X-WaveSLauncher Help File

Guillermo De Arana Schoebel
Research Assistant

March 25, 2025

Contents

1 Introduction

This user manual aims to first and foremost explain how to set up an experiment with the application developed for the X-WaveS pump and probe experiment, additionally an effort was made to transfer as much of my knowledge acquired when setting up the hardware side of the experiment.

2 Theory of detection

A full and comprehensive explanation of the Transient Grating Four-Wave Mixing setup used by the X-WaveS team is beyond the scope of this document, a certain familiarity with the pump and probe scheme is necessary to proceed. However, we will outline the basic theory behind lock-in amplifier (LIA) signal acquisition and explain how it relates to this particular setup, as the user must be familiar with this technique in order to properly configure the application.

In essence, the experiment is conducted similarly to many other pump-and-probe setups. A pulsed laser source emits a 5 kHz pulse train, which is directed onto a spinning chopper blade. This chopper is rotated such that only one out of every two pulses passes through (i.e., it operates at the $\frac{1}{2}$ subharmonic of the laser frequency). A photodiode then records the sample's response. The signals from half of the probe pulses correspond to a ”pumped” sample, while the other half correspond to an ”unpumped” sample. The output of the LIA is effectively proportional to the difference in intensity between the pumped and unpumped responses, providing a normalized measurement.

To understand how this is achieved—and to make the most of both our measurements and experimental time—we must first understand the internal workings of a lock-in amplifier.

2.1 LIA detection

2.1.1 What is a LIA?

A lock-in amplifier (LIA) is, in short, a device that extracts the signal power at a specific reference frequency. Therefore, whenever a LIA is employed, another actuator (in our case, a spinning chopper blade) must modulate the experiment's response so that it oscillates at the chosen reference frequency. If the signal has a non-zero amplitude at that frequency, the LIA can isolate and measure its power, effectively ignoring all other signal contributions that correspond to noise. This makes the LIA an ideal instrument for extracting weak signals in noisy environments.

In the following sections, I have included two complementary explanations of LIA operation. In my opinion, both are necessary for a full understanding. One explanation approaches the LIA from a signal processing perspective, while the other presents a more formal description of its internal operation. The former is helpful for intuition and practical understanding; the latter is more detailed and theoretical.

2.1.2 Lock-in Amplifiers, the intuitive idea

A Lock-in amplifier can be abstracted to just a couple of signal blocks (see Fig. 1)

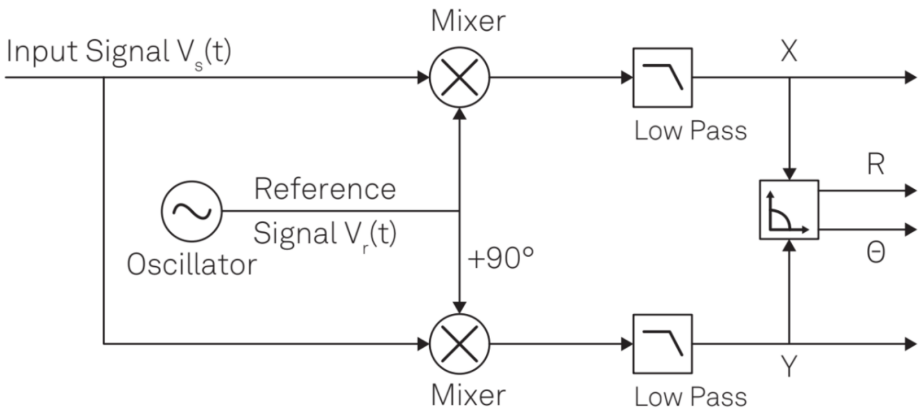


Figure 1: Lockin amplifier abstracted to basic signal blocks

Above, we see the internal block diagram for the signal processing algorithm used inside the lock-in amplifier (LIA). Before diving into the algorithm itself, it is worth briefly discussing the concept of *input range*, which is one of the few parameters the user can control in this application.

The LIA is an instrument that receives analog voltages and digitizes them before running its signal recovery algorithm. Prior to digitization, the input signal is passed through an internal amplifier whose gain depends on the configured input range. For example, if the user selects a 10 mV input range, the LIA will adjust its internal gain so that signals exceeding 10 mV will approach saturation. In simple terms, the user is determining how much the signal is amplified before digitization.[1]

Setting the input range too low may cause saturation, while setting it too high may result in a loss of precision due to poor utilization of the analog-to-digital converter's resolution. Fortunately, most modern LIAs—including the one used in this application—support an *auto-range* function that automatically selects the optimal range for the incoming signal.

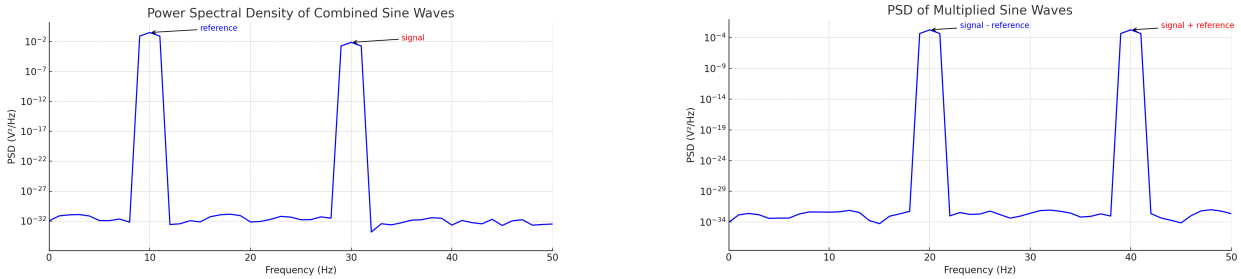
Now that the signal has been digitized, we can begin to describe how the LIA algorithm works. Two signals are required to extract the power of a low-level signal: one is the signal from the photodiode, referred to as the *input signal* $V_s(t)$, and the other is a *reference signal* $V_r(t)$, which tells the LIA at which frequency to extract the signal power. As a first-order approximation (a "white lie"), we may consider the reference signal to be the 5 kHz trigger frequency of the laser source.

In the diagram above, we see that the circuit is duplicated with a 90° phase shift between the two branches. For now, we will evaluate only the top half (see section 2.1.3) . There, we observe that the input signal is "mixed" with the reference signal. This mixing process will be described in more detail in the following section, but for now we assume that the mixer $\otimes$ takes two signals, $V_s(t)$ and $V_r(t)$, and outputs a third signal: $V_s(t) \times V_r(t)$.

Below, we show both signals represented in the frequency domain before and after mixing. As a simplified example, let us assume both signals are pure sine waves at 10 Hz and 30 Hz, respectively. Each will appear as a Dirac delta function in the frequency spectrum:

The mixer takes the two delta peaks at f_{sig} and f_{ref} and produces two new frequency components: one at $f_{sig} - f_{ref}$, and another at $f_{sig} + f_{ref}$. In our example, we have assumed that the input signal $V_s(t)$ is purely sinusoidal. In reality, the signal from the experiment is a train of pulses, rich in harmonics and noise, and its frequency spectrum is more complex.

[1] The user does not need to manually correct for this gain—the LIA internally rescales the output to reflect the actual input voltage. That is, 1 μV_{rms} at the output corresponds to 1 μV_{rms} at the input.



(a) Before mixer

(b) After mixer

Figure 2: Power spectral densities before and after the mixer.

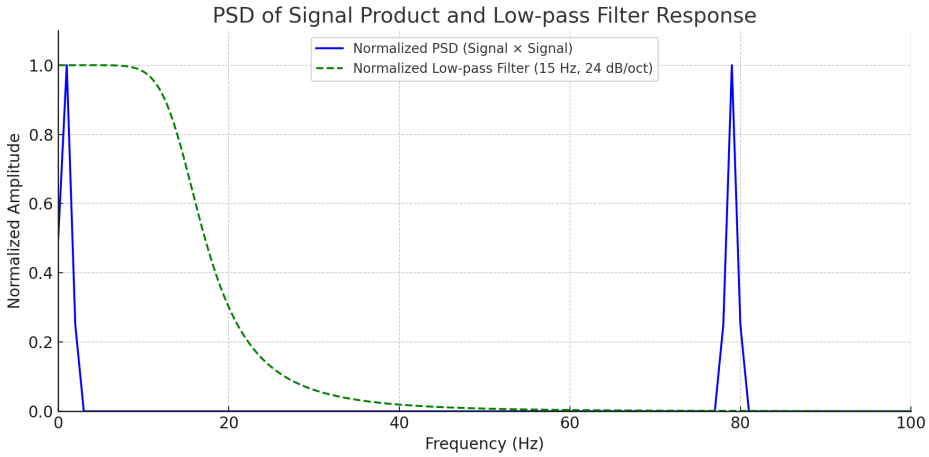


Figure 3: PSD after Los-Pass filter

However, from Fourier theory we know that any periodic signal can be decomposed into an infinite sum of sine waves. Therefore, when the mixer receives a complex signal with many harmonics and noise components, it behaves similarly: the spectrum is spread out, and mixing creates shifted copies of each frequency component, just as illustrated in Fig. 2.

There is one particular frequency component that will land at a convenient spot in the spectrum: if the first harmonic[2] of the signal is oscillating precisely at $f_{sig} = f_{ref}$, the mixer will generate components at $f_{ref} - f_{ref} = 0\,\text{Hz}$ and $f_{ref} + f_{ref} = 2f_{ref}$. All other frequencies—including noise and harmonics—will be shifted to other regions of the spectrum. Since the signal power of interest now lies at 0 Hz (i.e., DC), it can be extracted with a low-pass filter: a type of circuit or digital signal processing (DSP) algorithm that attenuates frequencies above a certain cutoff frequency.

Fig. 3 shows how a low-pass filter, placed after the mixer, attenuates frequency components far from 0 Hz. This filter can be characterized by two parameters: the *cutoff frequency* and the *filter roll-off*. The roll-off describes the slope, in dB/Oct, at which the filter attenuates increasingly higher frequencies. The cutoff frequency is defined as the frequency at which a signal's power drops to half its value when passing through the filter (for the filter shown in Fig. 3, this occurs at approximately 18 Hz). An equivalent and more commonly used metric is the filter's *time constant*, given by:

[2]Note that we do not typically study the first harmonic directly. For further details, see Section 2.1.4.

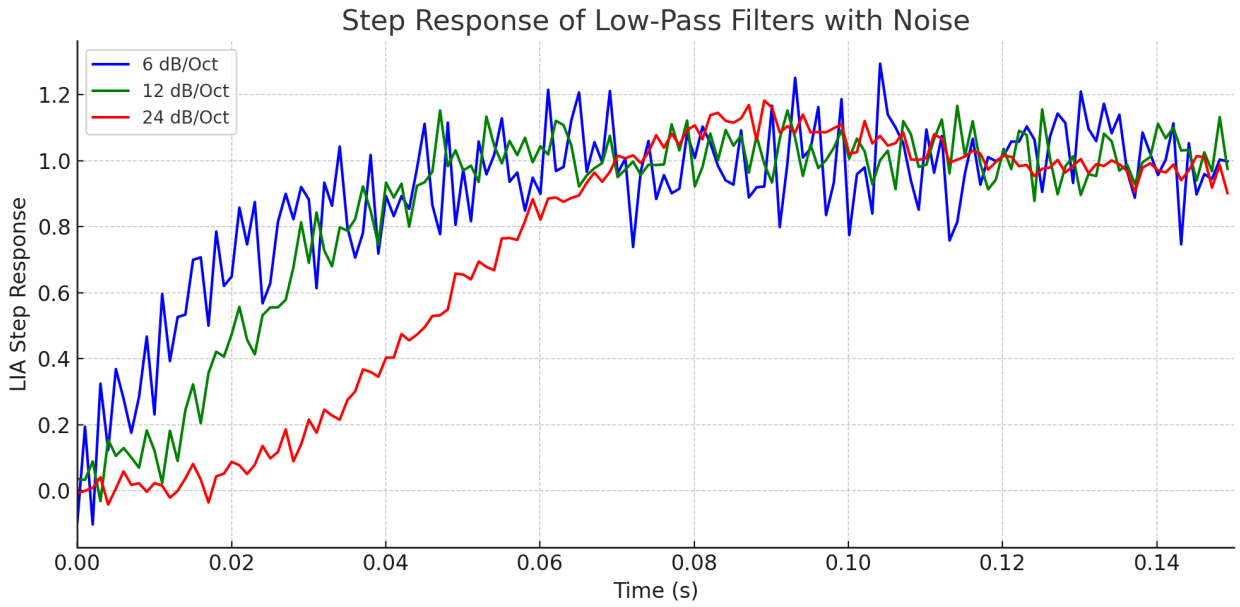


Figure 4: LIA's output for step input signal + noise under different low pass filter roll-offs.

$$\tau = \frac{1}{2\pi f_{\text{c.o.}}}$$

Both definitions are interchangeable, though historically, the time constant has been more widely used for reasons we will explore later.

Increasing either the filter roll-off or the time constant reduces the amount of noise in the LIA output, resulting in a measurement that more accurately reflects the signal power at f_{ref} in the signal V_{sig}. At this point, the user may wonder: *Why not always run the experiment with both parameters set to their maximum values? Why provide control over these at all?*

The reason is that a filter becomes increasingly *sluggish* as the time constant or roll-off increases. To illustrate this, consider running four versions of the same experiment, each with a different filter roll-off. Suppose we move the delay stage into time-zero and record the LIA output in each case. The results would resemble those shown in Fig. 4. There, the immediate rise in signal power is most quickly reflected by the filter with the lowest roll-off, 6 dB/Oct. This is followed by the 12 dB/Oct configuration, and finally the 24 dB/Oct setup.

On the other hand, the noisiest signal is observed with the 6 dB/Oct configuration, while the cleanest response comes from the 24 dB/Oct filter. In essence, the lock-in technique—like most signal recovery method—presents a trade-off between precision and speed. The researcher must select a combination of filter roll-off and time constant that balances signal-to-noise ratio (SNR) and total measurement time.The application will make sure to wait until the signal has reached 99.9% of it's step response.

With this, we conclude our intuitive overview aimed at helping the user understand how to configure a LIA. Perhaps the most straightforward way to conceptualize the LIA is as a device that takes an input signal, shifts the component at f_{ref} down to 0 Hz, and then measures its signal power. The user controls how this power is extracted—balancing precision and response time—by adjusting the parameters of a low-pass filter.

This explanation is sufficient for most general use cases and should provide a solid foundation for effectively configuring the application. However, for users seeking a deeper understanding of the internal workings of the LIA, the following section offers a more formal and rigorous treatment of its operating principles.

2.1.3 Lock-in Amplifiers, the math

Lock-in amplifiers trace their origins to the early days of computation, in the first half of the 20th century. At that time, digital abstractions did not exist—computation and data analysis were carried out using analog electronics. Circuits were physically interconnected to perform operations on continuous signals. In many cases, researchers needed to compute the Fourier Transform (FT) of a signal at a specific frequency, and this gave rise to the lock-in amplifier: a type of analog computer. While such analog computers are now obsolete, the algorithmic structure they implemented survives in modern digital lock-in amplifiers.

Let us begin with the mathematical expression for the Fourier Transform:

$$\hat{g}(f) = \int_{-\infty}^{\infty} g(t)\, e^{-j2\pi ft}\, dt$$

Applying Euler's formula to expand the complex exponential, we obtain:

$$\hat{g}(f) = \int_{-\infty}^{\infty} g(t)\, [\cos(2\pi ft) - j \sin(2\pi ft)]\, dt$$

Each term in this expression can be interpreted as a physical operation carried out by analog electronics. The signal g(t) can be multiplied by a cosine at the reference frequency f_{ref} using a mixer circuit. To obtain the sine term, the reference waveform can be phase-shifted by 90° and sent to a second mixer. This results in two separate signals:

$$X = \int_{-\infty}^{\infty} g(t) \cos(2\pi f_{ref} t)\, dt \quad \textit{and} \quad Y = \int_{-\infty}^{\infty} jg(t) \sin(2\pi f_{ref} t)\, dt$$

In practice, these integrals are realized through low-pass filters, which suppress all frequency components except for the one centered at DC. Since the product of g(t) and the reference waveforms shifts the desired frequency content down to zero (as explained in earlier sections), low-pass filtering effectively computes the time-averaged value—or integral—of each term.

Finally, the in-phase (X) and quadrature (Y) components are combined to recover the amplitude and phase of the original signal at the reference frequency:

$$R = \sqrt{X^2 + Y^2}, \qquad \theta = \tan^{-1}\left(\frac{Y}{X}\right)$$

This operation translates the result from Cartesian to polar coordinates, completing the reconstruction of a single frequency component of the signal. In essence, the lock-in amplifier implements a localized, real-time Fourier Transform for one frequency, using a structure inherited from its analog ancestors.

This explanation finally paints the whole picture on the diagram at Fig. 1 however there is one last white lie we need to tumble from the previous sections, that concerns the reference frequency fed to the LIA.

2.1.4 How the LIA normalizes the pump and probe signal

As explained in previous sections, the lock-in amplifier (LIA) is a device that extracts the signal power (or, more formally, the amplitude of the Fourier Transform) of an input signal V_{sig} at a specific reference frequency f_{ref}. We also reasoned earlier that the LIA does not strictly require a sinusoidal input — any periodic signal with a consistent harmonic component at f_{ref} can be demodulated. At this point, the reader may reasonably ask: if our laser already emits a periodic train of pulses at 5 kHz, why do we need to chop the beam? Why not simply feed the pulse train directly into the LIA and extract the signal power at that frequency?

While this approach could work in principle, it presents two critical problems that the use of a chopper elegantly solves.

First, any background process or optical artifact that also produces signal content at 5 kHz—such as scattered laser light or fluorescence—will contaminate the measurement. Second, in pump-and-probe experiments, strong excitation may lead to the phenomenon known as *printing*, where the transient grating is inadvertently imprinted on the sample, causing persistent changes that distort subsequent measurements.

By modulating the pump beam with a mechanical chopper operating at half the laser repetition rate (i.e., at 2.5 kHz), we effectively turn the pump on and off for alternating laser pulses. This modulation causes the sample response to alternate as well: one pulse interacts with a "pumped" sample, the next with an "unpumped" one. The lock-in amplifier, configured to detect at this subharmonic frequency, will then extract the signal power corresponding to the *difference* between the pumped and unpumped responses — effectively normalizing the measurement and canceling any static or background contributions.

Figure 5 illustrates this concept through a simulated example. Consider a pulse train at 5 kHz, where every pulse initially has identical amplitude (blue trace). Its frequency spectrum consists of harmonics at 5 kHz, 10 kHz, 15 kHz, ... Now suppose that every second pulse is slightly attenuated (shown with a green hue). This subtle difference introduces a new modulation at 2.5 kHz. One can intuitively imagine a 2.5 kHz sine wave superimposed onto the pulse amplitudes—its peaks aligning with the higher (pumped) pulses, and its troughs with the attenuated (unpumped) ones. This alternating behavior effectively encodes a new oscillatory component into the signal, which manifests in the frequency domain as a spectral peak at 2.5 kHz.

The amplitude of this subharmonic component is directly proportional to the difference in height between consecutive pulses. Measuring at this frequency with a lock-in amplifier serves a dual purpose: it normalizes the signal by rejecting static or parasitic contributions—such as scattered light—that may also oscillate at 5 kHz, and it suppresses artifacts due to sample printing. In such cases, the printed response may persist even when the pump is nominally "off," reducing the contrast between alternating pulses and thus lowering the measured power at 2.5 kHz.

Conclusion This is why the pump beam is modulated using a chopper, and why the lock-in amplifier is configured to detect at the chopper frequency—half the laser repetition rate. By doing so, we isolate the dynamic response induced by the pump, reject static and parasitic signals, and ensure the measurement reflects the true, normalized difference between the pumped and unpumped states of the sample.

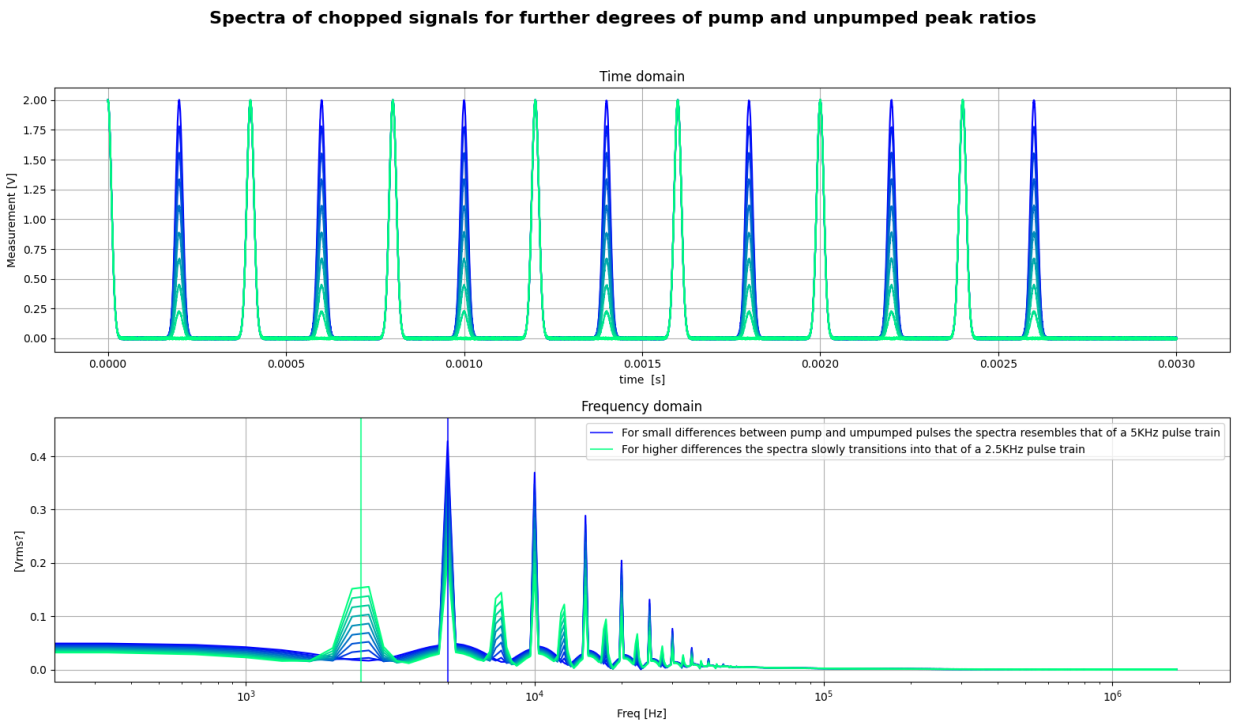


Figure 5: Simulated pulse train and its corresponding frequency spectrum. Subharmonic modulation appears when every second pulse is attenuated.

3 How to configure an experiment

This section explains how to use the application to set-up an experiment as well as some quirks that the user might find useful

3.1 Downloading and Installing the application

Install this application from scratch on a Windows computer by following these steps:

- Download Git in the computer from this link and follow the basic installation instructions, we won't do nothing fancy with this, it will only allow us to download the application itself from the online repository
- Open/Create a folder where you would like the applications folder to be created, right click in the folder and open CMD or PowerShell, in either type

```
1    git clone https://github.com/Guilldeas/
        Pump_Probe_Measurement_Automatization.git
```

- Before running the files you'll need to install python 3.10.5 from this link follow installation instructions and make sure to check the option to add to PATH

  Once you are done installing this specific version of python verify your installation by opening the command line the same way as before on the X-WaveSLauncher folder and checking that you get the same result

```
PS C:\Users\...\X-WaveSLauncher\
    Pump_Probe_Measurement_Automatization> python --version
Python 3.10.5
```

- Now that your python installation was successful it's time to create a virtual environment, this will allow you to install all of the python packages needed for this application without conflicting with any other python applications in your machine. To do this ope your `Pump_Probe_Measurement_Automatization` folder and in it right click to open a Powershell/CMD terminal, in it run the following line

```
py -3.10 -m venv Measurement_Automatization_venv
```

Activate your virtual environment

```
Measurement_Automatization_venv\Scripts\Ativate
```

- if this step triggers the following error:

No se puede cargar el archivo porque la ejecución de scripts está deshabilitada en este sistema.

Then run

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Unrestricted
```

And retry

```
Measurement_Automatization_venv\Scripts\Ativate
```

- Now that your environment is activated (you should see the name of the environment in parenthesis preceding your user name in the terminal) install the packages needed to run the application by running

```
pip install -r requirements.txt
```

This will take a while, please don't close the terminal/computer and do not press `Ctrl+C` while the installation is taking place.

Verify the installation was succesful by running:

```
python X-WaveSLauncher.py
```

And checking that the application indeed runs as normal.

3.2 Running the application

To run the application however you don't need to open CMD/Powershell on the correct path, activate your environment and run the python script every time, there is an easier solution for that:

- Go into `Pump_Probe_Measurement_Automatization` (your installation folder), locate the file `run_X-WaveSLauncher.bat` and create a shortcut for it, move that shortcut wherever you see fit and run the program by double clicking that shortcut. **Warning:** Moving the original files out of the installation folder may break the program, this is not true however for anything inside the Output folder.
- The user will need to access the Output files as well, moving the original output folder outside of the installation is not advisable either. Create a shortcut for it and move it to wherever you see fit. If the Output folder is not present simply run a dummy experiment and the Output folder will be created automatically

3.3 Initialization Screen

Upon booting the application the user is welcomed with a screen prompting them to `Initialize devices`, clicking this button initializes a series of steps where the computer connects to the delay stage and LIA and configures them to an ”experiment ready” state. This intialization is time consuming due to the delay stage homing process however it only needs to be performed once as long as the application is open, closing and relaunching the application will loose connection to the devices and thus will prompt the user to initialize again. Since the initialization process is time consuming this application was developed such that the user **does not** need to leave it idle during initialization and can immediately continue setting up the experiment parameters, there is only one functionality locked until initialization is established which is launching an experiment.

The steps taken by the application during initialization are logged on a secondary window, this serves the purpose of informing the user of the state that the delay stage and LIA are configured to.

The values present on the initialization screen are only for information purposes, however the user can still work around this (See section 3.5) .

3.4 Experiment Screen

At any moment throughout the process the user may change screens from the Initialization screen to the Experiment screen through the `File` menu on the top left. In the Experiment screen the user will be able to configure how the experiment takes place by setting different parameters. These values may be scrolled faster by switching between them with Tab.

What follows is an application centered explanation, follow the links for further information.

Input cells

- `experiment file name` Sets the name for the folder where data will be stored. The application prevents the user from using the same file name twice to prevent any sort of overwriting, if you really wish to use the same file name you'll have to manually go into the output folder and delete the homonymous folder **Warning:** The application does not check that the value input by the user here is safe, please abstain from using special characters and follow Windows rules for naming folders, failing to do so will trigger an error and prevent the application from saving the experiment data: Naming Files, Paths, and Namespaces
- `filter roll-off [dB/oct]` Sets the filter slope of the Low-Pass filter at the LIA. (For more information on how these values influence the measurement see section 2).
- `time constant [s]` Sets the LIA's filter frequency/time constant.
- `Error measurement type` The LIA is able to compute an approximation to it's output noise by computing the standard deviation of it's buffer (see SR860's manual for a detailed explanation), in this application the error measurement is computed as the combined standard deviation of the channel X and Y. However this is a time consuming process [3] so the user can choose whether to employ it or not through the following options: `Never` No error measurements will be performed during the experiment and thus no signal error column will be written into the output excel, `Once at the start` noise measurements will only be computed at the start and the output column for noise will be padded with this initial value, `At every point` will measure noise after every data point and will write it on the output file.

[3] For low time constants error measuring at every point may account for 20% of the experiment execution time! Play around with the different options in the Experiment screen, test them before execution with the estimation tool and observe the final tally up of execution times at the end of the experiment log to optimize the total experiment execution time.

- `Autoranging type` The LIA has an internal amplifier that can amplify the signal before digitizing it and performing it's internal algorithm, thus the best Signal to Noise Ratio will be obtained when choosing an amplification [4] that maximizes the signal level without saturating the amplifier, the LIA is able to approximate this by itself through autoranging however this process is also time consuming and thus the user may again choose whether to employ it or not through the following options: `Never` The LIA will not attempt to optimize it's gain and will run the experiment with whatever gain was set on it's panel (this yields faster experiments but delegates the responsibility of avoiding saturation to the user), `Once at time zero` The gain will only be adjusted once to it's upper limit, at the beginning of the experiment the stage will be sent to time zero, the application assumes this is the delay where the signal will be the strongest and thus more prone to saturating the LIA's amplifier so the experiment, the rest of the experiment will be performed with this gain thus yielding a faster experiment at the expense of a loss in SNR for lower signals `At every point` will measure noise after every data point and will write it on the output file.
- `rel time zero [ps]` Specifies the delay time at which the pump and probe pulses meet (for a proper explanation on the absolute and relative naming conventions see Fig. 6)
- `Number of scans` Sets the number of scans that the application will perform, scans are a way to repeat the whole experiment and average it.
- `abs time start [ps]` Sets the start point for a particular leg or range, this start point does not need to be specified before or after a previous leg, the application does not check for overlapping, however it does enforce that the user does not enter an invalid delay outside of the 4002ps range that the delay stage can do, in essence it prevents the user from attempting to send the stage outside of it's 600mm range. Finally users may introduce decimal values, negative values and use scientific notation like $1E-2$ or 2e3. These rules also apply for `abs time end [ps]` and `step [ps]`. (For a proper explanation on scans and legs naming convention see Fig. 7)
- `abs time end [ps]` Sets the end point for that particular leg or range.
- `step [ps]` Sets the length of the increment at every delay point in the leg, this step size is limited to 4002ps. Choosing a step size larger than the total leg length will result in a leg that will scan only 2 points: at the start and at the end. Choosing a step that is not a divisor of the leg length will cause the final step to be shortened such that the leg ends wherever the user specified.

Buttons

Clicking any of the buttons causes the application to check the user input cells for incorrect/unsafe values and fails to complete the buttons action if any of the parameters does not pass the verification. All of the buttons can be used during initialization, only `Launch experiment` will fail to execute.

- `Save parameters` Will save the parameters at the experiment screen for the the next time the user launches the application.
- `Edit number of trip legs` Changes the amount of legs that the scan will execute.
- `Estimate experiment timespan` Gives a rough approximation of how much the experiment will take and when will it end based on the computers time, the user may use this function to play around with the experiment parameters before committing to an experiment and launching it. This approximation is estimated for the speeds and accelerations that the application is originally shipped with and may differ from the actual elapsed time if the user edits them
- `Launch experiment` Will launch an experiment only if the user input values are safe and the intialization has been completed

[4] The amplification level can be set manually in the LIA's panel through the input range controls.

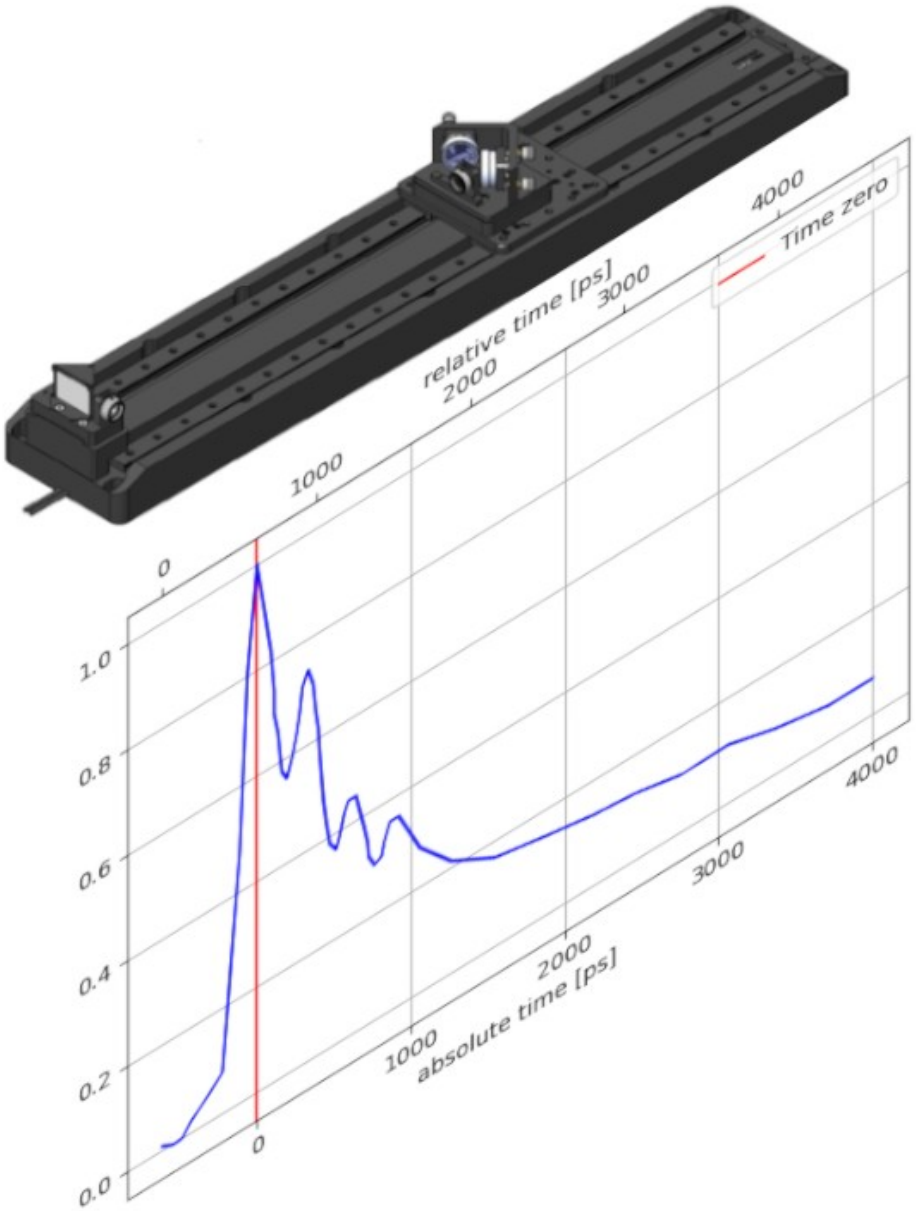


Figure 6: Absolute and relative naming conventions, absolute time measures time with respect to time zero, that is 0ps is the delay at which pump and probe pulses coincide, everything before is negative and everything after is positive, this convention is used to specify the start and end positions for every leg. Relative time on the other hand measures time with respect to the delay stage at 0mm displacement, it is only used so that the user can specify where time zero is located initially and then the application will reference all other measurements and outputs to that delay.

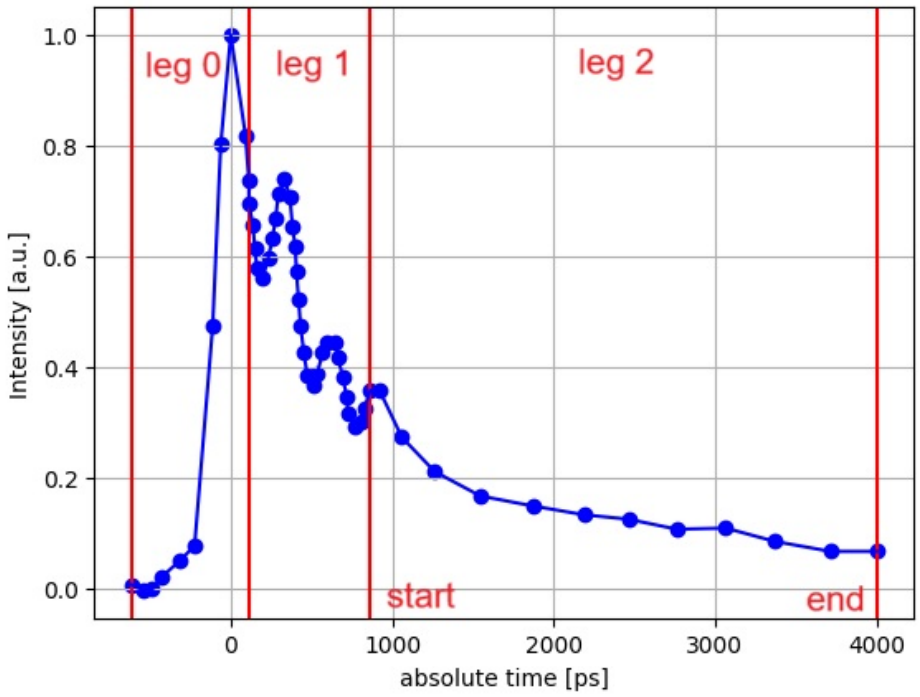


Figure 7: **Scans and legs naming convention:** Illustrated in the screen is a whole scan, each scan can be triggered several times to average them together. Each scan is made up of legs, these legs are ranges where the user may choose to scan with more finer or coarser steps, thus increasing resolution in critical parts without wasting time scanning slowly in non crucial parts. Each leg is defined by 3 different variables, where (or rather when) it starts, when it ends and the length or step of the increments

3.5 Further Manual Customization

The device states that can be configured through this application are limited to the most general use case scenario at the time of development, thus the user might feel like these options are limited, for instance the initialization step configures the LIA to measure voltage signals through input channel A but the user might someday want to measure a current signal through channel B, such a device state cannot be configured through this application, it can only be done manually in the LIA's panel, however since the application will always configure the LIA to measure a voltage signal at initialization the user has to make sure to wait until initialization is finished before manually switching the input to current measurement or else the application might overwrite it back to voltage.

If the user intends to change a parameter manually they should study the different logs on screen to identify when it's an appropriate time to manually overwrite their desired parameter. There are two different logs that the application will display: one during initialization and another while the experiment takes course, in general the application will always inform the user on what it's doing so that the user can decide when to make a manual change such that it is not later overwritten by the application.

3.6 Monitoring Window

Launching the experiment will pop up a monitoring window with various widgets:

- `graph` This window displays in a main graph the experiment as new acquisition points arrive, the user may zoom in, pan, reset view and save a screen capture with the navigation toolbar below it. The graph will display new traces in a gradient of color from black to yellow, additionally it will display an average in dashed blue lines if the number of scans exceeds one.
- `log` The different steps taken by the application are logged on the left so that the user gets a measurement for the state of the experiment. At the end of every scan this log displays a tally up of the time spent on every sub-step (waiting for filter settling) for an average acquisition point, this gives the user an idea of what parameters are taking more time and thus allows them to further optimize the experiment time.
- `Stop experiment early` The user might realize that the experiment was incorrectly set up after launching an experiment and might then wish to halt the experiment execution and return to the Experiment Screen, simply closing the monitoring window won't suffice, since the applications front end and back end run on different threads the experiment will continue silently. Clicking the `Stop experiment early` will cause the application to wait for a specific step in the measuring loop where it is safe to end it. **Warning:** Closing the application's main window while an experiment is running will end the communication with the devices abruptly and their behavior cannot be guaranteed to be safe, please always use this button to halt experiments or wait for an experiment to finish before closing the application
- `estimated finishing time` A text in the screen reminding the user when it is estimated the experiment will finish.

3.7 Saved data

During the experiment execution data is saved in the `Output` folder inside the installations folder `Pump_Probe_Measurement_Automatization`, the name of the folder is selected by the user beforehand by filling the `experiment file name` in the Experiment Screen. Each scan is then stored in it's own independent subfolder, each scan subfolder contains an excel sheet and an image of the graph[5], the content of this scan subfolder reflects the state of the experiment when that particular scan finished, each excel sheet contains the following:

[5]The image of the graph is a screen capture from te graph at monitor window and it will reflect the current zoom and pan set by the user when the scan ended

- a header with the time at which that particular scan ended, and a list of configuration parameters necessary to replicate the experiment (from the applications side at least): `rel time zero`, `time constant`, `filter foll-off` and `input range`[6]
- Following that the user will find the data assorted in columns, the first columns `Absolute time [ps]` displays the delays at which the data was acquired
- `Time absolute on axis error` the second column is placeholder for the delay error, it should be noted that this error should be taken as a lower bound as it only takes into account one of the positional errors in the delay stage and does not consider other time related errors like pulse-width, etc.
- `signal level` the third column holds the acquired data from the LIA at this particular scan
- `signal error` the fourth column is optional and may either not be present, be padded with the same value or hold data for signal errors at every point (see section 3.4)
- `average of previous scans` the fifth column is also optional and may not be present for the first scan since there is no data to average, for scans above 1 it will always be present, however the user must note that since scan subfolders are created at the end of every scan the average at a specific subfolder (say scan 3) will only represent the average for scans up to that one (1 through 3) while the average at a subsequent scan folder (say the folder for scan 9) will hold an average computed for more scans (1 through 9), in essence **the average computed for all scans is stored in the last scan subfolder**

3.8 Errors

Software errors that might occur during the application's operation are in general displayed as error boxes to the user and should not crash the GUI, these error messages are "propagated up" through the code, that means that a user familiar with the code would in principle be able to track at which function did the error originate, the error message may include tips for the user. Following is a short non-comprehensive list of errors known by the developer and how to troubleshoot them.

Delay stage with serial number ... not in device list First check whether delay stage driver is switched on, if not switch it on, connect and disconnect it's USB. If that doesn't work check whether Kinesis software is open, close it and connect and disconnect again.

LIA not in COM5 The user has verified that the LIA **is** recognized by the PC in device manager but it is not recognized at the default port `COM5` and is instead showing at some other port like for instance `COM3`, to change the default port to the new port (*or any of the default configuration parameters*) go to the installation folder and open the following file with a text editor like VSC or Notepad++

```
"C:\...\Pump_Probe_Measurement_Automatization\Utils\default_config.json"
```

this file is a JSON (*see JSON python docs*) and it contains the default parameters to initialize the devices, editing any of the parameters from their default

```
"USBPort": "COM5",
```

To something else

```
"USBPort": "COM3",
```

Will change the default parameters. **Warning:** Changing default parameters like Acceleration and Maxvelocity above it's limit the delay stage, edit these parameters at your own risk.

Note: While logs are active during initialization or monitoring some error messages may be captured by the log and displayed in screen, additionally some error messages are not as verbose in the messageboxes as

[6] This input range is only captured at the end of the scan and does not guarantee that it was used throughout the whole scan

they are in the command line, if you are looking to troubleshoot an error you may open X-WavesLauncher.py with your IDE of choice and change line 22 to the following and save

```
print_on_cmd = True
```

This will skip printing errors in logs which allow you to see the errors printed on a terminal. To run from terminal open the installation folder, right click and open CMD/PowerShell and run:

```
Measurement_Automatization_venv\Scripts\Ativate
python X-WaveSLauncher.py
```

“Final report of the commercial starship Nostromo, third officer reporting. [...]
This is Ripley, last survivor of the Nostromo, signing off.”

— Ellen Ripley*, Alien (1979)*