowpass" in Response type.
* Set Fpass at 900 (or a smidge above) and the bandstop at 1000 (below 1200).
* Set the sample frequency to 3000 Hz.
* Leave everything else at default.
* Click "Design filter" at the bottom.





Once your masterpiece is ready, click "Files," then "Export," and choose "Export to Workspace" and "Export as Objects." Name your filter (let's call it "mylowpassfilter") and hit export.

After exporting lets get back to the Matlab interface!

**Filtering Out the Noise**

Now, let's tackle the unruly "dirtyData" with our freshly crafted filter and matlab filter function. Behold the magic in code:

Code snippet

% filtering

filterData= filter(mylowpassfilter, dirtyData);

filterDataFd = abs(fft(filterData));

**Admire the clean data and your new skillset**

Visualize and admire your result!

figure;

% Plot the magnitude spectrum of the filtered signal

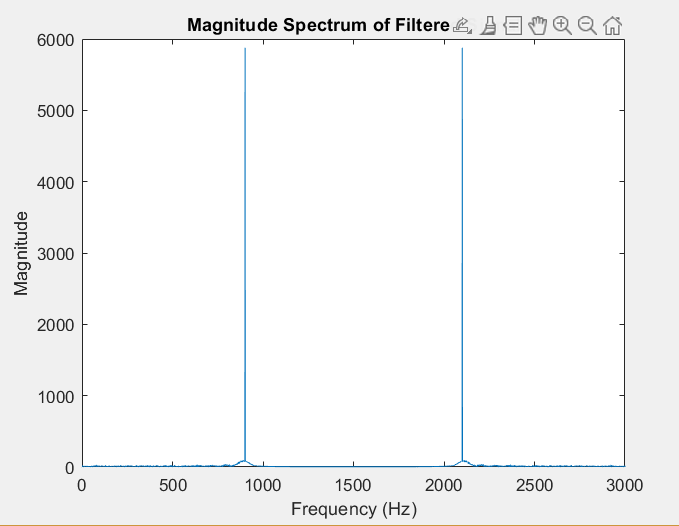
plot(filterDataFd);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Magnitude Spectrum of Filtered Signal');

Output



Hooray! Revel in the glory of your new skill. You've just unleashed the power of filtering in MATLAB. Cheers to cleaner signals and a brighter signal processing journey

**The grand finale and beyond**

As our MATLAB journey comes to a close, let's gather around for a quick recap. You've conquered the basics, danced with codes, and unleashed the power of plots. MATLAB, your trusty companion, has proven itself a signal processing wizard.

But hey, this isn't the end; it's just the beginning. Imagine applying these newfound skills to real-world scenarios. Picture this: filtering out the unwanted noise from your favorite song, creating a symphony of pure audio bliss. Filters aren't just lines of code; they're the maestros orchestrating harmony in our tech-driven world.

**See You Next Time:**

As we bid adieu for now, remember, this is just a pause, not the end. What's next on our agenda? More textbook assignments to polish those skills, preparing you for the grand stage of real-life applications. Soon, you'll be weaving MATLAB's magic in music and silencing the noise in the data-driven world of robotics. Stay curious, stay excited—see you in the next tutorial!

If you have any questions or have any specific tutorial you want me to make, reach out to me on my facebook page. [Phronexus Freelance services](https://web.facebook.com/profile.php?id=61555879295314).

Here is the full code:

% sample frequency 3000 Hz, this means a max freq of 1500Hz can be used

% without aliasing distortion

Fs = 3000;

% Sample time

T = 1/Fs;

f1 = 900;

% generate data

t = 0:T:1-T;

frequencies = 0:1:Fs-1;

cleanData = 4 \* sin(2\*pi \*f1 \*t);

f2 = 1200;

noise = 12 \*sin(2\*pi\* f2 \*t) + 0.2 \* randn(size(t));

dirtyData = cleanData + noise;

plot(t, cleanData);

xlabel('Time ');

ylabel('Amplitude');

title('Time Domain plot of clean signal');

figure;

plot(t, dirtyData);

xlabel('Time ');

ylabel('Amplitude');

title('Time Domain plot of dirty signal');

dirtyDataFd = abs(fft(dirtyData));

cleanDataFd = abs(fft(cleanData));

figure;

plot(dirtyDataFd);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Magnitude Spectrum of dirty Signal');

figure;

plot(cleanDataFd);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Magnitude Spectrum of clean Signal');

% filtering

filterData= filter(mylowpassfilter, dirtyData);

filterDataFd = abs(fft(filterData));

figure;

% Plot the magnitude spectrum of the filtered signal

plot(filterDataFd);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Magnitude Spectrum of Filtered Signal');