USB LOCKIN 250
LOCKIN AMPLIFIER 10 mHz to 250 kHz

Anfatec Instruments AG
Melanchthonstr. 28
08606 Oelsnitz /V.
Germany
Tel.: +49 (0) 37421 24212
Tel.: +49 (0) 37421 24221
http://www.anfatec.de
email: mailbox@anfatec.de

Table of contents

1 GENERAL INFORMATION

1.1 SPECIFICATION OF THE USBLOCKIN250

1.1.1 GENERAL PARAMETERS

Digital Dual-Phase Lock-In Amplifier	
Dynamic Reserve	> 135 dB (1)
Input Noise (low noise mode)	< 4 $nV_{rms}/Hz^{0.5}$ @ 100 kHz
Remote Control	USB 2.0
Time Constants	10 µs ... 5 s / 1 µs in sync mode
Sensitivity	10 nV ... 10 V
Phase Resolution	0.0001°
Amplitude Deviation (0..250 kHz)	< 1 %
Phase Deviation (0..250 kHz)	< 0.5 °
Maximum Frequency	2 MHz
Weight	450 g
Housing Size	20 cm (L) x 10.5 cm (W) x 5 cm (H)

1.1.2 SIGNAL INPUT

Voltage Input	BNC
Input coupling:	dc or ac (f_{-3dB} = 2 Hz)
Input Impedance	1 MΩ || 20 pF
Damage Threshold	+/- 12 V
Input sampling rate	20 MHz
Bandwidth	dc to 250 kHz (f_{-3dB} > 250 kHz)
Input Ranges	± 3.5 Vrms, ± 350 mVrms, ± 35 mVrms
Input Sensitivity	10 nV to 10 V
Typical Input Noise:	Uac = 0 V, 50 Ω @ input → see table
RollOff	6 dB/oct, 12 dB/oct, 24 dB/oct
Time Constants	10 µs ... 5 s
Amplitude Accuracy (dc to 250 kHz)	< 1 %
Gain deviations between dynamic ranges:	< 1 %
Phase shift accuracy (dc to 250 kHz)	< 0.5°

Input Noise measured with Uac = 0 V and 50 Ω @ input

Frequency	Filter \ Range	High dynamic reserve	Normal	Low Nosie
250 kHz	10 ms	< 60 $nV_{rms}/Hz^{0.5}$	< 7 $nV_{rms}/Hz^{0.5}$	< 4 $nV_{rms}/Hz^{0.5}$
100 kHz	10 ms	< 60 $nV_{rms}/Hz^{0.5}$	< 7 $nV_{rms}/Hz^{0.5}$	< 4 $nV_{rms}/Hz^{0.5}$
10 kHz	10 ms	< 60 $nV_{rms}/Hz^{0.5}$	< 8 $nV_{rms}/Hz^{0.5}$	< 6 $nV_{rms}/Hz^{0.5}$
1 kHz	1 s	< 1 $\mu V_{rms}/Hz^{0.5}$	< 160 $nV_{rms}/Hz^{0.5}$	< 15 $nV_{rms}/Hz^{0.5}$

1.1.3 Reference Output

Internal Oscillator	10 mHz .. 250 kHz (1 MHz)
Frequency Resolution	< 10 mHz
Frequency Accuracy	+/- 25 ppm from 0 °C to 70 °C [1]
Amplitude Accuracy (dc to 250 kHz)	< 0.2 % + 10 µV (Out to In) [2]
Reference Output Voltage	< 1 mVpp ... 15 Vpp
Output Noise @ 100 kHz	@ Uac = 1 mV, < 200 $nV_{rms}/Hz^{0.5}$
Output sampling rate	>= 20 MHz

1.1.4 External Triggering by PLL

Frequency range	1 Hz .. 1 MHz
Locking time	< (100 ms + 10 Cycle)
Phase error	< 4 deg @ f < 1 kHz
Input amplitude	TTL- and Sine-Mode (> 20 mV_{rms})
Phase delay	~ 1 µs

1.1.5 Analogue Outputs

Sampling rate	250 kHz
Resolution	18 bit
Output Voltage	-10 V … 10 V
Output Noise	< 300 nV/sqrt(Hz)
Accuracy	< 2 %

1.1.6 General

Interface	Plug & Play USB 2.0 interface
Drivers	Windows NT/2000, Windows XP, Win7 32 bit
Power consumption	12 Vdc, 1 A
Warranty	2 years

1.1.7 Standard Part List

USBLockIn250	1
Power Supply 12 V dc	1
USB cable M-M , 2 m long	1
Manual	1
Software on USB-stick	1
Low-Noise BNC cable 2 m long	1

1 starting from S/N -200
2 revised on 8.1.2019

1.2 LICENCE FOR THE SOFTWARE

The Source code for the remote control software and the DLL source code are provided with the General Public Licence (GPL). All examples for LabView can be freely implemented and changed by the user.

2 INSTALLATION

2.1 SYSTEM REQUIREMENTS

- compatible PC
- 20 MB free hard disk space
- Windows NT 4.0 / Windows 2000 or Windows XP (32 bit) or
- Windows 7 (32/64 bit) or
- Windows 10 (32/64 bit)

Figure 1: Power Supply for 12 V dc. USB-Connector.

2.2 GET STARTED

- Connect the power supply to "Power"
- Connect the USB interface to your PC
- When Windows asks for a driver, then provide the path to the driver on the USB-Stick under "Driver" depending on your Windows Version. Otherwise see "Driver Installation".
- Start the software "lockin.exe" once as **Administrator**[3]

2.3 PROVIDED SOFTWARE

On the delivered USB Stick, you find

Driver	in	.\Driver
GUI software	in	.\App\Win32\LockIn.exe – start to test the device under 32 bit
	in	.\App\Win64\LockIn.exe 1 – start to test the device under 64 bit .\App\Win64\LockIn.exe 2 - if two devices should work simultaneously

NOTE: In Windows7, the ini-file "user.ini" might be copied automatically in another directory. If you want to avoid this, make the directory that contains the ini-file writeable by any user.

Source Code of the lockin.exe	in	.\Manual\App\Delphi7 .\App\DelphiXE7
DLL file	in	.\DLL\Win32 or .\DLL\Win64
Examples using the DLL	in	.\DLL\....
Source Code of the DLL	in	.\DLL\DLLSourceXE7

2.4 SOFTWARE INSTALLATION

- Install the correct driver (32 bit or 64 bit) – if not yet done during plug-in
- Run the lockin.exe once as Administrator

For DLL-based applications:

- Check whether the application is 32 bit or 64 bit
- Copy the related DLL (.\DLL\Win32 or .\DLL\Win64) to your project folder

3 This copies the scaling data for the lockin amplifier into the registry of the PC. Afterwards, all other applications and all users can use these registry entries.

2.5 Driver Installation

Connect the USB lockin first. If the system asks for a driver, browse on the USB-Stick for:
.\Driver*version*\x86 for 32 bit or
.\Driver*version*\x64 for 64 bit operating system.
Select the inf-file.

If the system does not ask automatically, there are two possibilities to solve the problem:
A) install through device manager

- Click on Start
- Enter "devmgmt.msc" in the Search Line to open the Device Manager
- Select the "Anfatec Lockin Interface" and right-click to "Update Driver Software"
- "Browse for driver software on your computer"
- provide the path "%YourPath/Driver/version/.."
- select the inf-file
- Say OK when Windows recommends not to install from an unknown driver vendor

B) install with setup program in "C:\Users\USBLockin\Driver"
In Windows XP and Windows 2000 start "setup.bat".
In Windows 7 start "setup7.bat".

Allow the program "setup.bat" - to make changes on your PC. During installation, a command window is opened and closed when finished. For the first installation, this procedure might take about 20 seconds.

2.6 Connections to the USBLockIn250

Power - requires 12V DC, 1 Amp
- Innerpin is positive

USB - interface for remote control via USB2.0

"Ref-In" - is the TTL or sine wave input that allows to trigger the USBLockIn250 from any external source. In order to use it, one has to set the PLL ON with the software.

"TTL-Out" - is a TTL type (rectangular 5 V wave) trigger output for other devices. Use Ref-Out, when a sine wave is required.

"Aux1" - is the analogue output No 1. It is connected to either

- the Panel meter 1 or
- PID output or
- constant voltage output

It provides an analogue signal of the software-selected channel with the software-selected scaling.

For more information here.

"Aux2" - it is connected to either

- the Panel meter 2 or
- PID output or
- constant voltage output

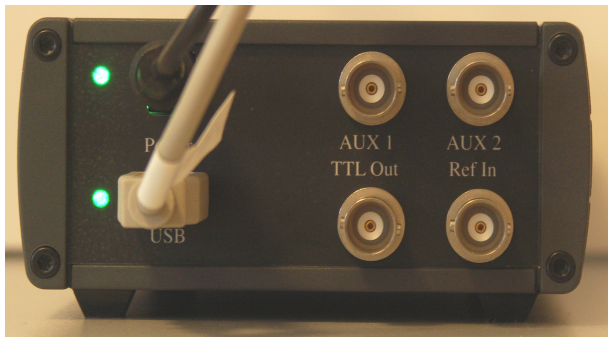


Figure 2: Backside of the USBLockIn250. The top LED is the power LED. The bottom LED indicates that the USB connection is ON.

Figure 3: Front side of the USBLockIn250. The red LED indicates an overload.

It provides an analogue signal of the software-selected channel with the software-selected scaling. For more information see page 15.

LEDs - the USB LockIn has three LEDs, one on the front side and two on the backside. The top backside LED (next to the power connector) indicates that the power supply is connected. The bottom backside LED (next to the USB connector) indicates that the USB interface is connected. The front side LED indicates an overload when it starts shining in red. During normal operation this LED is invisible.

"IN" - is the analogue input of the lockin in which the signal to be detected is plugged in.

"RefOut" - is a sine wave output whose amplitude is set by software and whose frequency equals the currently selected centre frequency of the lockin amplifier.

2.7 Driver Update

Install updated drivers through device manager

- ☑ Click on Start
- ☑ Enter "devmgmt.msc" in the Search Line to open the Device Manager
- ☑ Select the "Anfatec Lockin Interface" and right-click to "Update Driver Software"
- ☑ select "**Browse my computer for driver software**"
- ☑ provide the path "%YourPath/Driver/version/.."
- ☑ select the inf-file

Important Note: It is necessary to update the driver for each USB port at which the USBLockIn250 will be connected. Otherwise, the old driver will be used.

3 LOCKIN AMPLIFICATION BASICS

3.1 THE GENERAL IDEA OF LOCKIN AMPLIFICATION -> LOCKIN

A lockin amplifier is a phase sensitive bandpass filter with a centre frequency f and a bandwidth Δf. It has two output signals in parallel: either the amplitude R and the phase φ, or the real part X and the imaginary part Y.

- The centre frequency f is the frequency at which the LockIn is searching for the input signal.
- The bandwidth Δf provides the frequency range around this centre frequency, from which the output signal is derived.
- The output signal R is the measured amplitude of the analysed signal at the centre frequency f.
- The phase φ is the relative phase shift between the analysed signal and an internal reference signal with the same frequency.

For analysing the input signal, the LockIn needs an internal frequency reference with a certain amplitude and phase. (see Fig. 4).

The bandwidth Δf of a LockIn can be visualised by a simple setup. A generator applies a constant frequency f_{in} signal to the input of the LockIn and the LockIn is sweeping its centre frequency f around the input frequency f_{in}. Figure 5 shows that the amplitude at the input frequency is always measured with the same value. For other frequencies, the measured amplitude is dependent on the distance between the actual centre frequency and the input signal frequency.

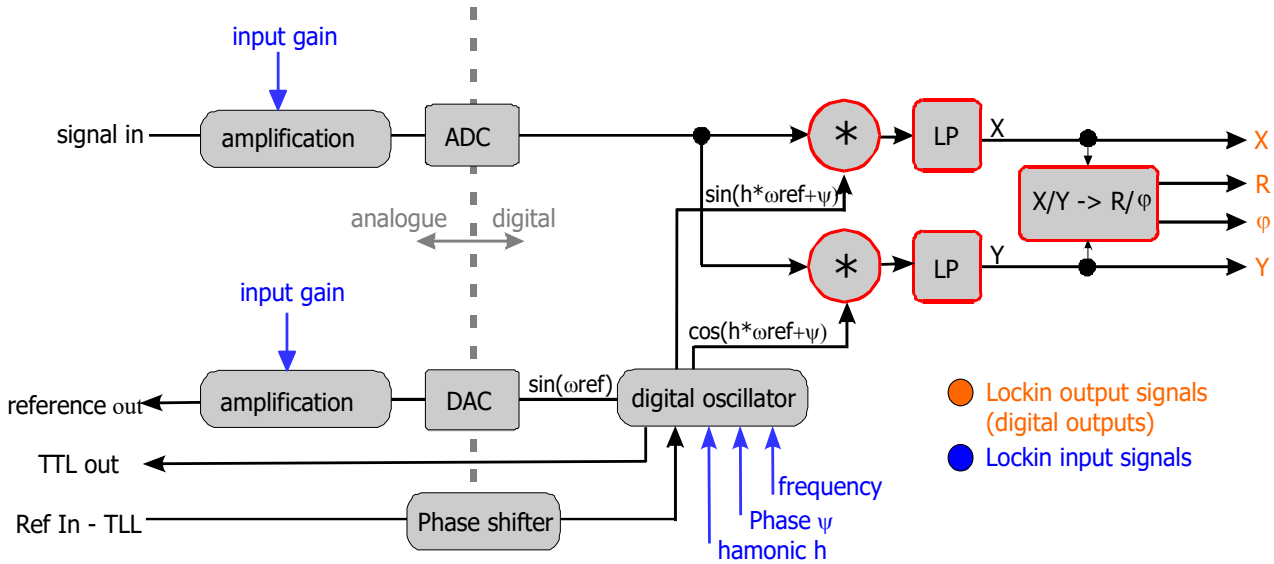


Figure 4: Schematic diagram with the basic components of the lock-in amplifier.

A schematic diagram with the basic components of the lock-in amplifier is shown in Figure 4. First of all the received signal is amplified and digitized. Divided into two separate channels, the signal is multiplied by the mentioned reference signal (with frequency ω_{ref}) and the 90° phase-shifted reference signal respectively. The reference signal is generated with the lock-in's digital oscillator. An additional implemented phase displacement Ψ enables to compensate phase differences caused by the measuring equipment. After multiplication, the resulting signals are low-pass filtered and provide now information about the real part X and the imaginary part Y of the analysed signal relating to the phase position of the reference signal. Out of them the amplitude R as well as the relative phase shift φ are calculated.

3.2 MATHEMATICAL DESCRIPTION

A digital LockIn realises it's filtering by certain mathematical procedures.
Think, Y(t), the signal to be analysed, is dependent on a value X(t). For instance: we measure a photo voltage U(t) which depends on the light intensity given by an LED I(t). If the background light intensity is much too high to detect small intensity changes of the LED light, one can modulate the light intensity I(t) at a certain frequency ω_{ref} and detect with a narrow band filter at this frequency.
This X(t) is modulated as $X(t) = X_0+X_1 \cos(w_{ref}\, t)$. Then, Y(X(t) can be developed into a Taylor-Series:

$$Y(X_0+X_1\cos(\omega_{ref}\,t))=\sum_{k=0}^{\infty} Y^{(k)}(X_0)\cdot\frac{X_1^k}{k!}\cdot\cos^k(\omega_{ref}\,t)\cdot Y^{(k)}(X)=\frac{d^k Y}{dX^k}|X_0 \quad k\in N$$

Equ.(1)

For the value of Y at the time t, all measurement values Y(t) detected in the period τ are used. This period τ should be a whole-numbered multiple of $2\pi\omega_{ref}$. Due to the modulation of X(t) at ω_{ref}, modulations of Y(t) at the frequencies $m \cdot \omega_{ref}$ with m = 1,2,... are expected. These modulations equal the Fourier components $Y(m \cdot \omega_{ref})$ with m = 1,2,... of the input signal Y(t), which are given as:

$$Y(m\omega)=\frac{1}{\tau}\int_{-\tau/2}^{\tau/2} Y(X(t))e^{-im\omega t}dt$$

and together with Equ. 1

$$Y(m\omega_{ref})=\sum_{k=0}^{\infty} Y^{(k)}(X_0)\frac{X_1^k}{k!}K_m^k$$

with

$$K_m^k=\frac{1}{\tau}\int_{-\tau/2}^{\tau/2}\cos^k(\omega_{ref}\,t)e^{-im\omega_{ref}t}dt \quad .$$

The factors K_m^k get zero for k to infinity. For small k, the K_m^k remain too big to be neglected. In order to neglect n-th order parts, the n-th derivative of Y(t) to X(t) has to be negligible.
The first 10 (k = 1 ... 10) coefficients K_m^k for the first 4 harmonics m = 1 ... 4 are given in the following table:

	k =									
m	1	2	3	4	5	6	7	8	9	10
1	$\frac{1}{2}$		$\frac{3}{8}$		$\frac{5}{16}$		$\frac{35}{128}$		$\frac{63}{256}$	
2		$\frac{1}{4}$		$\frac{1}{4}$		$\frac{15}{64}$		$\frac{7}{32}$		$\frac{105}{512}$
3			$\frac{1}{8}$		$\frac{5}{32}$		$\frac{21}{128}$		$\frac{21}{128}$	
4				$\frac{1}{16}$		$\frac{3}{32}$		$\frac{7}{64}$		$\frac{15}{128}$

3.3 Time constant Definition

The "Time constant τ" is a parameter of the low pass filter after demodulation. It defines the detection bandwidth Δf_{Det} of the lockin amplifier.
In order to visualize this detection bandwidth, the input of the lockin amplifier is connected to a constant input signal (here: 1 V_{rms} amplitude, 100 kHz). The internal reference frequency is swept over the input frequency.

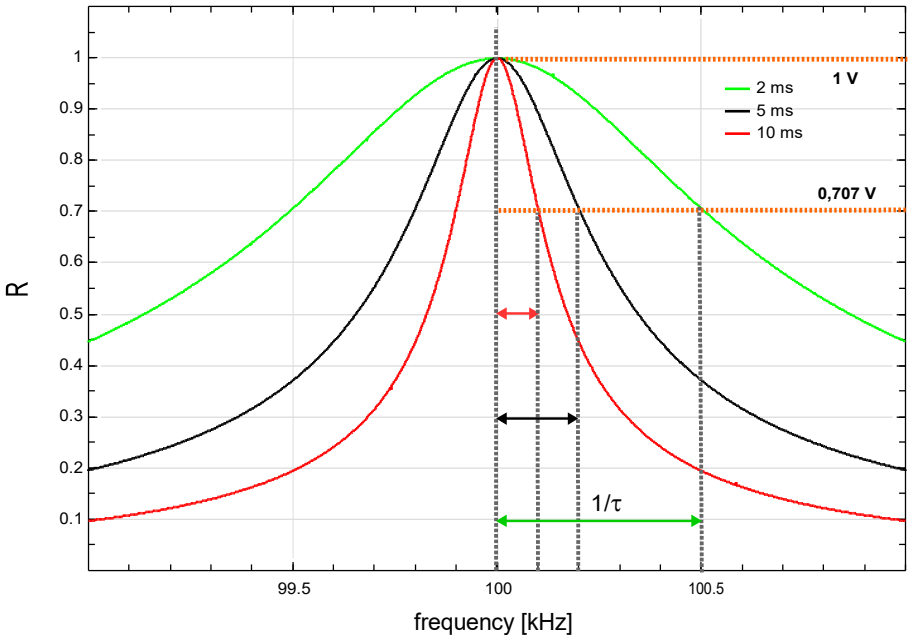
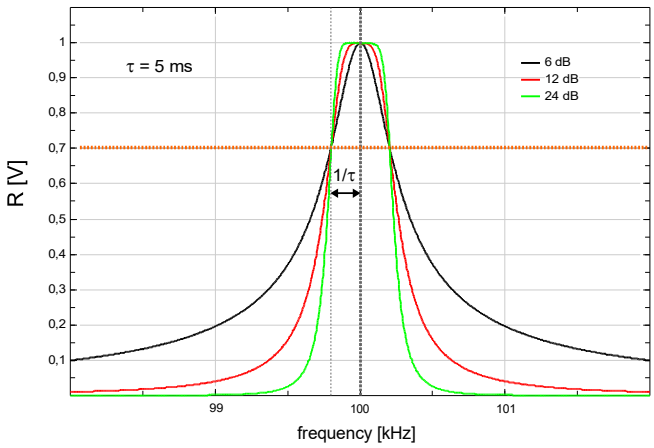
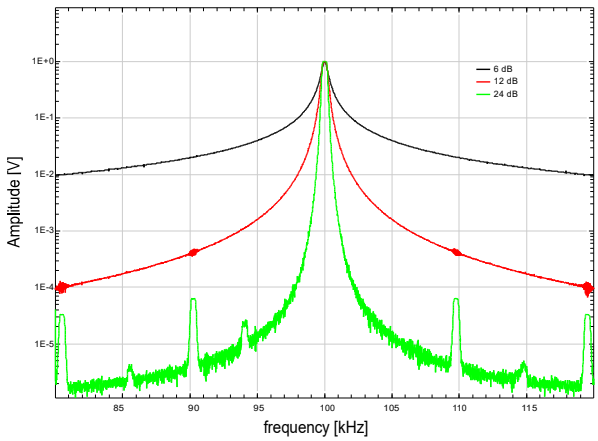


Figure 5: Visualization of the Detection Bandwidth for various time constant settings.

A signal, which is Δf_{Det} off the centre frequency of the lockin amplifier, will be detected with 0.707 times its amplitude.

3.4 RollOff Definition





The slope of the signal decay in Fig. 5 is defined by the Roll-Off without affecting the detection bandwidth.

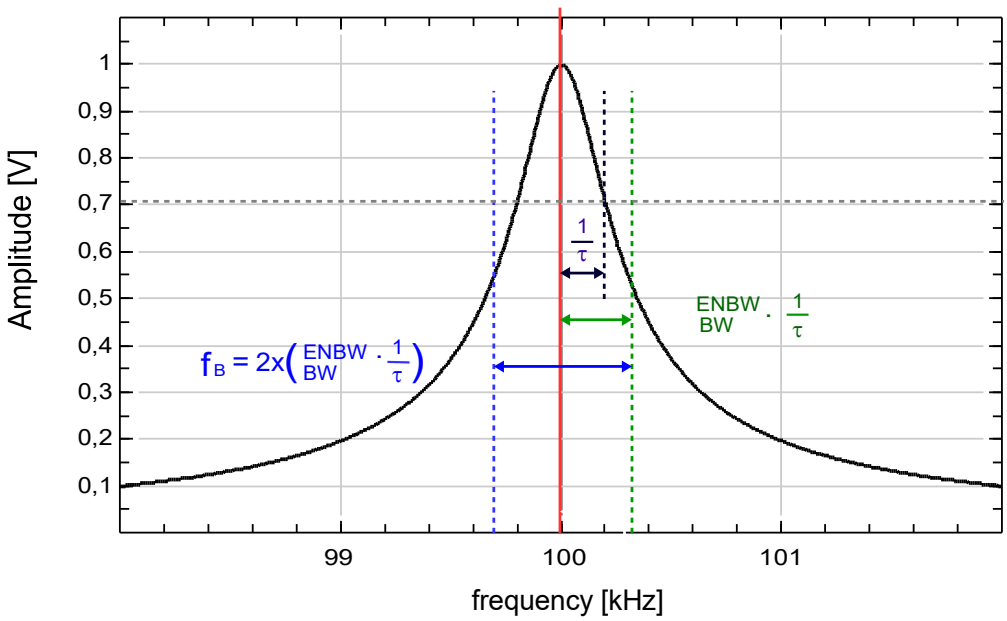
3.5 Noise Measurements

Lock-in amplifiers are capable to measure noise. They detect a signal at the centre frequency $f_{ref} = \omega_{ref} / 2\pi$ with an equivalent noise bandwidth. The equivalent noise bandwidth (ENBW) of a Butterworth low pass filter is the bandwidth which passes the same amount of noise as a perfect rectangular filter with the equivalent bandwidth.

The bandwidth of the lockin amplifier is determined by the slope and the equivalent noise bandwidth of the used Butterworth filter.

The normalized Butterworth filter noise bandwidths are:

Filter order n → RollOff	**equivalent noise bandwidth Butterworth Filter ($ENBW_{BW}$)**	**1/sqrt($ENBW_{BW}$)**
1 → 6 dB/oct	1,570796	0,79607
2 → 12 dB/oct	1,110721	0,948849
4 → 24 dB/oct	1,026172	0,987165



In order to scale noise spectra, the resulting data should be divided by the square-root of the used LockIn bandwidth.

$$f_B = \frac{2}{\tau} \cdot ENBW_{BW} \qquad u_n = \frac{u_{meas}}{\sqrt{f_B}} = u_{meas} \cdot \sqrt{\frac{\tau}{2}} \cdot \frac{1}{\sqrt{ENBW_{BW}}}$$

The integrated spectrum is taken with a time constant of 5 ms and a slope of 24 dB/oct (4th order). The ENBW factor is 1,03. The related bandwidth is then: f_B = 412Hz. In order to interpret an measurement of 2µV as noise in units of V/√Hz, the value should be divided by √412Hz. It results a value of 98,5nV/√Hz.

4 HARDWARE DESCRIPTION

4.1 AUXILIARY OUTPUTS

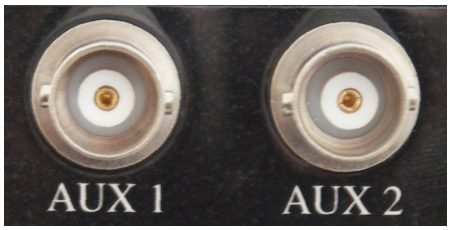


Figure 6: Analogue outputs with variable use.

The backside of the USBLockIn250 supports two analogue outputs named "Aux1" and "Aux2". Each can be used as

A) scaled outputs of the LockIn,
B) PID control output or
C) constant voltage output.

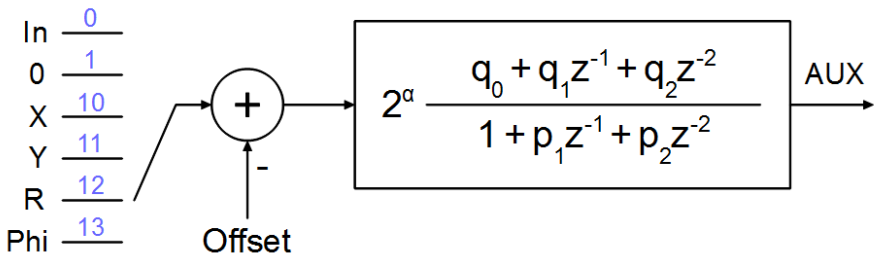


Figure 7: Principle of the AUX output generation

A) Scaled outputs of the lockin amplification results In, X, Y, R or Phi.

Note: use the right mouse button on the text of the meters to select the output scaling!! (see Fig. 9 referring to the *lockin.exe* software surface)

For X, Y, and R, one can set a range between 10 nV and 10 V. The lockin amplifier scales the data to -10 V... 10 V, so that "range" equals 10 V.

```
Example:
Assume, the lockin amplifier measures 10 mV amplitude signal (on R).
When the output scaling is set to a value below 10 mV (1E-8 … 10 mV),
Aux1 will show +10 V which equals an overload.
When the output scaling is set 20 mV for example, then these 20 mV equal
10 V maximum range and 10 mV equal 5 V output in relation to the 20 mV
range.
When the (maximum) range is 200 mV, then 200 mV signal would be shown as
10 V; and 10 mV are scaled to 500 mV, only.
```

This works analogue for Phi.

In order to access this functionality with the DLL, see description for command "SetLockInAux".

B) PID Output
The implemented algorithm is a PID lowpass from [1]. If Ti is set to "inf", simple PD is used.

[1] "Digital Self-tuning Controllers" Springer Publisher´s Chapter 4.2.1

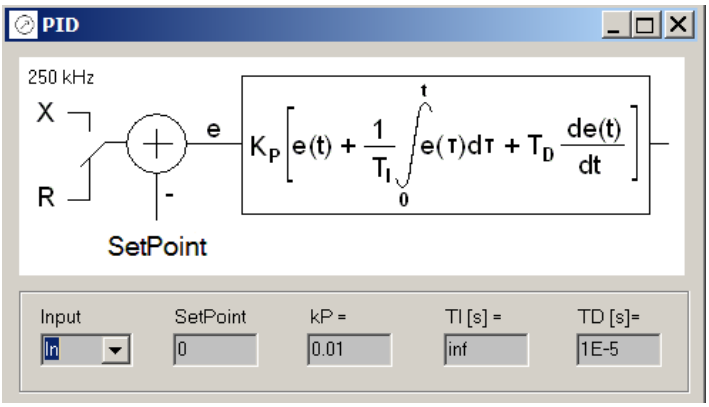


Figure 8: Schema of the analogue equivalent of the digitally implemented PID feedback.

Gray user inputs mean that the values are not set.

In order to access this functionality with the DLL, see description for command "SetLockInPID".

C) Constant Voltage Outputs defined by the user to control external experiments.

The output of a constant value of 1 V can be achieved by entering the following values in the PID form:

In order to access this functionality with the DLL, see description for command "SetLockInAux" with *"input"* = 1.

5 SOFTWARE DESCRIPTION (LOCKIN.EXE)

5.1 OVERVIEW

The GUI software *lockin.exe* opens with the following surface:

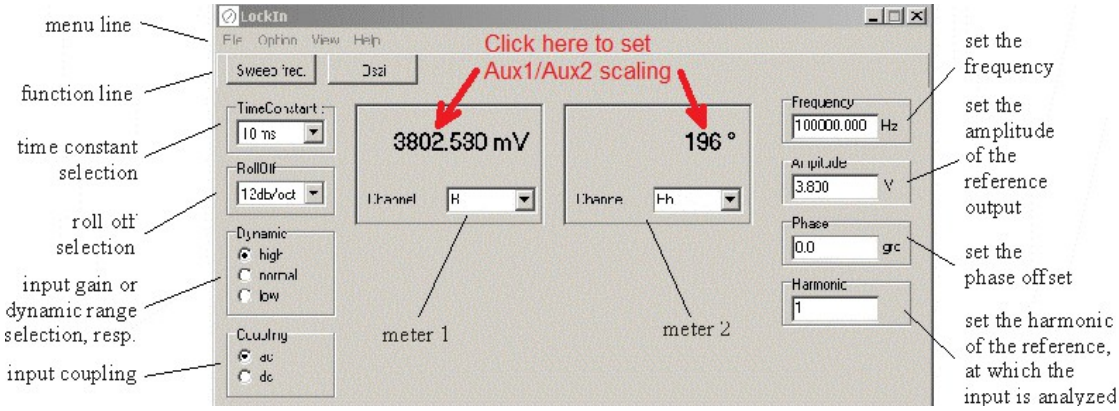


Figure 9: Main window of the LockIn program with description of the functions.

Figure 9 shows the main window of the LockIn program, which appears when the program is opened. For checking the actually detected values, the two meters are used. Basic input parameters (frequency, amplitude, offset phase and harmonic) can be selected in the right part of the window. Parameters, which concern the input stage (time constants, RollOff, and input gain) are chosen in the left part of the window.

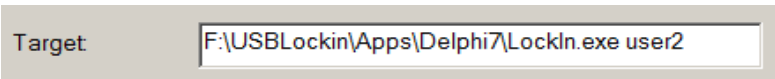
The frequency is either the internal frequency (black numbers) or the detected external frequency from the reference input (grey numbers).

The menu line allows typical Windows functions, while the button in the function line opens new windows with specific functions.

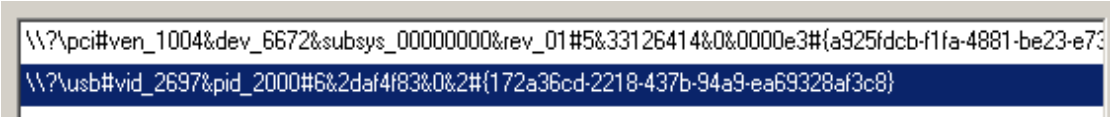
This software can be started as often as many USBLockIn250 are attached to the system. In order to start various USBLockIn250 with dedicated parameters, follow these steps:

1. Create as many desktop shortcuts to the *lockin.exe* as many devices you wish to operate

2. Right-click on each desktop shortcut and choose "Properties"
3. Add a parameter (e.g. user2) behind the entry in the target line:



4. Start each shortcut once, select the device to be associated with this shortcut:



The same procedure can be used to open the software with different settings.

5.2 Functions in the Menu Line

5.2.1 File

Exit - Exit the program

5.2.2 Option

- **overload**

Overload occurs, when the dc input signal exceeds the full scale sensitivity for the selected range. This full scale sensitivity is 7 V_{rms} for high reserve, 700 mV_{rms} for normal reserve and 70 mV_{rms} for low reserve. With this option can be selected whether a beep and/or a colour change is shown in case of overload.

- **ext. reference**

For LockIn-amplifier versions with PLL (Phase locked loop), the reference frequency can be either the internal oscillator frequency or an external reference frequency. If the "ext. reference" is selected, the internal PLL is enabled. The locked frequency is shown in grey in the frequency window. If there is no input signal connected to the reference input, a default frequency of 2 mHz is shown, but the LockIn is not working properly.
With disabled PLL, the frequency given in the frequency window is written in black.

5.2.3 View

It can be selected, which of the meters is shown.

- **Devices**

Select whether the strings for the lockin amplifier(s) should be shown. Useful when using multiple lockin amplifiers to select which one should be used. Is also used for LabView (see chapter "Multiple LockIn Amplifiers in LabView").

5.2.4 Help

"About" – shows the current program version and the version of the used hardware driver
"LockIn help" – calls the table of contents of the HTML help file supplied with the program
"Help as PDF" – calls the current manual (Manual.pdf) and opens it in the Microsoft Internet Explorer

5.3 Functions in the Function Line

5.3.1 Frequency Sweep

Opens the "frequency sweep" window

5.3.2 Oscilloscope

The oscilloscope works like a real 3-channel-oscilloscope. Content, scaling type and offset of the three channels is selectable.

Channel selection: is done from a drop down list, which shows only the available channels.
The two *numbers* above the drop down list for channel selection display the "scaling factor per vertical division" (= left number, hint: "y-scale in /div", example: 12.90 nV/div) and the mean value. Both are calculated from all data acquired from the left oscilloscope edge till the current oscilloscope time. Therefore, these numbers are subsequently re-calculated.

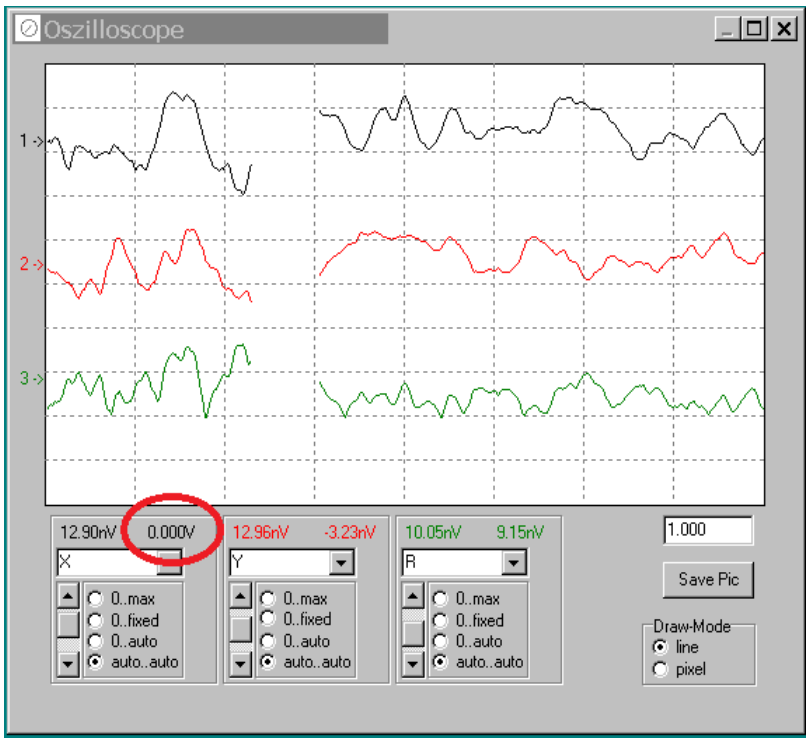
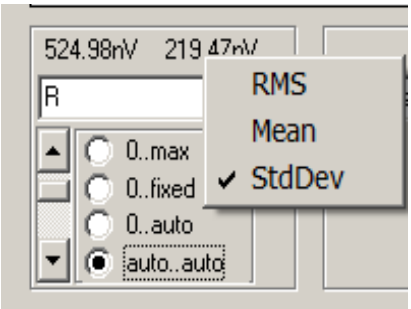


Figure 10: Screen shot of the software oscilloscope

Vertical scaling types:

- 0..max — the scaling is set to maximum value of the channel
- 0..fixed — the maximum value can be changed by a slider appearing on the right sight of the scaling type selection
- 0..auto — the program calculates the the optimum, but takes always "0" as minimum
- auto..auto — automatically scaled

"Time scaling" is done with the edit window (right sight) in seconds.
"Save Pic" saves the oscilloscope screen in a bitmap file.
"Draw mode" selects whether the data are drawn as dots or lines.



Right-click on the right number to change its meaning from RMS to Mean or StdDev.

5.4 Parameter settings

5.4.1 Time constant

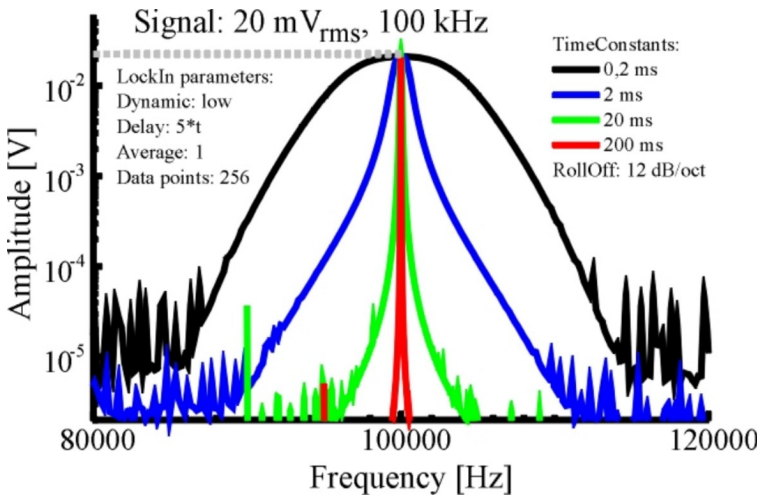


Figure 11: Effect of different time constants.

This option selects the used time constant τ for the low-pass filter. If applicable, the closest approximation of the digital equivalent to an analogue Butterworth filter is used. The corner frequency of the Butterworth filter is $1/\tau$.

5.4.2 RollOff

The meaning of the "RollOff" is shown in the curves in this Figure. It equals the degree of the lowpass filter. One can chose between 6 dB/oct (1st order), 12 dB/oct (2nd order) and 24 dB/oct (4th order).

5.4.3 Dynamic

This switches the input amplification of the LockIn. With "high" dynamic, input amplification is 1. The maximum signal amplitude is then +/-10 V. The "normal" input amplification is 10, which equals maximum signal amplitudes of +/- 1 V. When the low dynamic is chosen, the resolution of the LockIn is highest, but the signals cannot exceed 100 mV.

5.4.4 Coupling

If the specification of the instrument allows it, this option switches between DC coupled input and AC coupled input. Note: The 3dB corner frequency of the input high pass is around 2Hz. Reference frequencies around 2 Hz and below may cause misleading results.

5.4.5 The Meters

The meters display the LockIn output channels, which are X and Y as well R and Phi, in physical units.

5.4.6 Frequency

If written in black letters, this is the actual reference frequency which is used at the reference output and as reference frequency for the signal evaluation of the input.
Click with the right mouse button to switch from external to internal reference. In case of internal reference, the numbers are written in grey.

5.4.7 Amplitude

This is the amplitude of reference output.

5.4.8 Phase

Allows to give a phase offset between the reference output and the input. In the schematic in Fig. 4 , this phase is equal to the LockIn input parameter "Phase ψ".

5.4.9 Harmonic

Selects, which harmonic of the reference frequency is evaluated. The possible values range from 1 to 15. When selecting higher harmonics, take care, that, due to lowpass filtering, the maximum input frequency the LockIn cannot be higher than 2 MHz.

5.5 Sweep Frequency

This window serves the acquisition of frequency dependent spectra of any of the LockIn input channels. The frequency sweep uses the internal reference. When the PLL is enabled for normal operation, it will be disabled during frequency sweep.
The number of data points, parameters for the visualisation as well as for the saving and copying the acquired data can be changed in the option window.

5.5.1 Window Description

"Delay": is the time delay between each acquired data point. During spectrum acquisition, the frequency is set to the next value. Then, the system waits "Delay" and takes one single value from the acquired Channel.
As this delay has to be related to the time constant of the LockIn, the options in the drop down list for the delay are given in multiples of τ. Thus, independently on the time constant τ given in the main window, the time constant for the acquisition of the spectrum is always correct.
"from" and "to" define the values of the start frequency and the stop frequency. For the spectrum's acquisition, one chooses the wanted frequency range, and presses the "start-button". If the time constant was very high, the spectrum might take a while. In order to stop the acquisition, the start-button can be pressed a second time.
"Channel" is a drop down list of available data channels (X, Y, R, and Phi).
"Range back" - click with the right mouse button in the data screen, and a pop-up menu with list of four frequency ranges appears. The upper one is a standard range, which can be changed in the "Option/acquire" part. The next three are, from the bottom to the top, the last three used frequency ranges.

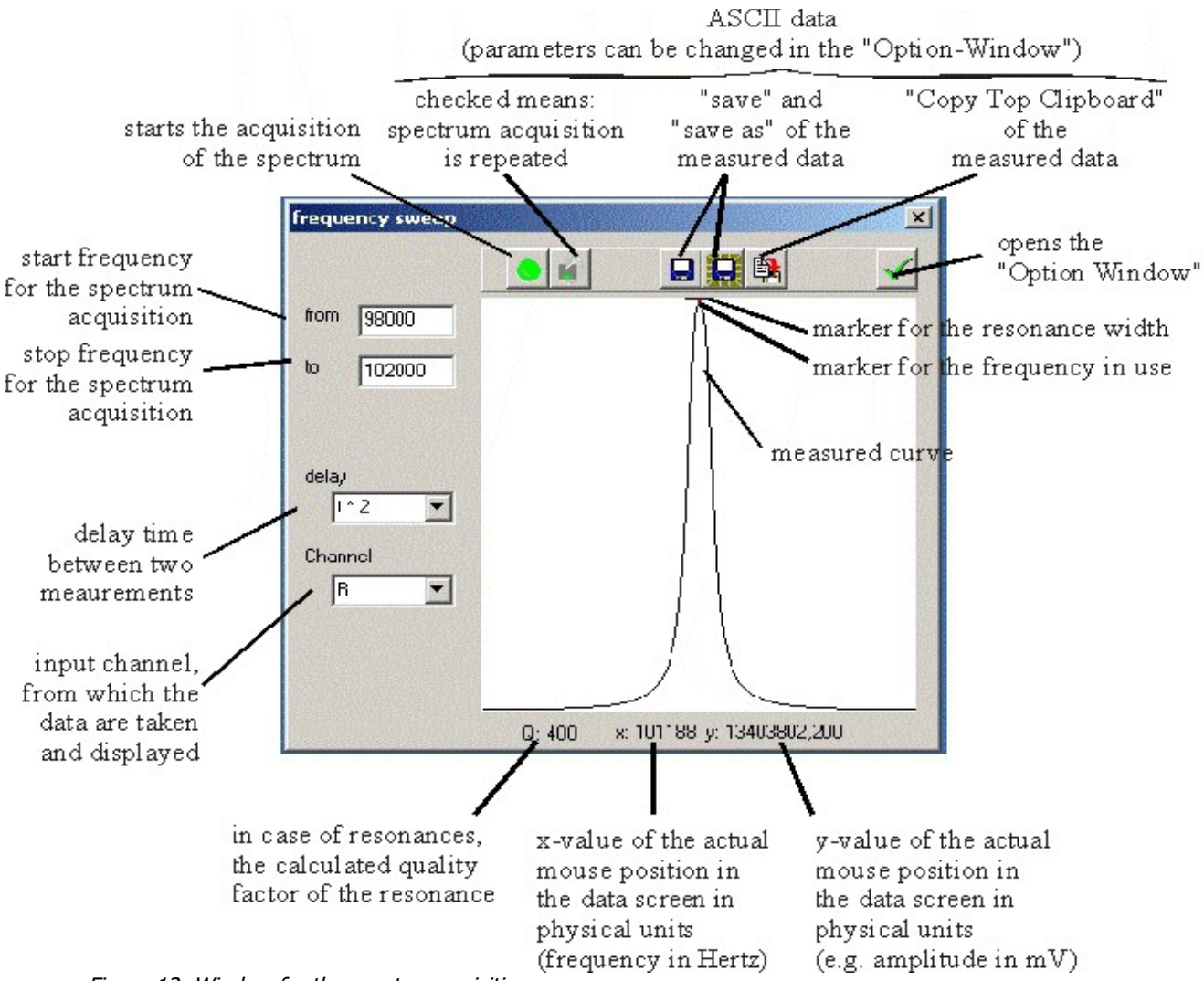


Figure 12: Window for the spectra acquisition.

5.5.2 Options for the frequency sweep

The option window provides three cards: **"save"** - parameters about the saving and copying format of data, **"acquire"** – parameters about the data acquisition, and **"view"** – parameters around the screen of showing the data.

<u>Save-Tab:</u>

The saved files and the data copied to the clipboard have an ASCII structure. The data are written in lines (each frequency value one line) and delimited by the given delimiter ("TAB" in the example) are saved. The frequency values are only saved, if "Save x-Axis" is checked. All history data are saved too and also delimited the same character.

Data file example:

1000,00 234,09
1200,00 237,98 ...

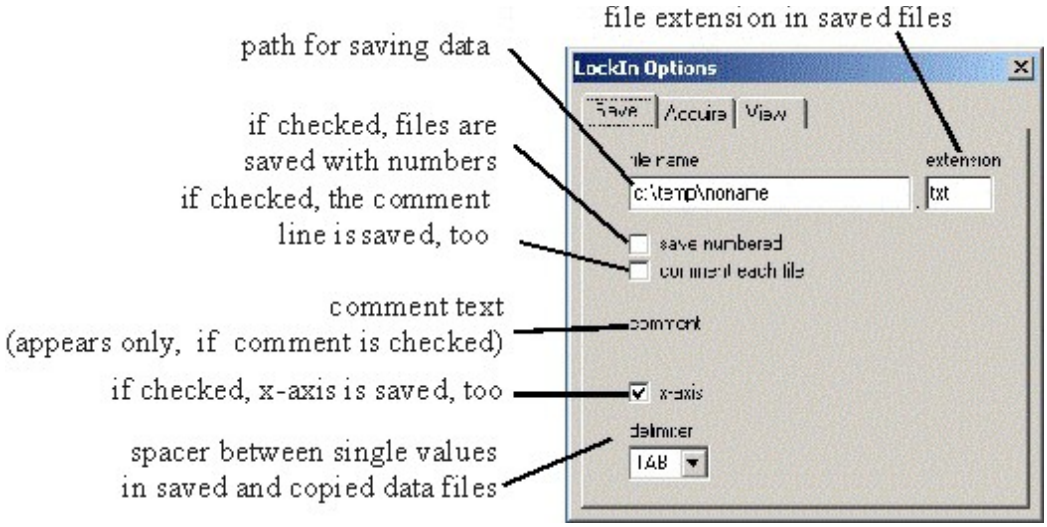


Figure 13: description of the card "save" in the sweep frequency options.

Acquire-tab:

"Wobble" - if a large range is scanned for overview purposes and the frequency peak, which should be found, is too small to be excited (because the single frequency steps are too big), the wobble option can help. If wobble is checked, the frequency is not kept constant during scan. It is varied (wobbled) between the neighboured values while the data are taken. This makes sure, that even small peaks can be found in an overview spectrum with only some 100 data points.

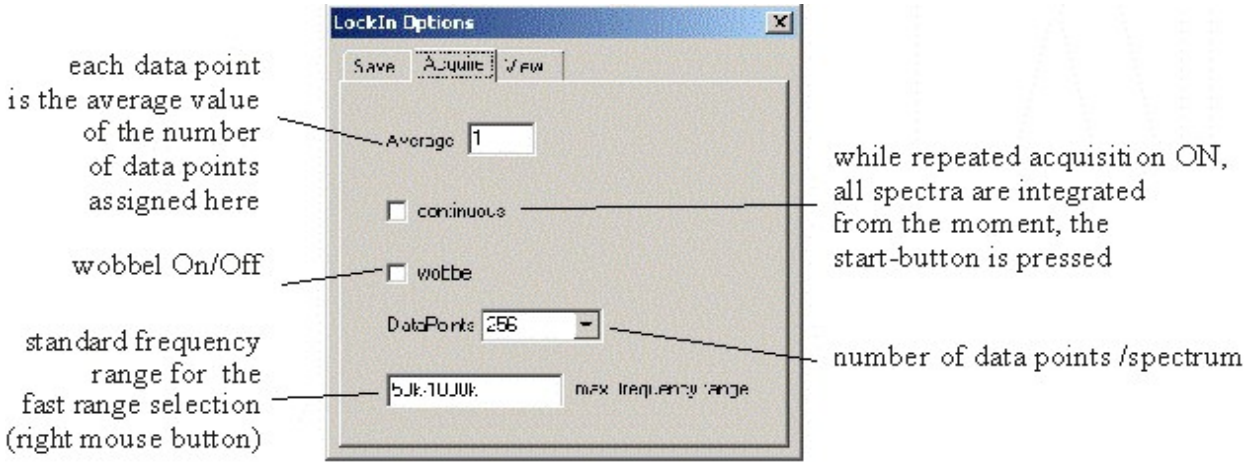


Figure 14: Description of the card "acquire" of the sweep frequency options.

"Standard frequency" - is the range, which appears at the topmost position, if the right mouse button is used in the data screen of the "Sweep Frequency" window.

View-tab:

In the data screen, several data curves can be displayed simultaneously.

Therein, the "History depth" is the number of old curves added to the actual one. If the depth is 2, the actual, the last and the last but one curves are displayed. The actual curve is always of black colour. The last is red, and the last but one is green. More curves get the usual next colours from the Windows™ palette.

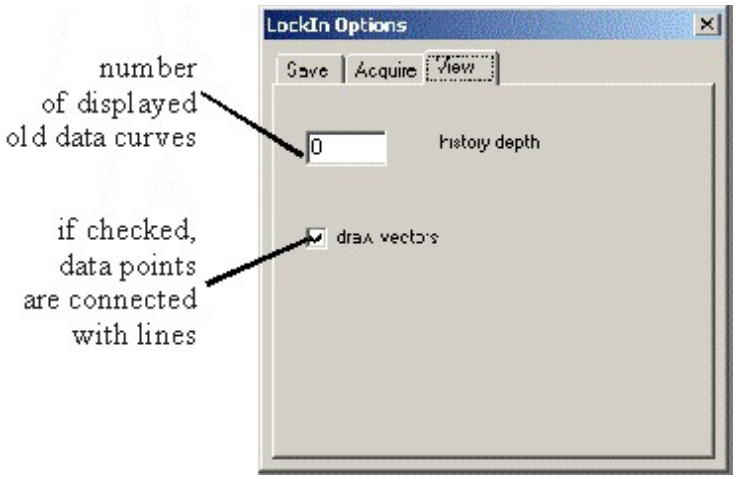


Figure 15: View options for the frequency sweep

If "**Draw vectors**" is checked, the lines are closed, otherwise data points are drawn as pixel, only.

6 EXAMPLES

6.1 DEMONSTRATE TIME CONSTANT

In the following example, we're using the waveform generator 'Picotest 5100A'.

Figure 16: external generator with connected with USBLockin

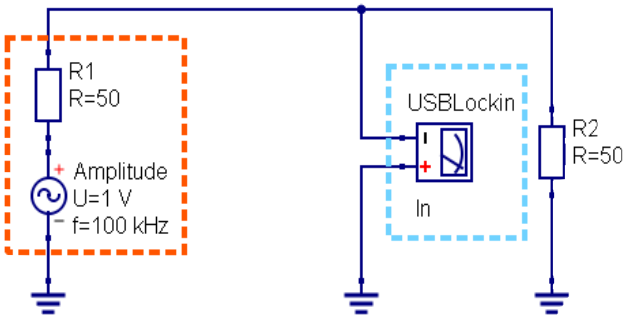
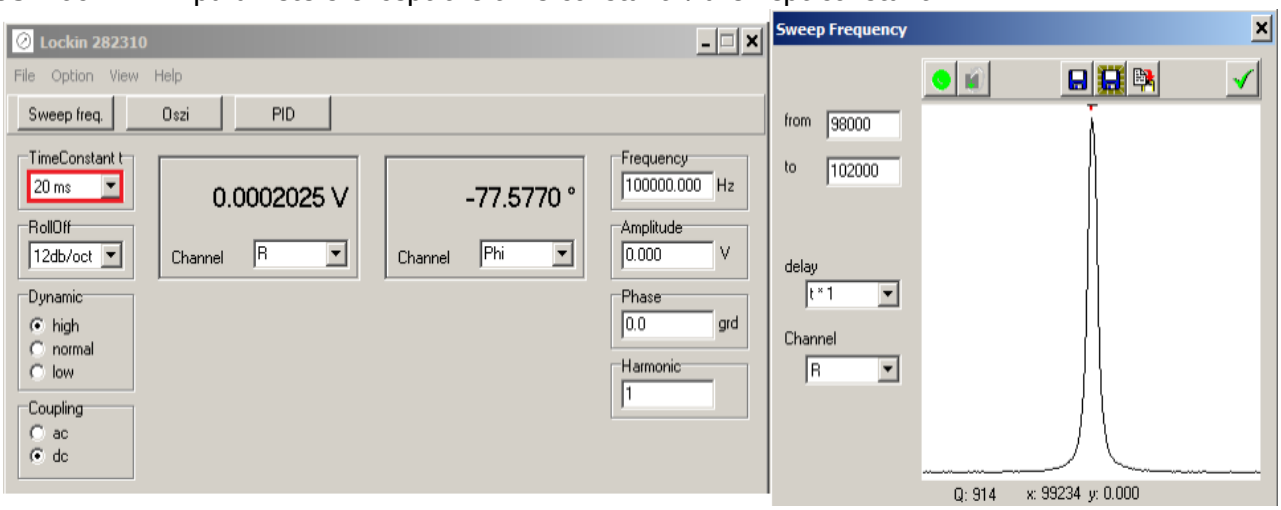


Figure 17: qucs/usb_sinus.sch

In order to demonstrate the effect of the time constant, an external generator is connected to the USBLockin. All parameters except the time constant τ are kept constant.



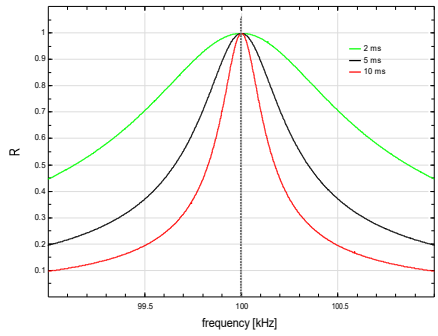


Figure 18: Curve diagram which demonstrates the different time constants

We acquire the spectrum 3 times, each with a different time constant: 2 ms, 5 ms, 10 ms. The spectra are saved or copied to clipboard in order to be visualized in another program. Teh result is shown in Fig. 11.

A more detailed explanation, how this time constant is related to the detection bandwidth of a lockin amplifier, is given in chapter "Time constant Definition".

6.2 Demonstrate RollOff

In order to test the effect of the parameter RollOff on the shape of the pass band detected with a lockin amplifier, an external function generator is connected to the lockin amplifier's input. (see also Figure 23). Let's assume, the function generator provides 100 kHz with 200 µV rms-amplitude. Then, the lockin amplifier will detect 200 µV if it is set to 100 kHz. When the centre frequency is swept from 98 kHz to 102 kHz with a time constant of 5 ms (200 Hz detection bandwidth), the lockin detects a peak shape with 200 µV height and ±200 Hz width (Fig. 13).

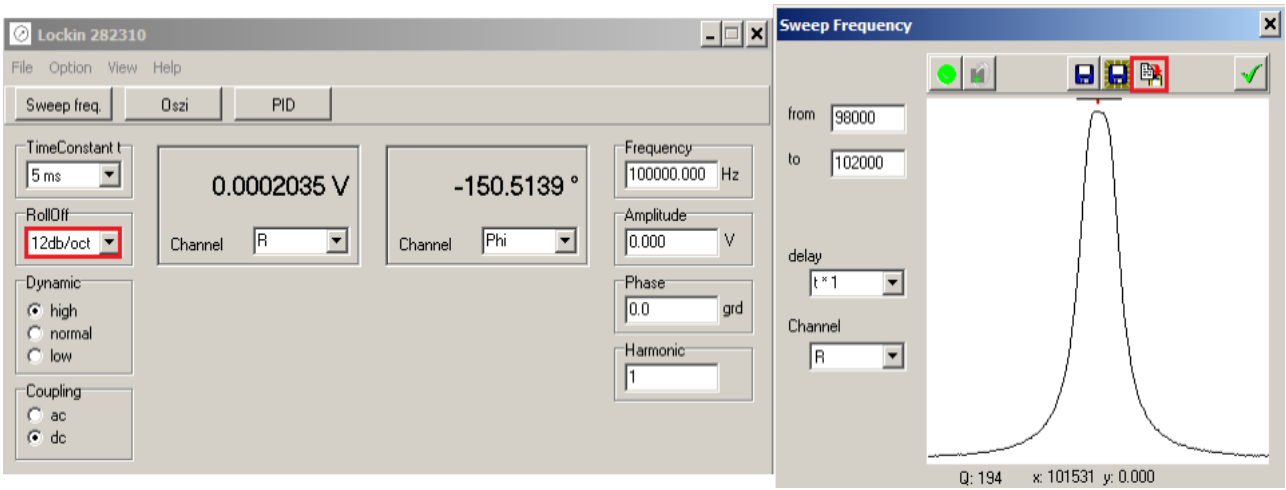


Figure 19: Panel display and frequency sweep result with 200 µVrms and 100 kHz at the input. Time constant: 5 ms

Now, this curve is taken with all available settings of RollOff (Fig. 14). One observes, that all three curves meet in two points that represent the 0.7 times the amplitude of the signal. This shows, that the detection bandwidth is kept independent on the RollOff. At the same time, the slope of the lockin filter increases according to the RollOff settings. On the double-logarithmic scale it is obvious, that 6 dB/oct equal one order of magnitude decay when the frequency is doubled. 12 dB/oct equal 2 orders of magnitude and 24 dB/oct four orders of magnitude, respectively.

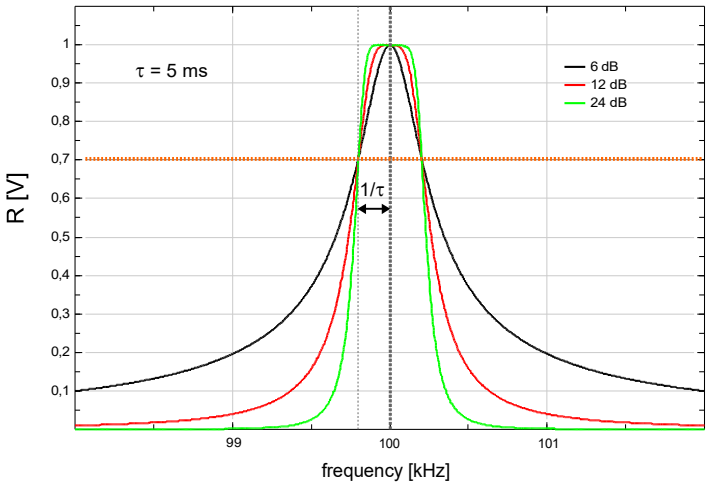


Figure 20: Detection bandwidth shaoe in dependence on the parameter RollOff.

6.3 Resonance Detection

It is very typical to employ lockin amplifiers to detect resonances of external systems. The reference output of the lockin amplifier excites the system, while the frequency dependence of its reaction is analysed plotting R and Phi vs. f.

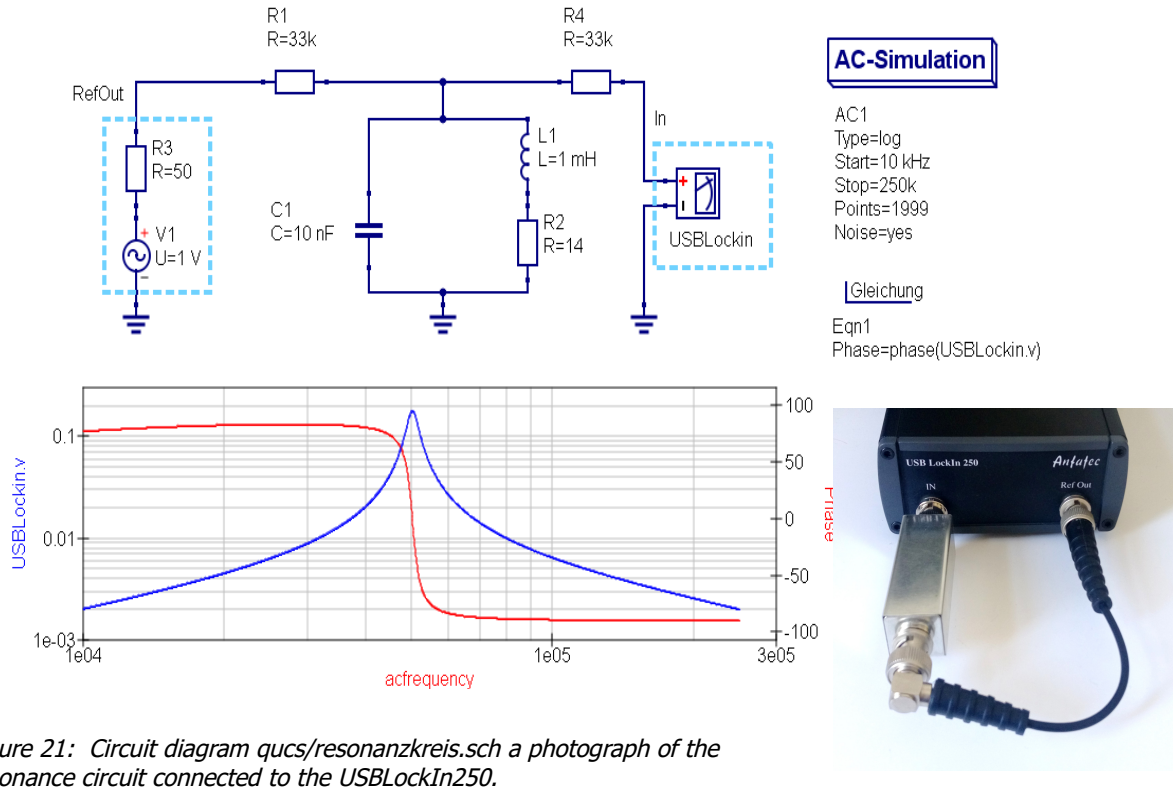
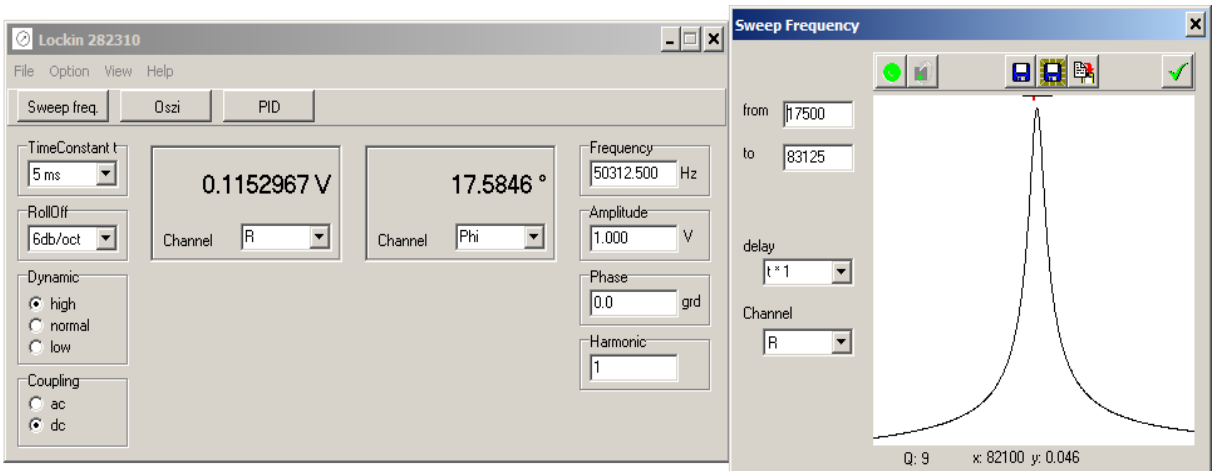


Figure 21: Circuit diagram qucs/resonanzkreis.sch a photograph of the resonance circuit connected to the USBLockIn250.

The application lockin.exe allows to sweep the frequency. The result is shown in these figures:



The circuit answers with 115 mV signal on 1 V excitation. The Q of this analogue circuit is quite low (only 9), so that the width of the detected resonance peak, whose centre frequency is found at 50 kHz, almost reaches 5 kHz.

6.4 THERMAL NOISE

6.4.1 SIMULATION OF THE NOISE VOLTAGE IN QUCS

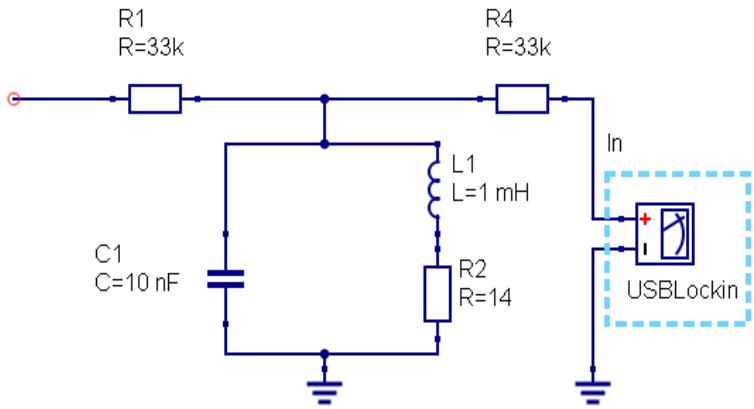


Figure 22: qucs/Noise_voltage.sch

In order to get an idea about the expected noise of the resonance circuit, its voltage noise was simulated by qucs (**Q**uite **U**niversal **C**ircuit **S**imulator) not connecting anything to the input, but analysing the output noise using the AC-simulation feature of qucs.

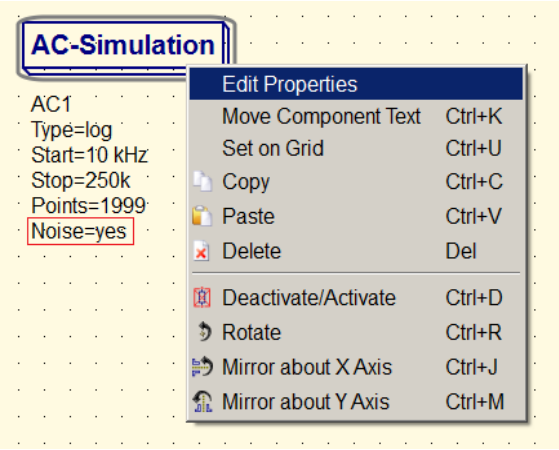


Figure 24: Change simulation properties

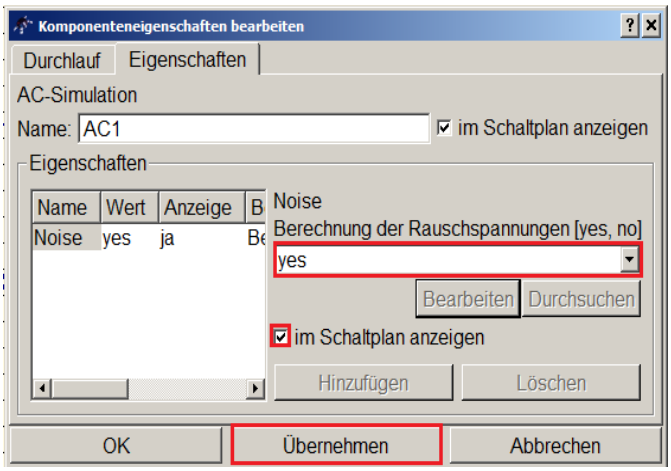


Figure 23: How to calculate noise voltage in qucs

The result of the ac-simulation is then plotted in a diagram and shows the noise voltage (extension .nv) to be higher around the resonance of this circuit:

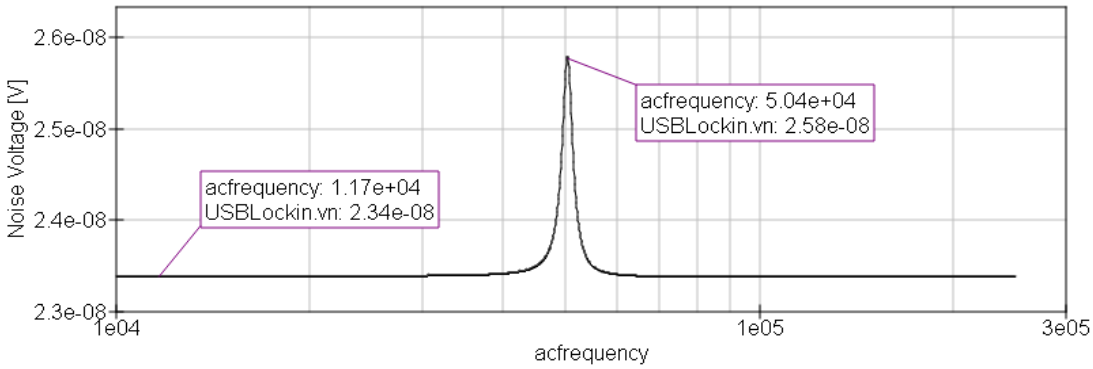


Figure 25: Diagram with noise voltage of the resonance circuit.

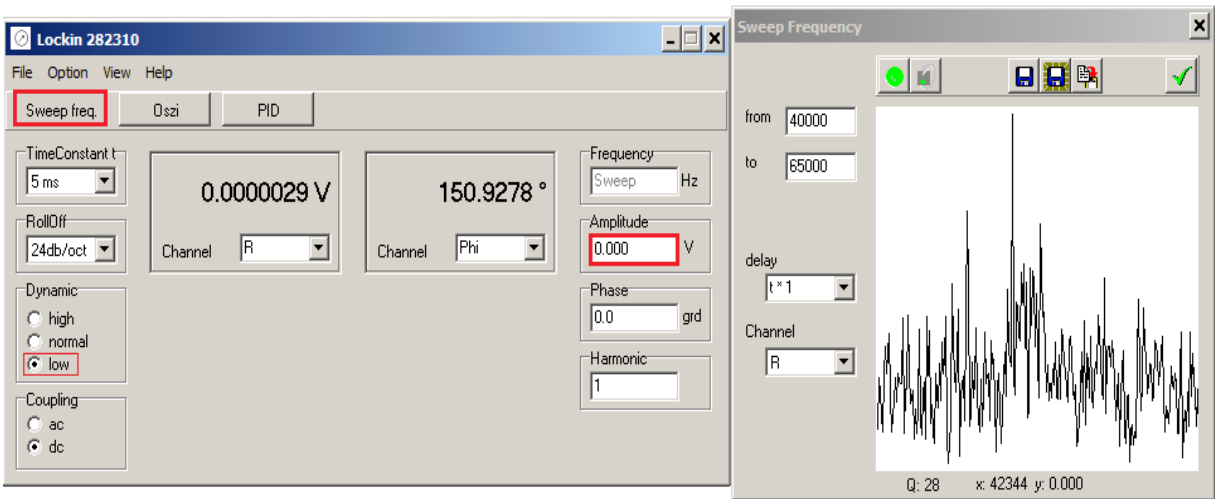
According to this, the peak noise of this resonance circuit should be 25 nV/sqrt(Hz).

6.4.2 Noise Voltage Detection

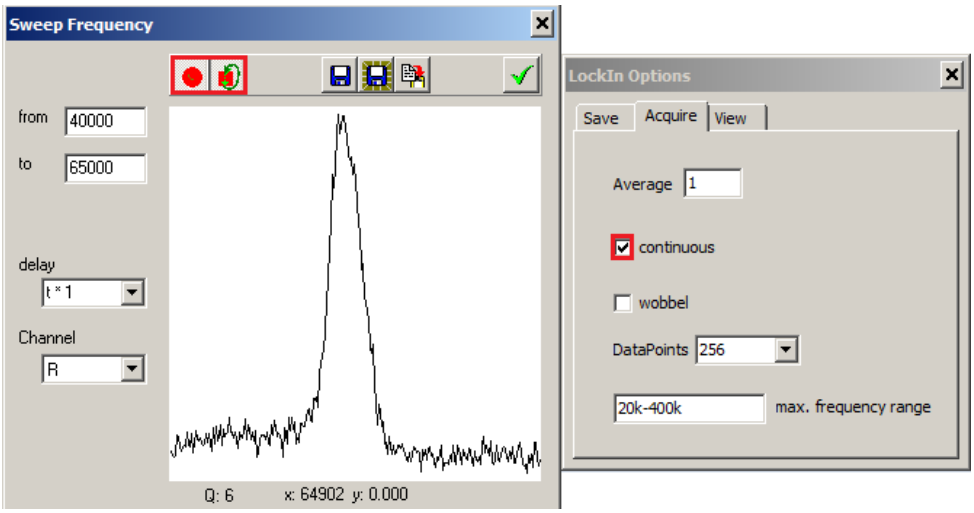
Just as in the simulation, the resonant circuit is now connected to the input of the lockin – itself having an open input (Figure 20). The amplitude of the internal reference of the lockin amplifier should be set to zero for noise measurements. This makes sure that there is not unindented cross-talk between out and input. Also, it is useful to use the input range with the lowest input noise, which is the the "low noise" range, so that the Eigen-Noise of the system has as less as possible effect on the result.

Figure 26: resonance circuit on input of USBLockin

A single frequency sweep over an appropriate range results in the following spectrum:



The software offers averaging of subsequently taken spectra. With this nice feature, the spectral noise density of a circuit can be detected in very high quality. This averaging is enabled in the options window by enabling the check box "continuous" and using the auto-repeat knob in the



spectrum acquisition window. Averaging over 15 minutes results in the curve shown here:

The peak value in this spectrum equals 2 µV. As the spectrum has been taken with 5 ms time

constant, which equals a detection bandwidth of 200 Hz, spectral noise density is 2 µV/sqrt(200 Hz) = 141 nV/sqrt(Hz). This is much higher than the expected peak noise according to the simulations in Chapter 6.4.1 .

Looking closer, one finds that there are several noise components involved in this detection. One component is the resistor noise of R4 that can be calculated with the **"Nyquist-formula":**

$$r_{noise} = \sqrt{4 k_B \cdot T \cdot R \cdot B}$$

k_B	Boltzmann constant 1,38*10^{-23} J/K
T	absolute temperature in K
R	Resistance in Ohm
B	(Δf), detection band width

$$r_n = 0.128 \frac{nV}{\sqrt{Hz \cdot \Omega}} \cdot \sqrt{R} \qquad \text{in } \frac{\text{nV}}{\sqrt{\text{Hz}}}$$

for $R = 33\text{k}$

$$r_n = 0.128 \frac{nV}{\sqrt{Hz \cdot \Omega}} \cdot \sqrt{33\text{k}\Omega} = 23 \frac{\text{nV}}{\sqrt{\text{Hz}}}$$

The total noise, however, also includes the input voltage noise and the input current noise. All these components are independent on each other, so that they do not add up linearly (U1+U2+... +UN), but orthogonally:

$$u_{n\ total} = \sqrt{U_{R1}{}^2 + U_{R2}{}^2 + \ldots + U_{RN}{}^2}$$

CALCULATION OF THE NOISE VOLTAGE

Voltage Noise R4

$$r_n = 0{,}128 \frac{nV}{\sqrt{Hz \cdot \Omega}} \cdot \sqrt{33\text{k}\Omega} = 23 \frac{nV}{\sqrt{Hz}}$$

Current Noise Input USBLockin (OPA ADA 4898)

$$i_n = 2{,}4 \frac{pA}{\sqrt{Hz}}$$

Voltage Noise Input USBLockin

$$u_n = 2{,}0 \frac{nV}{\sqrt{Hz}}$$ → this value can be measured with 50 Ohm at the input.

Total Noise

$$u_{n\ total} = \sqrt{r_n{}^2 + (i_n \cdot R)^2 + u_n^2} = \sqrt{\left(23 \frac{nV}{\sqrt{Hz}}\right)^2 + \left(79{,}2 \frac{nV}{\sqrt{Hz}}\right)^2 + \left(2{,}0 \frac{nV}{\sqrt{Hz}}\right)^2} = 82{,}4 \frac{nV}{\sqrt{Hz}}$$

In conclusion, the current noise of R4 and the resistor noise of R4 contribute much more to the

detected spectral noise density than the resonance circuit itself. Still, the noise enhancement in the resonance can be visualized with the lockin amplifier.

6.5 Fourier analysis of a square signal

Figure 27: USBLockin connected with external generator

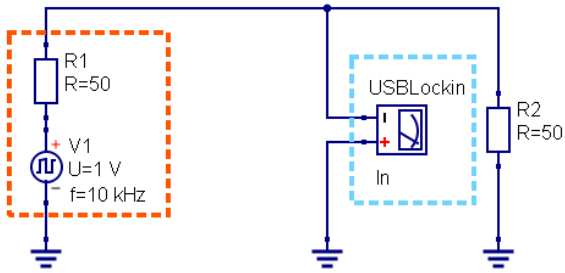
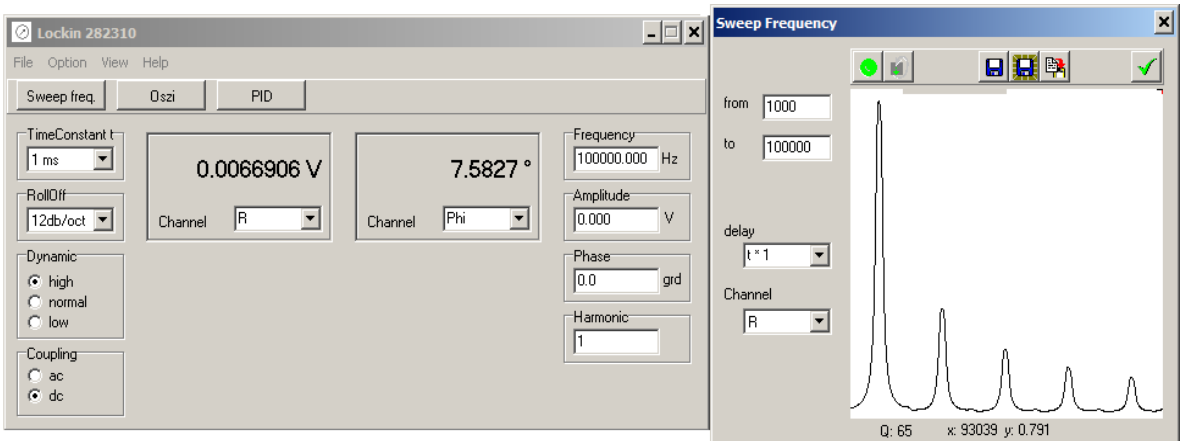


Figure 28: Circuit diagram; qucs/usb_square.sch

The output of a function generator is connected to the input of the lockin amplifier and the signal is terminated with 50 Ohm. The generator provides a square wave form with 10 kHz and 1 V_{rms}. The lockin amplifier divides this wave form into its original Fourier components, which are all odd higher harmonics of the base frequency of the square wave form:



The amplitudes of the single **Fourier components** are the peak amplitudes detected all odd higher harmonics of the generator frequency.

2k - 1 = odd multiples

$$\frac{1}{(2k-1)} \cdot \textit{Amplitude of the } k^{th} \textit{ fourier component of a square signal}$$

Adding all these sine wave with their respective amplitudes up, the square like wave form with a single frequency is regenerated.

$$x(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2\pi(2k-1)ft)}{2k-1}$$

6.6 ACOUSTIC RESONANCE

It is know that different geometrical shapes of bodies react different on acoustic signals.

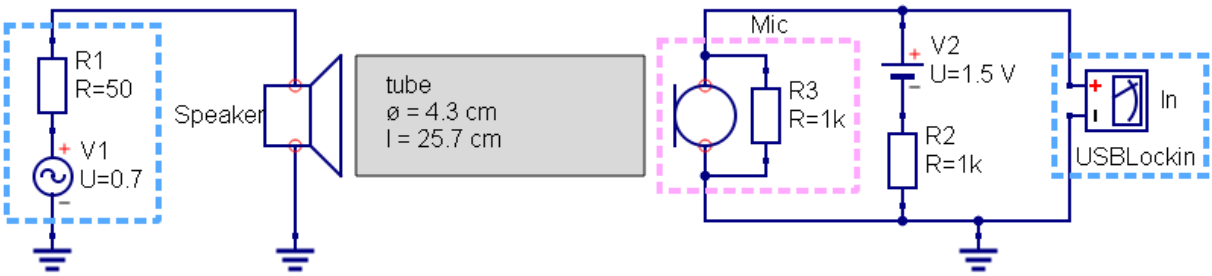


Figure 29: Circuit of acoustic resonance measurement; {qucs/Acoustic_resonance.sch}

In dependence on size and shape, they might amplify certain frequencies that are resonant to standing waves inside these geometries.

A tube, for instance, shows resonances dependent on its diameter and length according to the following formula:

The result of the following measurement was taken from a tube with a length L = 0.257 m and a diameter of d = 0.043 m.

Using the sound speed in air v = 343 m/s, the first 8 resonance frequencies can be calculated as shown in the table below:

$$f = \frac{n\,v}{2(L+0.8\,d)}$$

see Wikipedia

L = 0.257 m
d = 0.043 m
v = 343 m/s speed of sound
n = resonance node

n	calculation	measurement
1	588 Hz	498 Hz
2	1177 Hz	1058 Hz
3	1765 Hz	1645 Hz
4	2353 Hz	2264 Hz
5	2942 Hz	2900 Hz
6	3530 Hz	3536 Hz
7	4118 Hz	4155 Hz
8	4707 Hz	4774 Hz

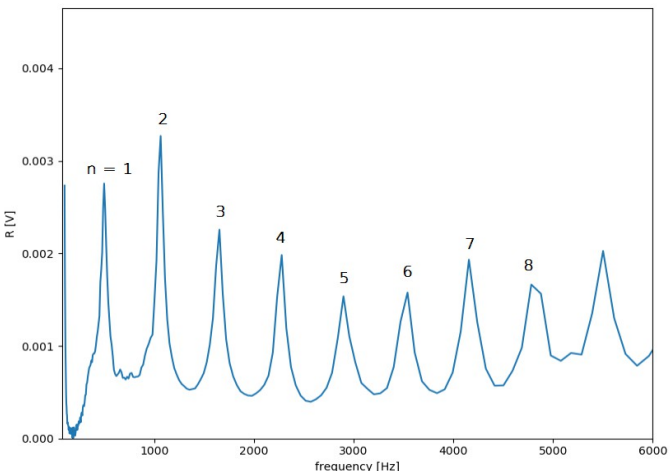


Figure 30: Diagram shows the acoustic resonance by using a tube. Output Amplitude 0.6 V

For the experimental realization, a loud speaker placed at the entrance of the tube was driven by the reference output of the lockin amplifier, while a microphone positioned at the open end detects the amplitude of sound. Now, the frequency of the loud speaker was swept while keeping its amplitude constant. As a result, the transfer function of this tube showing the acoustical signal to be amplified at the resonance frequencies of this tube can be detected.

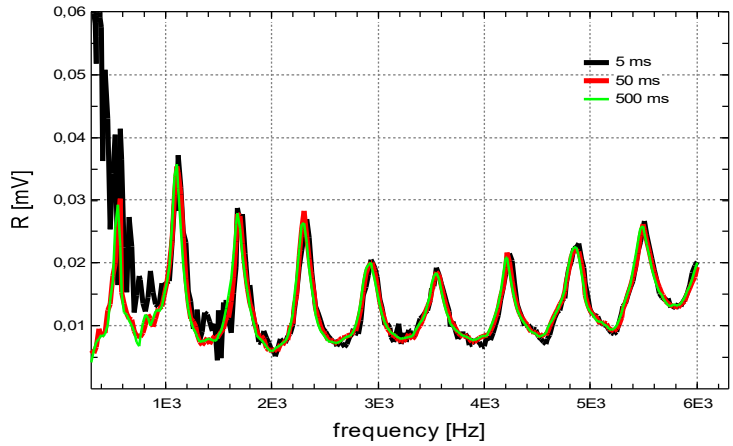
The maxima of this frequency spectrum can be compared with the calculated maxima finding a nice agreement between experiment and theory.

Variation of time constant :

τ = 5 ms, 50 ms, 500 ms
U_{ac} = 10 mV

RollOff = 12 dB
Harmonic = 1
Phase = 0°
Coupling: ac

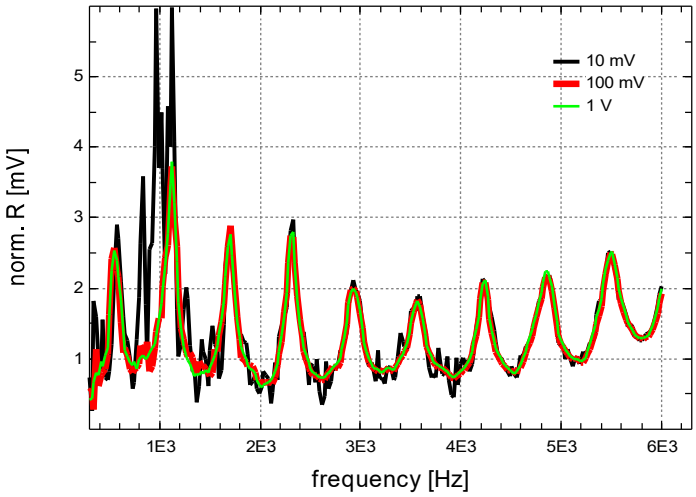
The shorter the time constant, the higher is the noise of the detected signal.



Variation of Amplitude

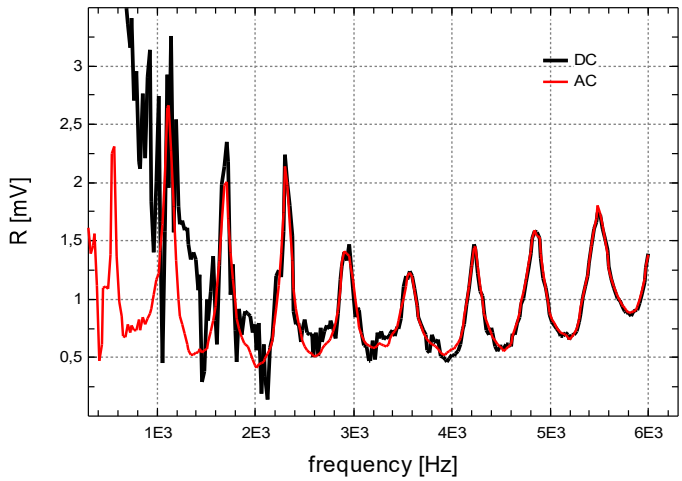
τ = 10 ms
U_{ac} = 10 mV, 100 mV, 1 V

Smaller excitations cause smaller signals, so that the effect of intrinsic noise increases.



AC/DC

τ = 20 ms
In addition to this, the effect of an dc offset on the input signal can be discussed at this example.
The microphone delivers a signal with a certain DC offset. With short time constants, this signal at f = 0 Hz is not suppressed enough and thus one finds a strong signal increase towards small frequencies.
AC coupling, as shown in the other measurements suppresses this effect.



7 DLL DESCRIPTION

The USBLockIn250 can be addressed from other programs and LabViewTM by calling functions provided in a DLL. In order to make use of these functions as transparent as possible, the following files are provided:

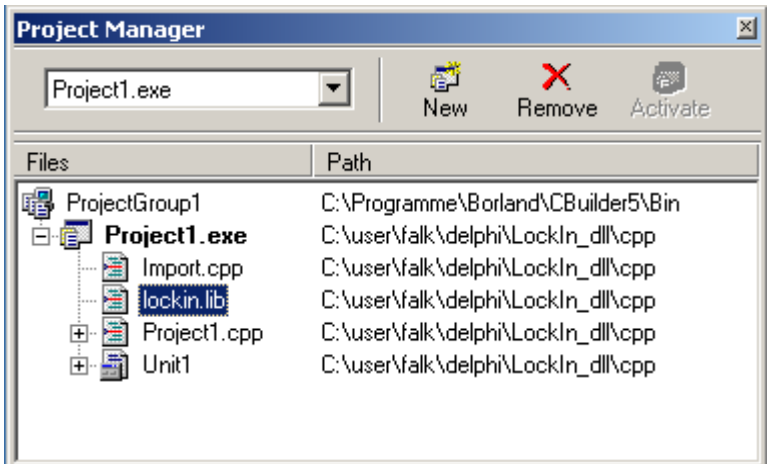
- source code of the DLL
- example in Borland C++ Builder 5.0 for using the DLL
- example in Delphi for using the DLL
- current versions of the DLL

This DLL is compatible with C calling conventions. The convention of the functions in C++ need an underline in front of function names.
Locations of the different DLL file version are described in Section Installation.

7.1 WORKING WITH THE DLL IN C++ PROJECTS

Add the lib-file generated from the DLL to the project in Borland C++ Builder 5.0 by opening 'Project / Add to Project' (see figure).

Chose the **lockin.lib** as the file to be added.



If the lockin.lib does not exist, it can be created from the command line:

- open the command prompt by running 'cmd' in 'Start/Run..' under the Windows Start Menu.
- Go to the directory in which the lockin.dll is located (e.g. 'C:\Windows')
- type: 'implib lockin lockin.dll'

  The lockin.lib is created in the same directory based on the lockin.dll.

To call functions residing in the DLL, headers are necessary, so add 'import.cpp' also to the project, which contains:

```
      extern "C"{

double      _declspec(dllimport) cdecl SetLockInFreq  (double freq);
void        _declspec(dllimport) cdecl SetLockInAmpl  (double Ampl);
double      _declspec(dllimport) cdecl SetLockInPhase (double Phase);

long        _declspec(dllimport) cdecl SetLockInHarm  (long Harm);long
            _declspec(dllimport) cdecl SetLockInRollOff (long Time);
```

```
Sdouble     _declspec(dllimport) cdecl GetLockInChannel (long Channel);
long        _declspec(dllimport) cdecl GetLockInStatus;
void        _declspec(dllimport) cdecl FillDataArrays (long *length,
            double *XData[], double *YData[], double *Rdata[], double
              *PhiData[]);
double      _declspec(dllimport) cdecl SetLockInTimeBase (double
              TimeBase);
long        _declspec(dllimport) cdecl IsUSBLockin;
long        _declspec(dllimport) cdecl SetLockInSync;
void        _declspec(dllimport) cdecl SetLockInAUX (long *Channel, long
              *Input, double *MaxRange);
char*       _declspec(dllimport) cdecl GetVersion;
long        _declspec(dllimport) cdecl SetUsedDevice (char *device);
long        _declspec(dllimport) cdecl SetLockInPID (long *Channel, long
              *Input, double *Offset, double *Kp, double *Ti, double *Td)
}
```

All these functions give the set value back when they are called, except 'SetLockInAmpl' and 'FillDataArrays'. 'SetLockInFreq' gives the set value back, when the system uses the internal oscillator (PLL = Off) or allows the user to read the external frequency, when then PLL is ON. 'SetLockInTimeBase' gives back the IdleCount which is calculated from the given TimeBase and PointsPerDivision. If the setting of the current device succeeded is given back by 'SetUsedDevice' and 'GetVersion' gives back a string that includes the hardware number of the lockin and the versions of sys- and dll-files.

7.2 DLL-FUNCTIONS

SetLockInFreq

```
function SetLockInFreq (freq : double) : double;
```

this function sets the centre frequency of the lockin to a value in Hz and returns the value of the frequency in Hz. If the PLL is OFF, the return value equals the sent value. If the PLL is ON, this function can be used to read the external frequency value.

SetLockInAmpl

```
procedure SetLockInAmpl (Ampl : double);
```

Sets the output amplitude of the reference output in V and gives no value back.

SetLockInPhase

```
function SetLockInPhase (Phase : double) : double;
```

sets the phase offset in degree and returns this phase offset in degree.

SetLockInHarm

```
function SetLockInHarm  (Harmonic : integer) : integer;
```

sets the harmonic at which the signal is analysed. The harmonic can vary between 1 and 15. It returns the harmonic value.

SetLockInPllOn

```
function SetLockInPllOn (PllOn : integer) : integer;
```

switches the PLL on or off (0) and gives the state of the PLL back.

PllOn = 0	PLL is off, internal reference is used
PllOn = 1	PLL is ON and on TTL input level
PllOn = 2	PLL is ON and sine wave input is used (not valid for AMU2.4 PCI card)

When using the external trigger (PLL is ON), please make sure, that later calls of SetLockInHarm(H) or SetLockInPhase do not reset the state of the PLL to zero.

Starting from software package Version 1.05, the functions "SetLockInPhase,", "SetLockInHarm","SetLockInFreq" and "SetLockInAmpl" do not overwrite the PLL setting anymore.

SetLockInTimeConst

```
function SetLockInTimeConst (TimeConst : integer) : integer;
```

sets the time constant for the low pass filter.

TimeConst =

0	10 µs
1	20 µs
2	50 µs
	0.1 ms
4	0.2 ms
....	
17	5 s

SetLockInRollOff

```
function SetLockInRollOff (RollOff : integer) : integer;
```

sets the RollOff for the low pass filter:

0	6 dB/oct
1	12 dB/oct
2	24 dB/oct

GetLockInChannel

```
function  GetLockInChannel (Channel : integer) : double;
```

reads the results in the channel number x, where x is:

0	real part = X
1	imaginary part = Y
2	amplitude = R
3	phase = Phi
4	In

It returns the read value of this channel in V.
Note: It is necessary to call **SetLockInHardGain** first.

GetLockInStatus

```
function  GetLockInStatus : longint;
```

Returns a value where bit 1 (corresponding to 2^1) is set, if the input of the USBLockIn250 is in overload state. If bit 2 (corresponding to 2^2) is set, the device driver has stopped and the hardware was removed.

SetLockInHardGain

```
function  SetLockInHardGain (Gain : integer) : integer;
```

Switches the input gain of the lockin between

1 high reserve

10 normal

100 low noise

NOTE:
This function "SetLockInHardGain" should be called at least once at the beginning of a program. Otherwise, the scaling of the R, X, and Y as well as of the outputs Aux1 and Aux2 might be wrong.

SetLockInCoupling

```
function  SetLockInCoupling (Ac : integer) : integer;
```

Switches the input coupling of the lockin between ac mode and dc mode

1 ac coupling

0 dc coupling

FillDataArrays

```
procedure FillDataArrays (pLength: PInteger; XData, YData, RData,
PhiData: PDArray);
```

Fills the given arrays with the corresponding X-, Y- and R-Data. The length of all four arrays must be 1024 (0...1023). LabView should reserve the memory and provide the pointer to these arrays for the DLL. The parameter *"lLength"* is to the number of indexes of the array that were actually filled.

SetLockInTimeBase

```
function SetLockInTimeBase(timeBase: Double): Double;
```

the input to this function "*timeBase*" is the time between two data points in seconds. The function returns the used time between data points in seconds.

IsLockInUSB

```
function IsLockInUSB: Longint;
```

This function returns "1", if the connected device is a USB-LockIn250 and "0" if it is not.

SetLockInSync

```
function SetLockInSync (Sync: Longint): Longint;
```

This function allows to set the number of sync cycles for synchron filtering mode and returns the value set.

SetLockInAux

```
procedure SetLockInAUX(Channel, Input: Longint; MaxRange : Double);
```

The USBLockIn250 has two analogue auxiliary outputs named with "Aux1" and "Aux2" – see also page 15. The parameter *"channel"* selects which connector is used (0 ↔ Aux1, 1 ↔ Aux2). The parameter *"input"* selects the type of signal that is provided at this output (In ↔ 0 , X ↔ 10, Y ↔ 11, R ↔ 12, Phi ↔13). "MaxRange" scales the signals according to the amplitude at the selected input channel. MaxRange of Phi is always 180° while for the other channels it can be any value between -10 V and 10 V.

```
Examples:
SetLockInAux(1,10,0.01) →  provides the measured value of X at the output
AUX2 scaled to 10 mV.
SetLockInAux(0,1,2.35) →  Aux1 is set to 2.35 V.
```

If the parameter "input" is set to 1, then the "MaxRange" parameter is used as output voltage.

```
Examples:
SetLockInAux(1,1,5.32) →  sets the output of Aux2 to 5.32 V
```

SetUsedDevice

```
function SetUsedDevice (FullName: PChar) : longbool;
```

When multiple USBLockIn250 are connected to the PC, each lockin amplifier can be selected by its ID. This command allows to enter a specific device information so that this multiple lockins can be addressed independently. (see also page 51)

SetUsedDevicePerIndex

```
function SetUsedDevicePerIndex (Index : integer);
```

When multiple USBLockIn250 are connected to the PC, each lockin can be selected by Index. (only supported on USB LockIn)

GetVersion

```
function GetVersion : PChar;
```

Returns the hardware serial number (e.g. 280007), the version number of the driver (e.g. 0.07.0.3) and the version of the dll itself (e.g. 1.0.6.2).

Example result

```
HW SerNr. :0
sys: 0.07.0.3
dll: 1.0.6.2
```

When the DLL is initialized, it sets the following default values:

Phase	0 degree
Harmonic:	1
PLL.:	off
Time constant:	10 ms
RollOff:	12 dB
Input Gain:	1 (high reserve)

SetLockInPID

```
function SetLockInPID (Channel, Input : LongInt; Offset, Kp, Ti, Td:
Double):boolean;
```

This functions sets the parameters for PID control.

The parameter *"channel"* selects which connector is used (0 ↔ Aux1, 1 ↔ Aux2).

The parameter *"input"* selects the type of signal that is used as input for the PID control (In ↔ 0 , X ↔ 10, Y ↔ 11, R ↔ 12, Phi ↔13).

The parameter *"Offset"* equals the set-point. *"Kp"* , *"Ti"* and *"Td"* are the feedback parameters

(see page 15).

SetLockInInputConf (only valid for devices with differential input)

```
function SetLockInInputConf (conf: Longint): Longint;
```

This functions switches between the different input modes.

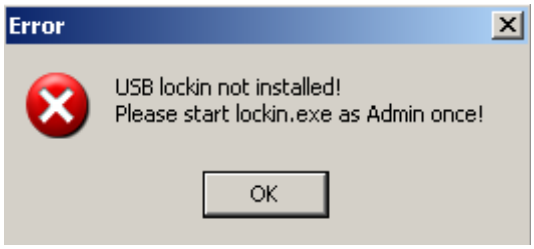
The parameter *"conf"* selects which connector is used (0 ↔ A , 1 ↔ A-B).

7.3 EXAMPLE PROGRAMS

Currently, there is a program examples provided in Borland Delphi 6.0 (demo.exe).
It can be found in the directory %YourPath/DLL/Delphi

8 REMOTE CONTROL WITH LABVIEW

Note: LabView and DLL-functions will not run properly, when the test software "lockin.exe" never has been started as Administrator. It requires properly set entries in the registry for the right scaling. When LabView is started before the entries in the registry exist, the following error message appears:



8.1 Files and Locations

The LabView driver is based on the provided DLL file "lockin.dll". The 'lockin.dll' is best located in the same directory as the vi-files or in any path that is known to LabView as library path. You have to use the DLL in the same Version like LabView (32 or 64 Bit).

Anfatec has prepared LabView examples for LabView6.1, LabView11 and saved some special applications also for Labview9.

The following vi-files are provided for LabView version 11:

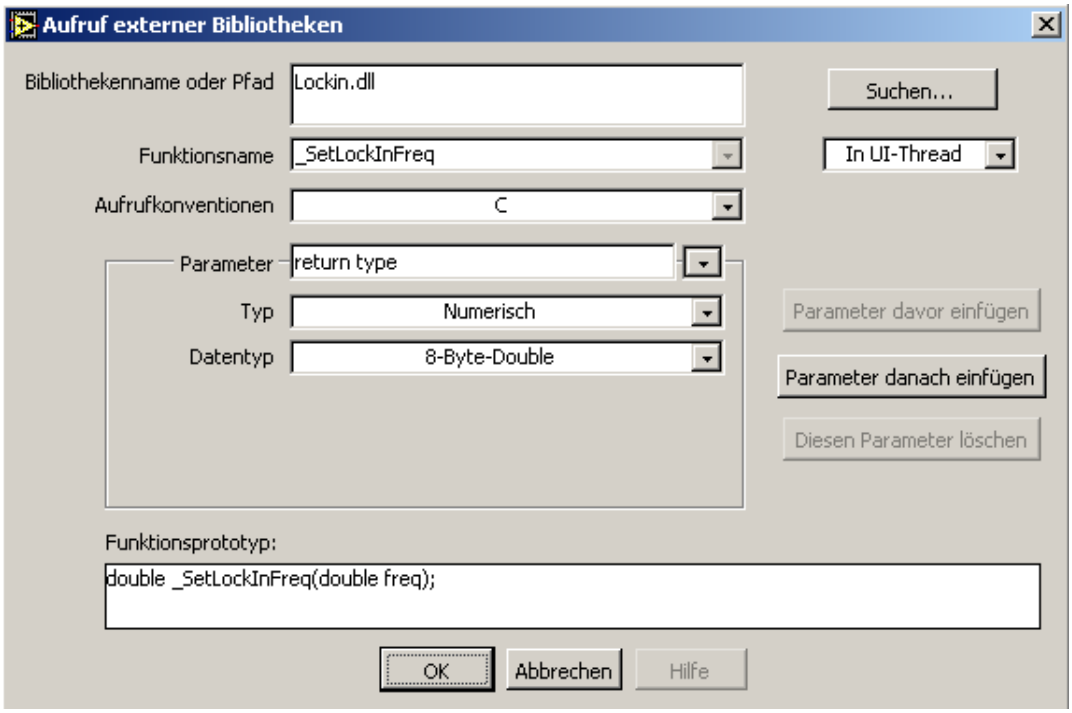
Output_Lockin.vi	read the four output channels X, Y, R and Phi
Input_Lockin.vi	sets the input and internal parameters of the USB-LockIn250
Named_Input_Lockin.vi	sets the input and internal parameters of the lockin specified by a device string (only useful if more than one USB-LockIn250 is used at the same time)
LIA_sweep_frequency.vi	acquires a spectrum of R
SetAUX_Simple.vi	shows how the AUX output channel reacts to different maximum ranges (Only USBLockin)
AUXRamp_Lockin.vi	demonstrates how one can create a ramp at the AUX 1 output channel (Only USBLockin)
SetAUX_Lockin.vi	has the same functionality as the "lockin.exe" (Only USBLockin)
GetUnitAndRangeFromString.vi	divides a supplied string into physical unit and range
GetTimeConstantValue.vi	translates the time constant index into a string referring to the real time constant in s or ms
Oszi_LockIn.vi	shows X, Y, and R in one oscilloscope screen
Named_Input_LockIn.vi	shows how to address several lockin amplifiers in parallel be calling them their names

Additionally, in LabView 9, a special application for a phase feedback called "Nose" has been programmed. (see page 49).

8.2 General Programming Directions in LabView

Switch into the diagram view of the vi-files. The parameters are set and read with the DLL functions or double click into this functions to open the window, which allows to correlate the function with a DLL-function and its exchange value:

The used library is the 'lockin.dll'. The example sets the frequency with the function 'SetLockInFreq'. The value is of numerical type and 8-Byte Double long.



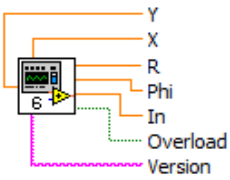
Needed data types in the DLL and their equivalent in LabView are:

- double numerical type, 8-Byte-Double
- integer numerical type, signed 32-Bit-Integer
- longint numerical type, signed 32-Bit-Integer
- PInteger numerical type, signed 32-Bit-Integer (Pointer to value)
- PDArray array type, 8-Byte-Double (1 dimension, array pointer)
- longbool numerical type, signed 32-Bit-Integer
- PChar string type, pointer to C-String

The frequency sweep example, for instance, consists of a loop, in which each single frequency value in the spectrum is calculated based on the input parameters 'Start Frequency', 'Stop Frequency' and 'Number of Steps', while the results are stored and displayed in a graphical window. It includes the sub-vi's 'Input_Lockin' and 'GetTimeConstantValue.vi'. They only connect the numerical data and DLL function.

8.3 Description of Example VI Files

The file **Ouput_Lockin.vi** reads the channels 0 to 4. Channel 0 equals X, channel 1 equals Y. Channel 2, 3 and 4 are R, Phi and In, respectively. The numbers given here are in the unit shown behind the numbers.



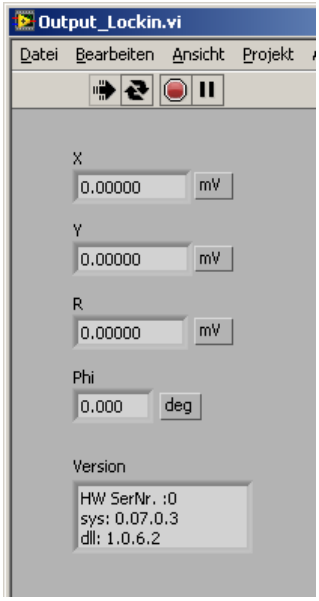
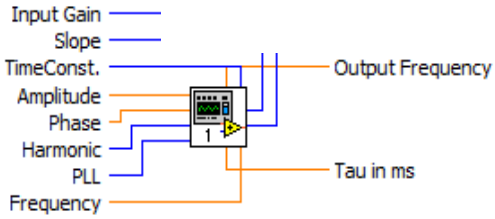


Figure 31: Frontpanel of the Output_Lockin.vi

The file **Input_Lockin.vi** allows to set the main parameters of the lockin. Input Gain, Time Constant (and the time Constant in ms) and RollOff are given as integers. The relation between the value and the integer is described together with the DLL. Frequency, Harmonic, Amplitude and Phase are of the numeric type 'Double'. They are used unit-less in this file, but their units are meant in Hz, Volts and Degree, respectively.
Beside the frequency input, the current frequency is displayed. Also, an additional switch allows to enable the PLL function. When the PLL is ON, this frequency display shows the external frequency, onto which the system is locked. If the PLL is OFF, it shows the same frequency that the user has set in the frequency input.



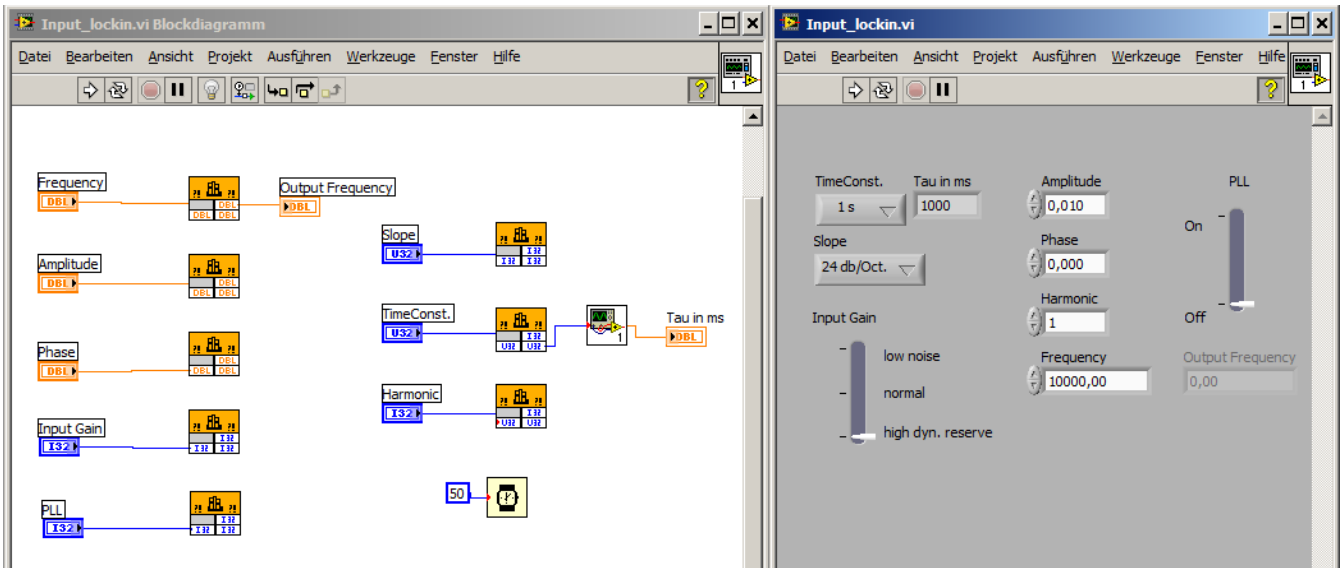
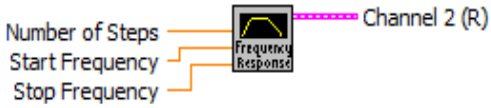


Figure 32: Block-diagram and Frontpanel of the Input_Lockin.vi

For the use of **Named_Input_LockIn.vi** please read the chapter "8.48.4 Multiple LockIn Amplifiers in LabView" on page 51.

The **LIA_sweep_Frequency.vi** is one possible example which shows how spectra might be acquired with the USB-Lockin in LabView. The user provides the corner data for the spectra acquisition (Start Frequency, Stop Frequency, Number of Values). Also the user can change parameters like time constant, slope, harmonic and amplitude. For change these, the sub-vi "Input_Lockin.vi" ılled.



When the VI-file runs, one spectrum is acquired. During the spectra acquisition, the current frequency is monitored. To get the delay what is needed, the sub-vi "GetTimeConstantValue.vi" is called and returns the time constant in ms.

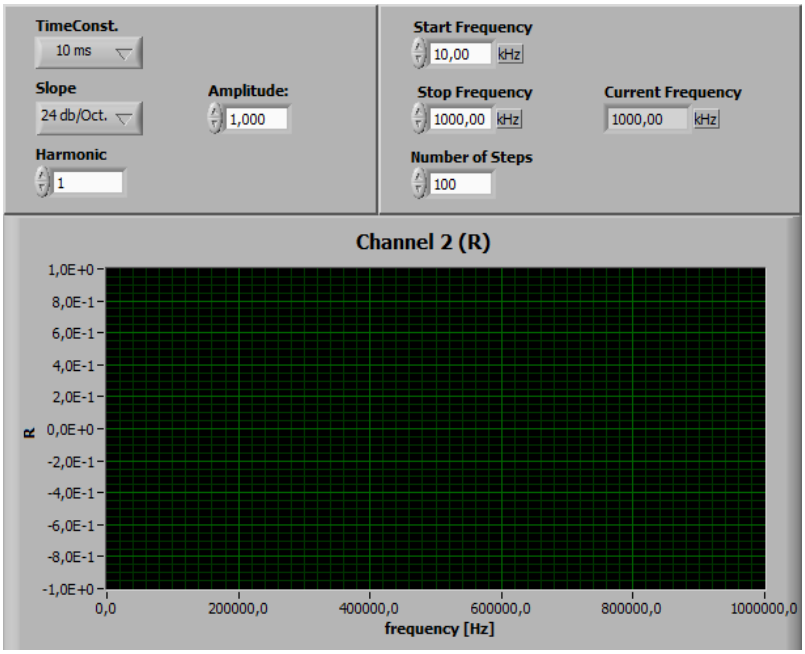


Figure 33: Fontpanel of the LIA_sweep_frequency.vi

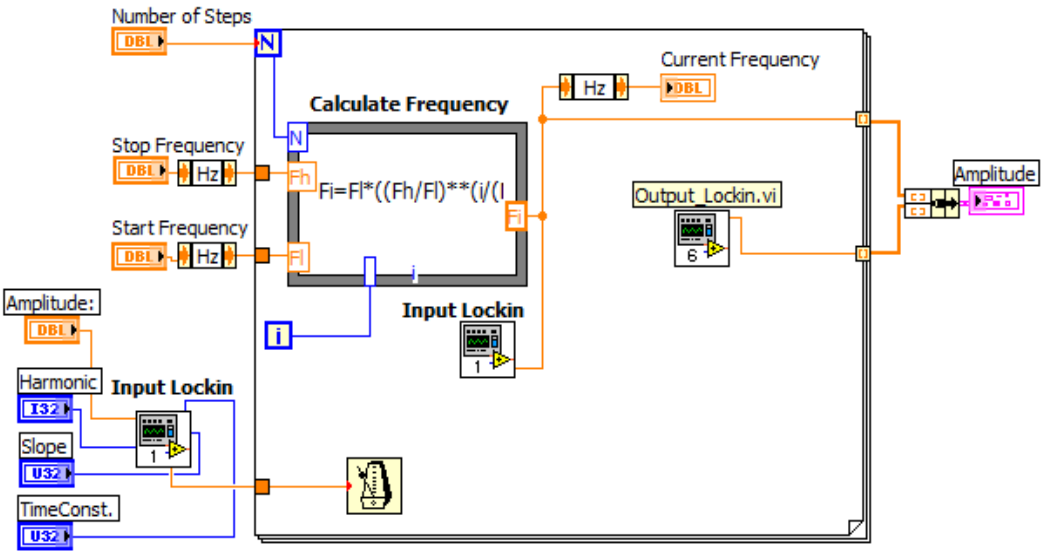


Figure 34: Block-diagram of the LIA_sweep_frequency.vi

Example Time Constant

See also **time constant**

In this example the time constant is set to 1 ms, resulting in $f_B = 1/T = 1$ kHz. In a frequency spectrum (see Fig. 41), f_B defines the offset from the center frequency at which an effective amplitude of 70,7 % of the peak voltage (here ≈ 0,7 V) is detected.

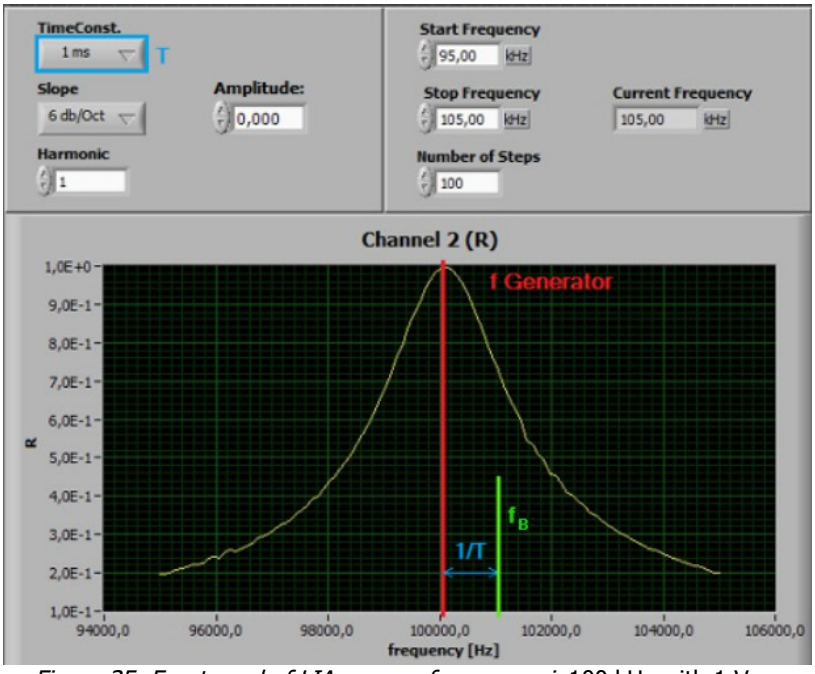


Figure 35: Frontpanel of LIA_sweep_frequency.vi. 100 kHz with 1 Vrms were supplied to the input of the lockin amplifier, while τ was set to 1 ms.

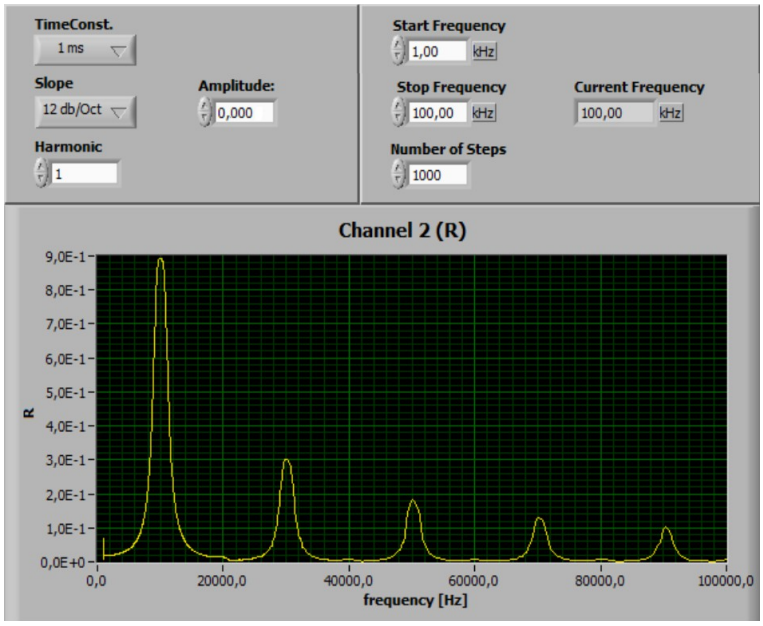
Example
Fourier Analysis of a square signal

(See also Fourieranalyse of a square signal**)**

For the measurement shown in Fig. 38, the input of the lockin amplifier is supplied with a 10 kHz square wave (containing all odd higher harmonics of 10 kHz) while the internal reference of the lockin is swept from 1 kHz to 100 kHz.

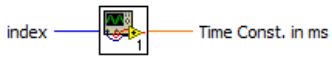
Fig. 38 shows the resulting pattern akin to the Fourier transformation of the signal. The width of the peaks is defined by the bandwidth set with 1 ms time constant.

Figure 36: Frontpanel of the LIA_sweep_frequency.vi



GetTimeConstantValue.vi

this converts the index (which is sent to the lockin amplifier to set the time constant) into a a name that allows the user to visualize the time



constant in s, ms or µs. This vi is used inside 'LIA_sweep_frequency.vi', for example.

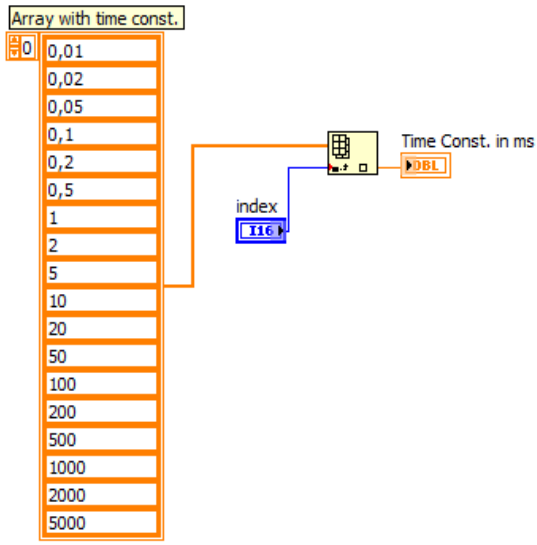


Figure 37: Block-diagram of the GetTimeConstantValue.vi

SetAUX_Simple.vi (Only USBLockin) uses the DLL-function "SetLockInAUX" to set the AUX0 channel to the selected maximum range. Therefore, it uses the SubVI "GetUnitAndRangeFromString.vi" to get MaxRange in Volts from the string provided by the "Display Range"-Combobox (e.g. "0.01" from "10 mV"). With an Output Amplitude of 10 mV, the AUX0 output at a display range of 20 mV should be 5 Volts. For channel number 13 (Phi) the maximum range is always 180 degrees. To get different AUX outputs here, use the Phase offset in the LabView-Frontpanel

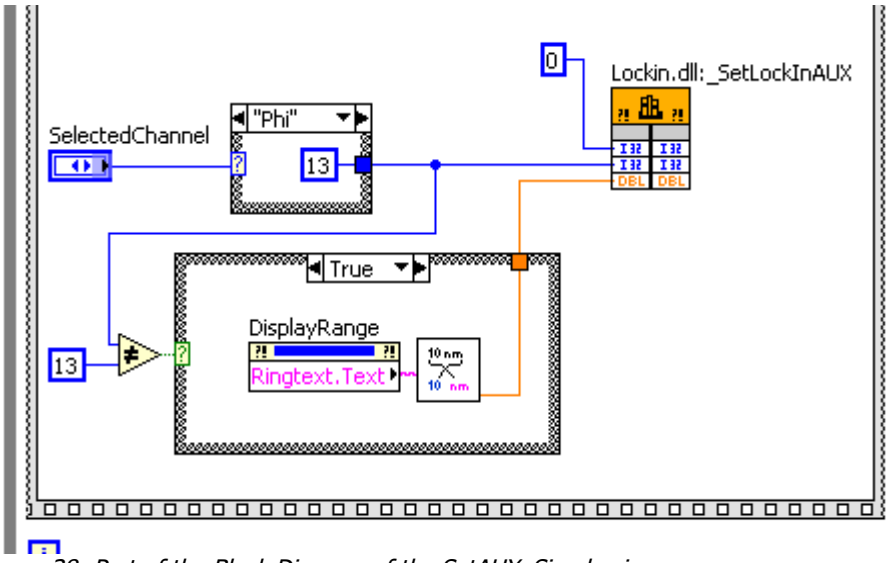


Figure 38: Part of the Block Diagram of the SetAUX_Simple.vi

AUXRamp_Lockin.vi (Only USBLockin) uses output channel 1 to create a ramp at AUX channel 0 & 1. The boundaries are given with "from" and "to". Taking the requested amount of "steps" into account, a formula calculates the step size and also returns a value "current AUX" displayed in Volts.

Example Diode (see **Diode AUX**):

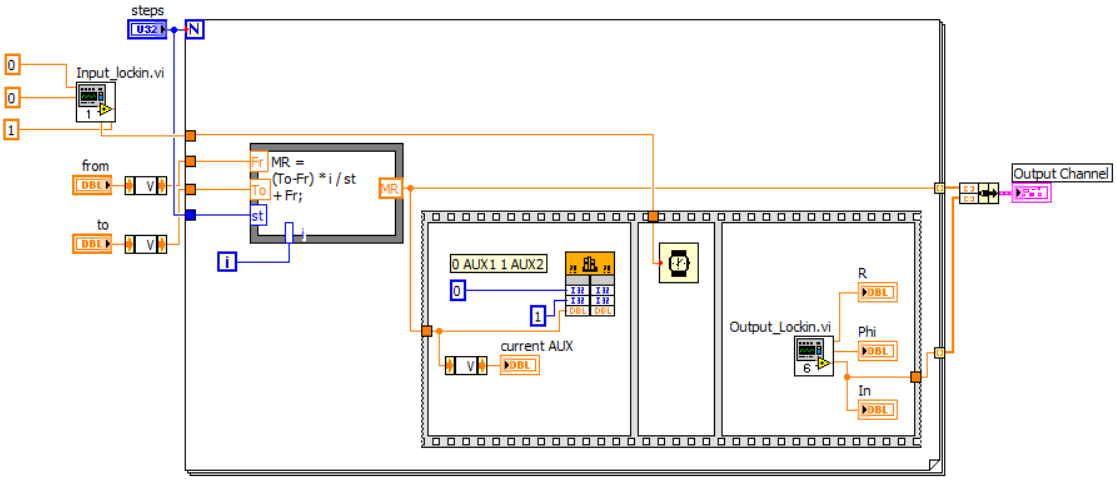


Figure 39: Block-diagram of the AUXRamp_Lockin.vi

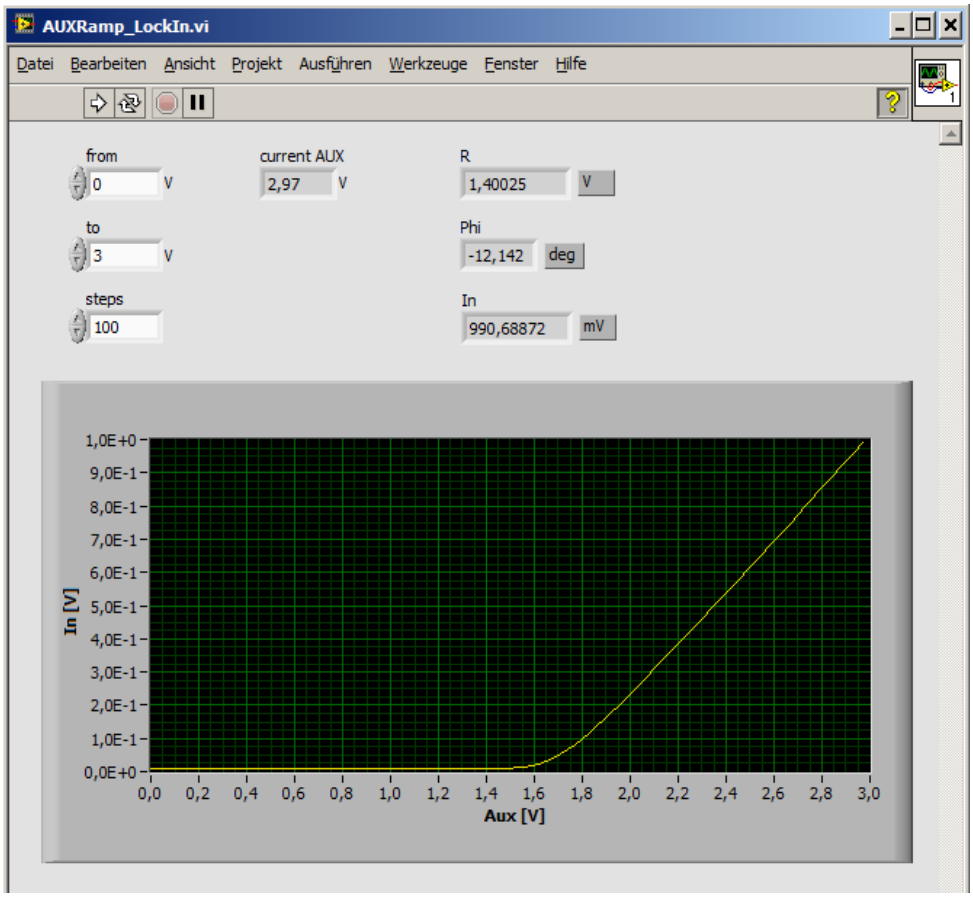


Figure 40: Frontpanel of the AUXRamp_Lockin.vi.

The Aux-output is connected to a diode and swept from 0 V to 3 V, while the DC current through the diode is detected as volatge drop over a resistor and recorded as DC signal "In" with the input of the lockin amplifier.

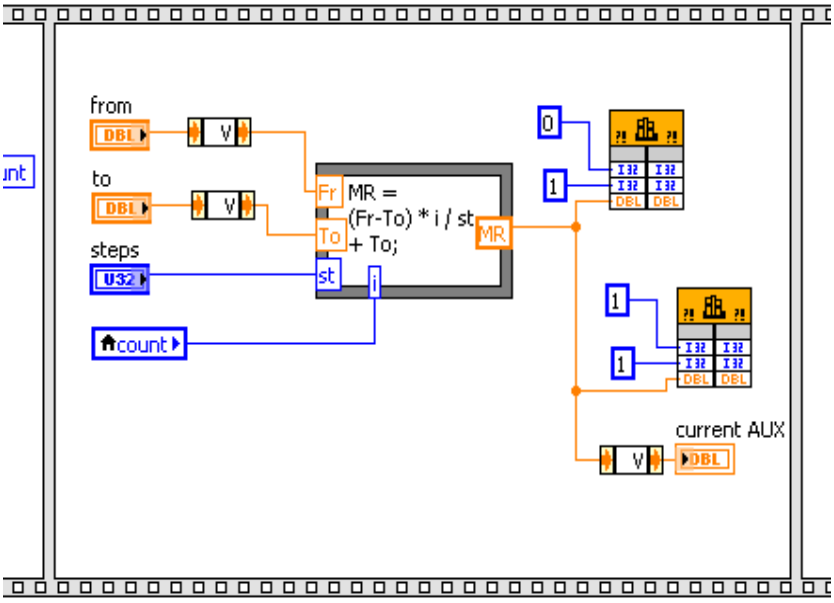
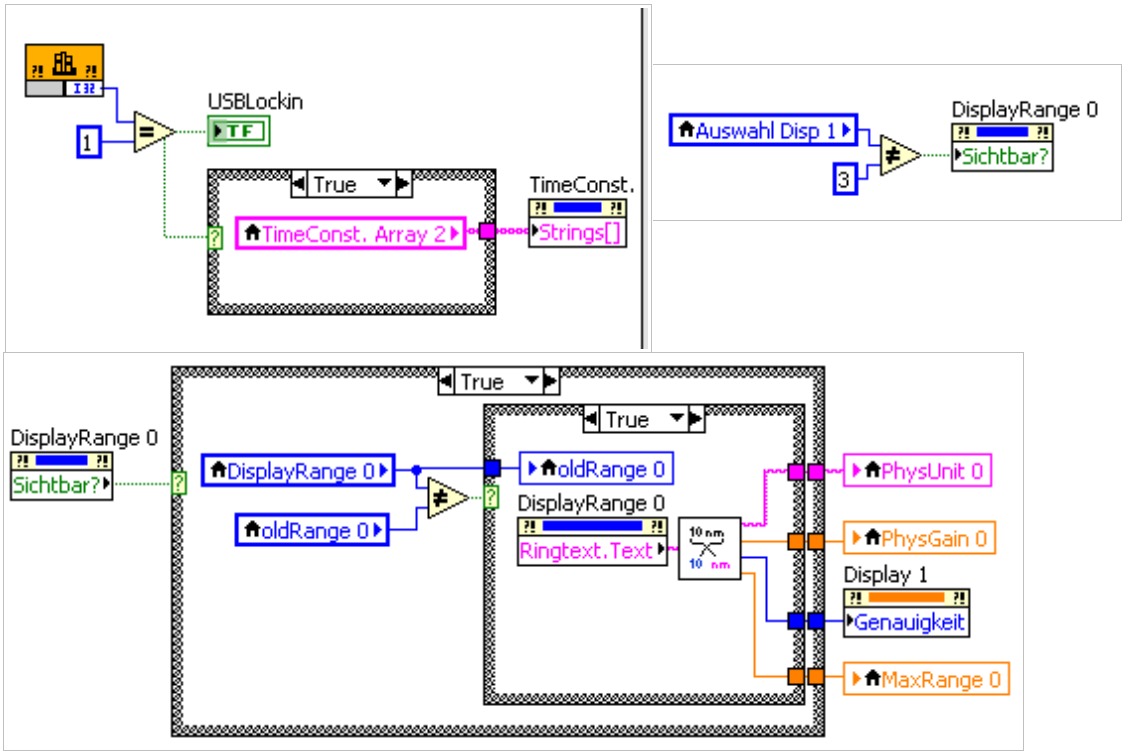


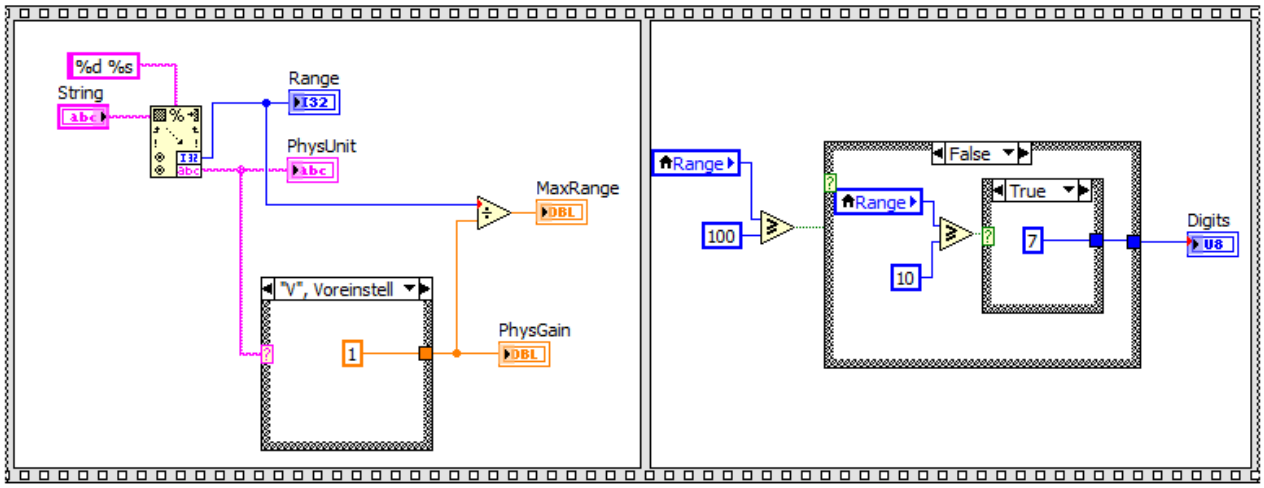
Figure 41: Part of the Block Diagram of the AUXRamp_Lockin.vi

A more advanced example how to use the USBLockIn250 in LabView is the "**SetAUX_Lockin.vi**" (Only USBLockin). It has almost the same functionality like the "lockin.exe". It fills the "TimeConst." Combobox with different arrays, in dependence whether a USB-LockIn250 or a PCI-LockIn is connected to the PC.

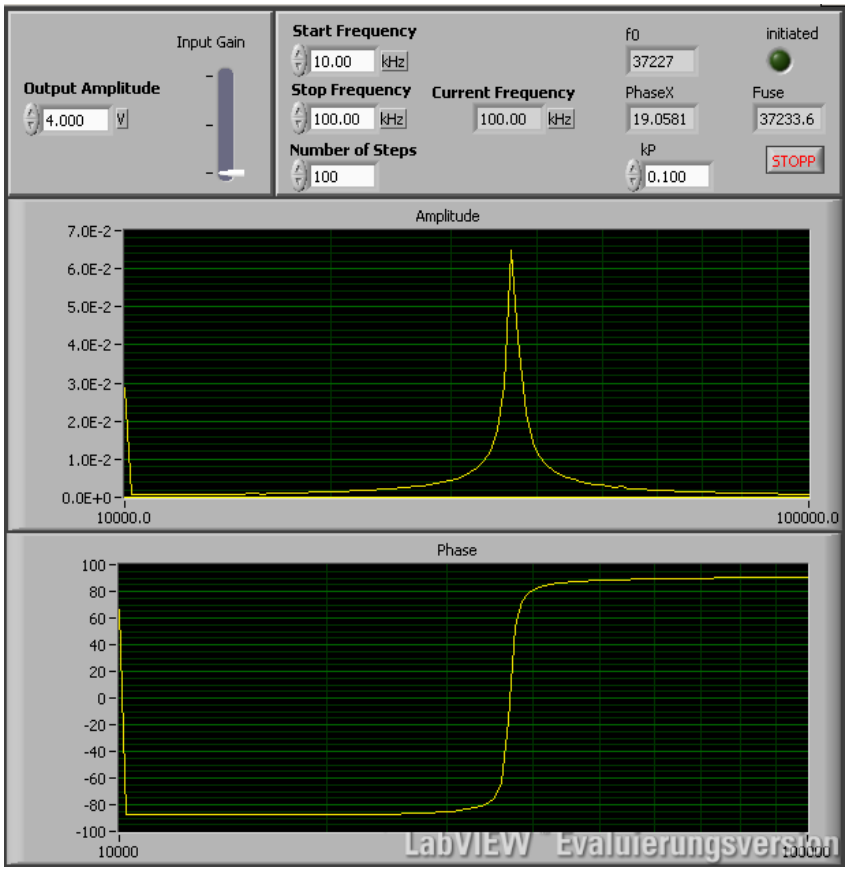
If the chosen output channel is "Phi" the "Display Range"-Combobox is not shown (its display range always is set to 180°). For the other channels, the MaxRange is calculated via the SubVI "GetUnitAndRangeFromString.vi".



The SubVI **GetUnitAndRangeFromString.vi** cuts a string consisting of value and unit into these two parts. The value from this operation (Range) is divided by the physical gain that is supplied by the physical unit (e.g. PhysGain is 1000 for a PhysUnit of "mV"). The count of digits to be shown in "SetAUX_Lockin.vi" is calculated in the second part of the VI. It depends on the Range of the supplied String.

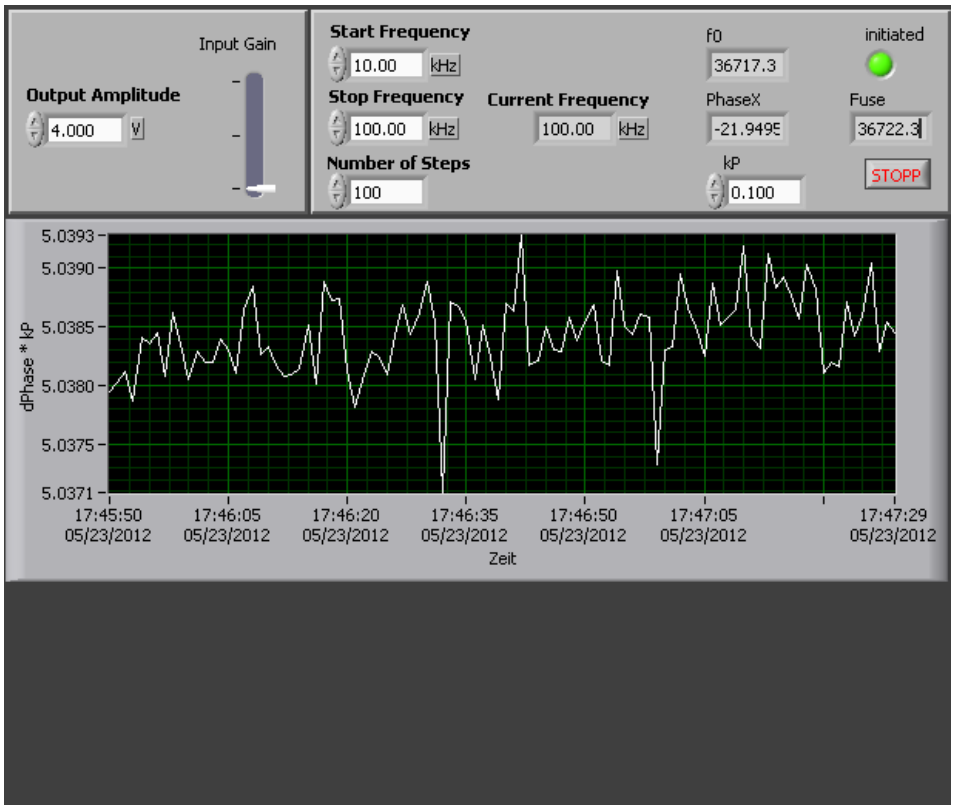


The VI "**Nase_LockIn.vi**" (in LabView9 folder, only) demonstrates how a PLL can be realized with the USBLockIn and Labview. It consists of two parts. Part 1 starts with an initialization. Start the VI after you connected everything and selected an output amplitude, the input gain as well as the start and stop frequency and the number of steps to be taken. The VI now takes a spectrum with the selected settings and shows the resulting amplitude and phase in the diagrams.



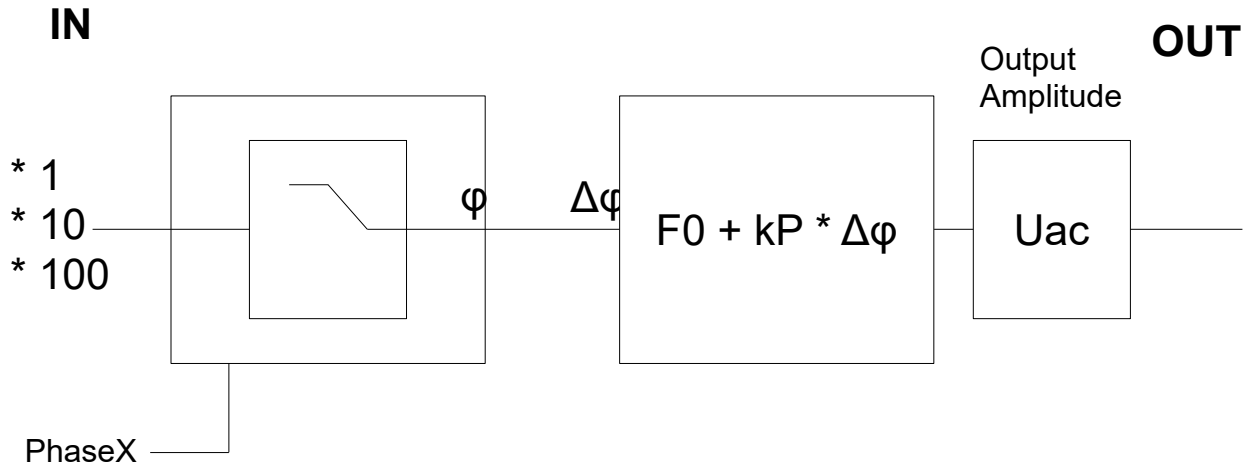
The corresponding block-diagram shows how the frequencies are calculated for each step and set via the VI "Input_LockIn.vi". Then the channels 2 and 3 are polled for getting the amplitude and phase at the actual frequency.

For the second step, please click with the left mouse button into the amplitude-graph at the resonance frequency. This frequency will be set and the phase at this point will also be set as the reference phase. These two values will be shown as "f0" and "PhaseX" accordingly. Now the two diagrams become invisible and a new graph appears. The new diagram shows the new phase multiplied with the chosen K_P over the passing time, like shown in the picture below:



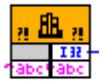
K_P * Phase will be added to the set frequency "f0" and stored in "F_{use}" to set the new used frequency of the LockIn amplifier.

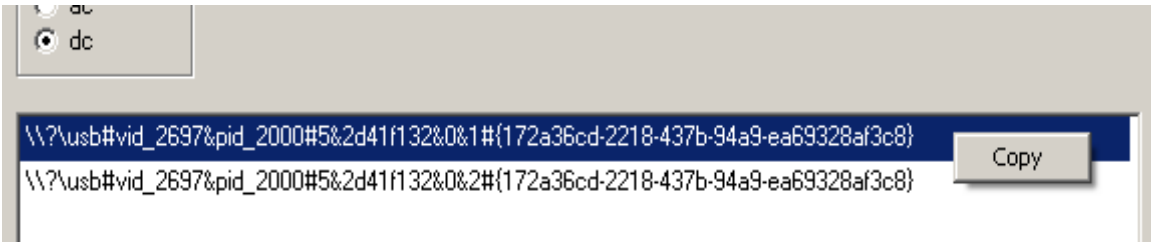
The corresponding scheme of this nose application is shown below:



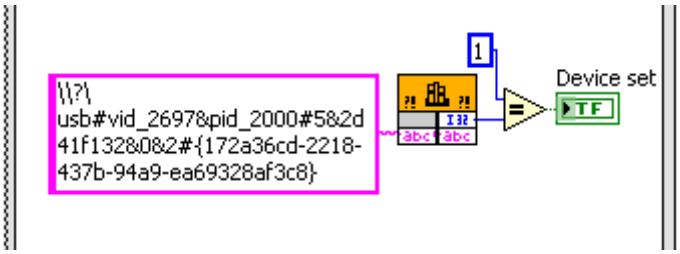
8.4 Multiple LockIn Amplifiers in LabView

To work with more than one USBLockIn250 at one PC, one has to follow these steps:

- generate a copy of the "Lockin.dll" with a different name (e.g. Lockin2.dll)
- open "Named_Input_lockin.vi" in LabView
- double-click on <u>each</u> 'external library'-symbol  and change the used library to the new ".dll"
- open the program "Lockin.exe"
- enable "View > Devices" (if not already shown)
- right-click on the device string that corresponds to the chosen USBLockIn250 and copy it



- paste the copied string into the input of the function "SetUsedDevice" in the vi-file



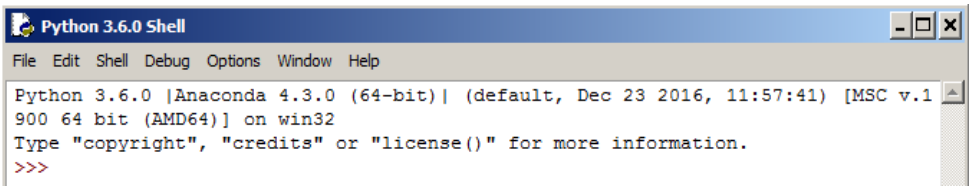
Now, this ".vi" is set specifically for the chosen lockin. Repeat the above steps to create more ".dll" and ".vi" for the other USBLockIn250.
(Don't forget to close the 'Lockin.exe' again for testing the '.vi'!)

9 REMOTE CONTROL IN PYTHON 3

Note: Python and DLL-functions will not run properly, when the test software "lockin.exe" never has been started as Administrator.

9.1 FILES AND LOCATIONS

After starting Python it shows whether it is a 64-bit or 32-bit version:



As a first step, copy the appropriate DLL (DLL/Win32 or Win64) in operation directory of your Python application. The 'lockin.dll' is best located in the same directory as the py-files or in any path that is known to Python as library path.
In our Examples we're using Python 3.6.0. with Anaconda 4.3.0. Also, the library **Matplotlib** need sot be installed to be able to run the examples.

9.2 CALL DLL FUNCTIONS IN PYTHON

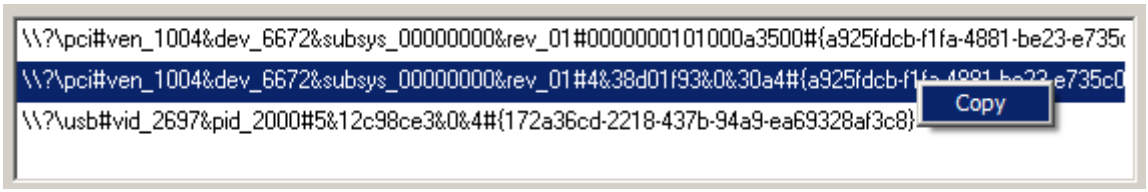
The 'lockin.dll' is imported as a CDLL. To call DLL functions in Python, one has to import their arguments as ctypes:

```
integer     : ctypes.c_int32
double      : ctypes.c_double
String      :                                          ctypes.c_wchar_p
```

9.3 MULTIPLE DEVICES IN PYTHON

To work with more than one Lockin Device at one PC, one has to follow the following steps:
(1) Each independent lockin amplifier hardware attached to the PC requires its own independent DLL. Thus, copy the Lockin.dll and rename it (e.g. into Lockin2.dll).

- The DLL and MyClass.py shall be found in the same directory like the Python files.
- The associated Lockin.exe has to be started at leats once as Administrator so that all required Registry Entries have been created in advance (Note: the name and place of the registry entries depends on whether 32-bit or 64-bit is used)
- The single lockin devices are accessed by calling their DeviceID. In order to find out the DeviceID, use the device list of the Lockin.exe and copy the full line (which is the complete Device ID) into Clipboard:



- Paste this "DeviceID" into the class MyDevice(DeviceID, DLL Name) in MyClass.py. The variable "DLL Name" is handed over as String.
  It is important to use double-backlashes inside the "DeviceID" string in Python as shown here:
  ID1="\\\\?\\pci#ven_1004&dev_6672&subsys_00000000&rev_01#0000000101000a3500#{a925fdcb-f1fa-4881-be23-e735c03bfb68}"
  As the string for this Device ID is so terribly long, its useful to hold it in a variable (here: ID1).

- For each hardware device, an independent "MyDevice" object has to be created:

```
import MyClass
Device1 = MyClass.MyDevice(ID1, "Lockin.dll")
Device2 = MyClass.MyDevice(ID2, "Lockin2.dll")
```

If only one device is connected to the PC, the ID of this device is not required to be known. One can call this single device by:

```
Device1 = MyClass.MyDevice(", "Lockin.dll")
```

9.4 Python Example Files

MyClass.py does not only contain the class MyDevice(Device ID, DLL Name) but also holds entries for all DLL functions. Start with:

```
import MyClass
Device1 = MyClass.MyDevice(", "Lockin.dll")
```

9.4.1 Spectrum Analyzer

Sweep_Frequency_LIA.py graphically displays the results of a frequency sweep taken with the lockin amplifier. As input parameters, start and stop frequency as well as the number of points in the spectrum are entered.

SIMPLE FREQUENCY SWEEP

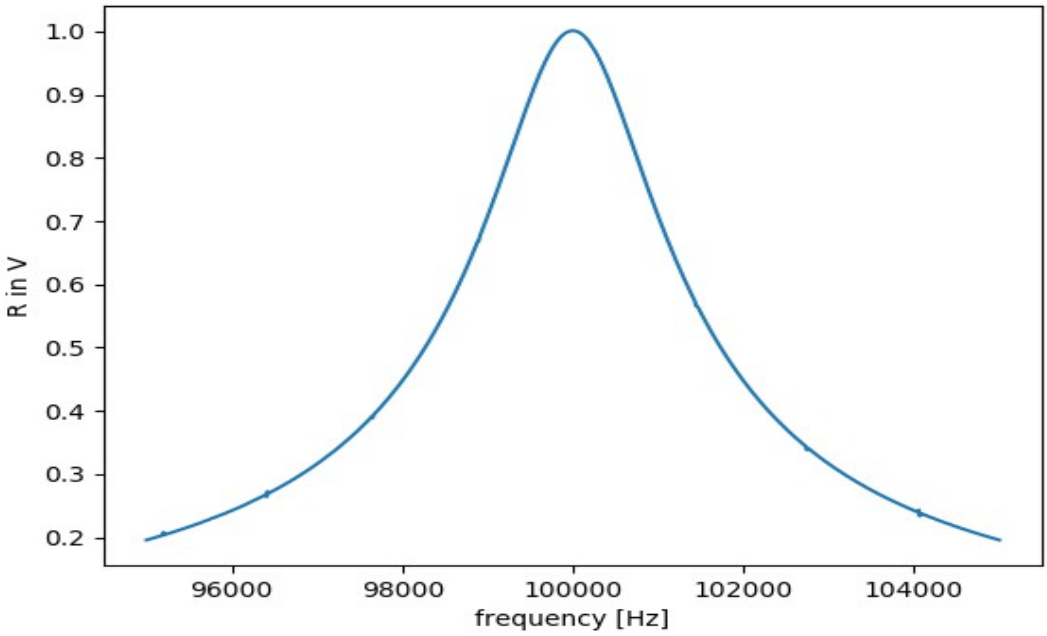


Figure 42: Detected amplitude of a sine-wave input signal with 1 V amplitude and 100 kHz frequency from an function generator. The time constant is set to 1 ms, so that the detection bandwidth is 1 kHz.

One nice way to understand the effect of the **time constant** onto the bandwidth of lockin detection is to provide an ideal sine wave with one well defined frequency from an external function generator to the lockin's input. Then, sweep the centre frequency of the lockin. If the centre frequency equals the external sine wave frequency, it detects exactly its amplitude. If the centre frequency is 1/t off the external sine wave, it detects 0.7 times the amplitude. It practically is folding the lockin's detection bandwidth with the delta-spike like input and thus visualizing the detection bandwidth itself.

EXAMPLE FOURIER ANALYSIS

Lockin amplifiers can be used to transform time dependent signals into the frequency space and thus detect periodicities in these signals. This method is also known as Fourier analysis or spectrum analyser function. One easy to understand example is the **Fourier analysis of a square signal.** Square signals can be mathematically generated from a series of an infinite amount of polynomials containing the odd orders only. Thus, a f0 = 10 kHz square wave analyzed with a spectrum analyser shows all frequency components (n+1)*f0 with n being a whole number n = 0 ... ∞.
In analogy, the frequency spectrum taken with a lockin amplifier shows the result in Fig. 53:

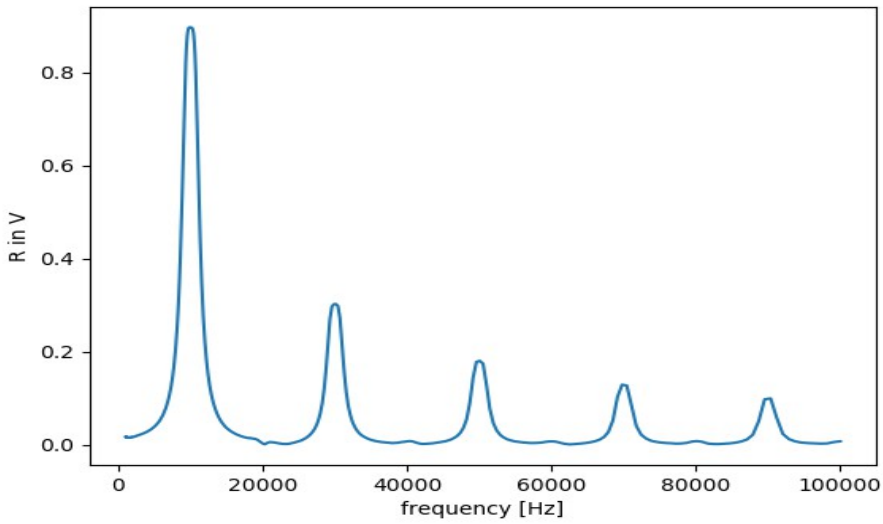


Figure 43: Fourier analysed square wave with 10 kHz frequency. Detection bandwidth: 1 kHz (τ = 1 ms)

9.4.2 TRANSIENT RECORDER

The application **Transient_recorder.py** records the input signal versus time and displays it graphically:

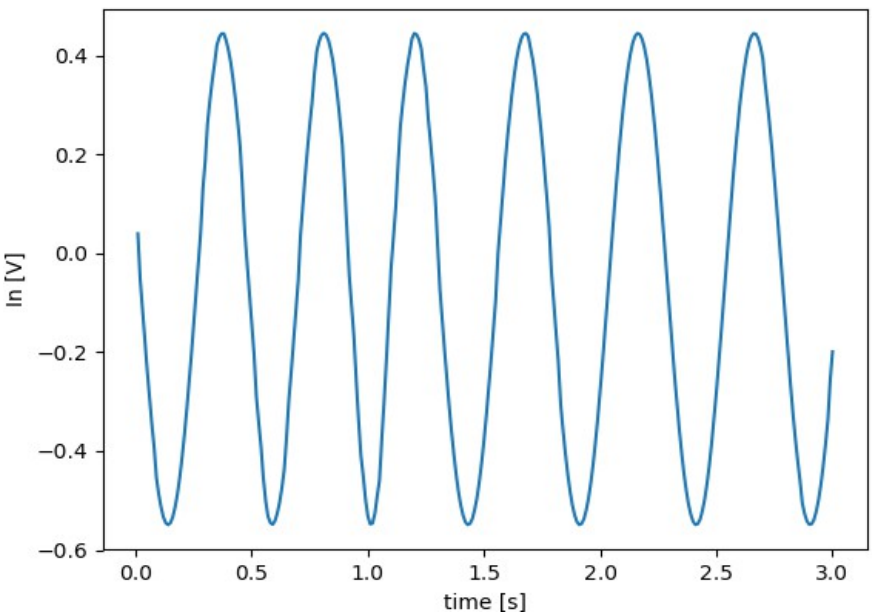


Figure 44: Graphical output of the transient recorder application.

9.4.3 Network Analyser

In contrast to a Fourier transform analysis, Network analysers usually display amplitude and phase vs. frequency and the amplitude is shown in a double logarithmic plot (Bode Plot). Thus, the application **Network_analyzer.py** generates two graphs, one for the Amplitude ® and the second for the phase, displaying them in one plot with two axis

Example Measurement Resonance

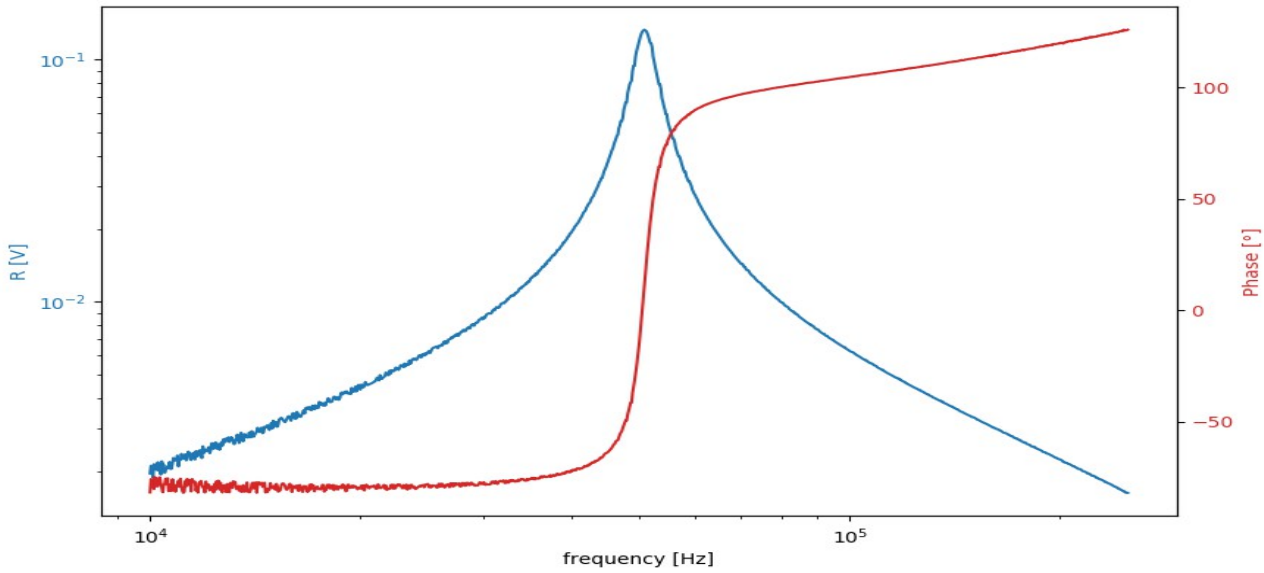


Figure 45: Bode Plot of a 50 kHz oscillation analysed with a lockin amplifier.

Example Acoustic Resonance

It is know that different geometrical shapes of bodies react different on acoustic signals. In dependence on size and shape, they might amplify certain frequencies that are resonant to standing waves inside these geometries.

$$f = \frac{n\,v}{2(L+0.8\,d)}$$

A tube, for instance, shows resonances dependent on its diameter and length according to the following formula:

The result of the following measurement was taken from a tube with a length L = 0.257 m and a diameter of d = 0.043 m. Using the sound speed in air v = 343 m/s, the first 8 resonance frequencies can be calculated as shown in the table below:

n	calculation	measurement
1	588 Hz	498 Hz
2	1177 Hz	1058 Hz
3	1765 Hz	1645 Hz
4	2353 Hz	2264 Hz
5	2942 Hz	2900 Hz
6	3530 Hz	3536 Hz
7	4118 Hz	4155 Hz
8	4707 Hz	4774 Hz

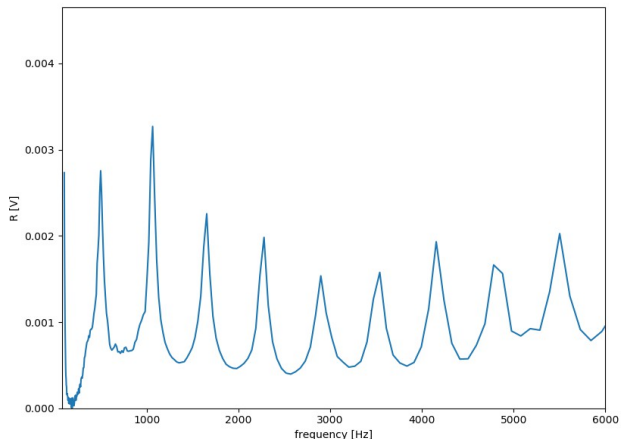
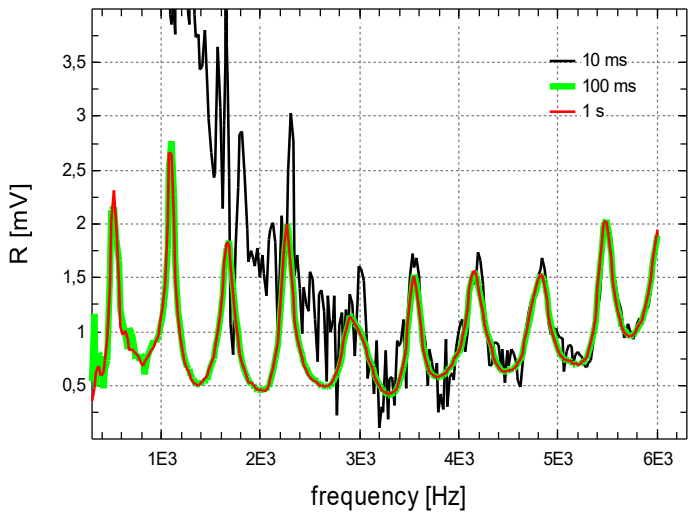


Figure 46: Diagram shows the acoustic resonance by using a tube (python). Output Amplitude 0.6 V

For the experimental realization, a loud speaker placed at the entrance of the tube was driven by the reference output of the lockin amplifier, while a microphone positioned at the open end detects the amplitude of sound. Now, the frequency of the loud speaker was swept while keeping its amplitude constant. As a result, the transfer function of this tube showing the acoustical signal to be amplified at the resonance frequencies of this tube can be detected.

The maxima of this frequency spectrum can be compared with the calculated maxima finding a nice agreement between experiment and theory.

Figure 47: Acoustic resonance in dependence of the time constant



(Ampl. 0.7 V)

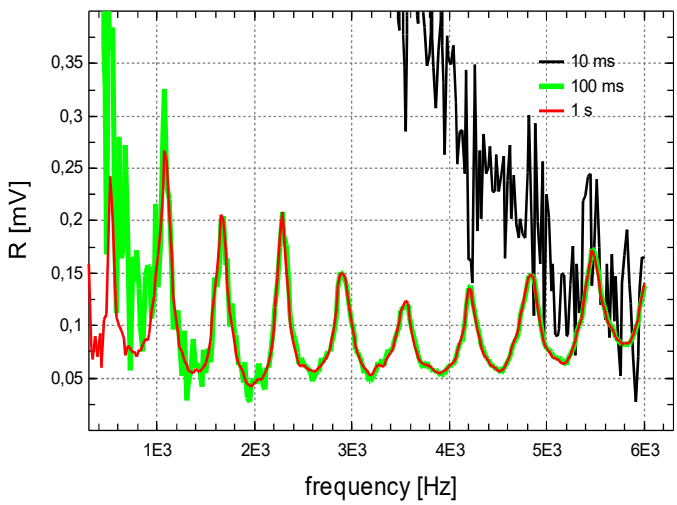


Figure 48: Acoustic resonance in dependence of the time constant (Ampl. 70 mV)

In this example, the width of the found resonance peaks is much wider than the detection bandwidth. This can be verified by changing the detection bandwidth (100 ms time constant vs. 1 s time constant). As both time constants deliver the same result, the measurement is not distorted by the detection bandwidth.

Too small time constants (10 ms) on the other hand result in a larger noise level.

In addition to this, the effect of an dc offset on the input signal can be discussed at this example. The microphone delivers a signal with a certain DC offset. With short time constants, this signal at f = 0 Hz is not suppressed enough and thus one finds a strong signal increase towards small frequencies.

This signal increase gets even larger if the total signal becomes smaller because the excitation amplitude is dropped from 700 mV down to 70 mV (Fig. 55)

9.4.4 Parametric Sweep (Only USBLockin)

The application **Parameter_sweep.py** creates a ramp at an AUX Channel. It calls the DLL function SetLockinAUX(Channel, Input, Voltage). "Channel" selects between the AUX Channel either to be 0 (AUX1) or to be 1 (AUX2). The "Input" defines which of the available input signals is displayed: In → 0; X → 10; Y → 11; R → 12; Phi → 13.
Finally, it displays two graphs: In over AUX and AC - R over AUX.

Example Diode AUX

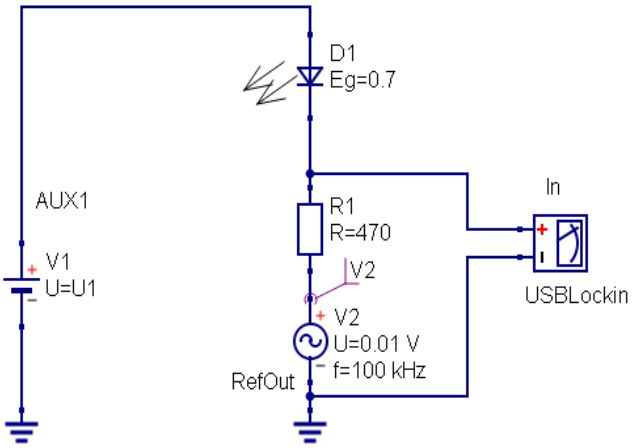
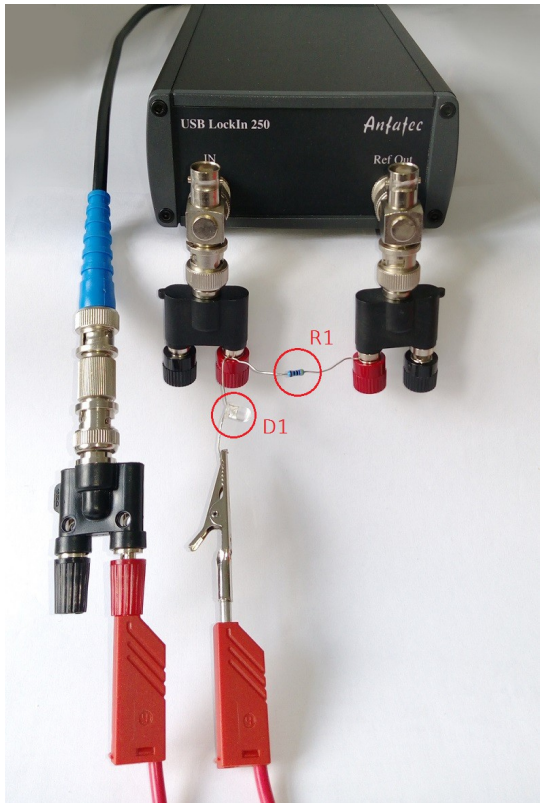


Figure 49: Circuit diagram of qucs/App_Diode.sch



These two images show the schematics of the circuit used for the experiment as well as a photograph of the experimental setup.
In this example, a diode (light emitting diode) is used as DUT (device under test). It nicely

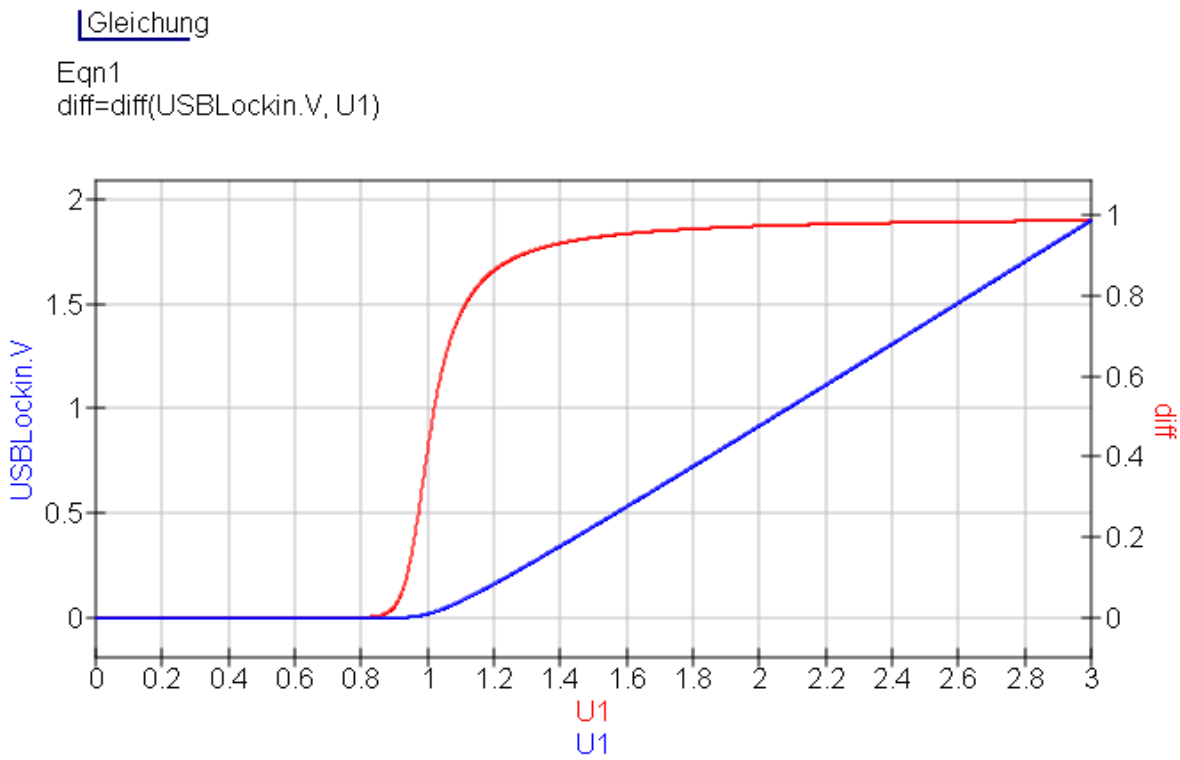


Figure 50: blue: approximate voltage over R1; red: differentiation voltage over R1(blue)

demonstrates that a lockin amplifier practically detects the derivative of a signal vs. its modulation source. The blue graph In(Udc) shows the voltage dropping over the resistance in series and doing so indirectly detects the current through the diode. The red graph R1(Udc), on the other hand, exactly looks like the derivative of the blue graph.
In order to emphasize this, the results In(Udc) can be taken into a calculation software, the dI/dV value can be calculated mathematically and plotted together with R(Udc) in one diagram:

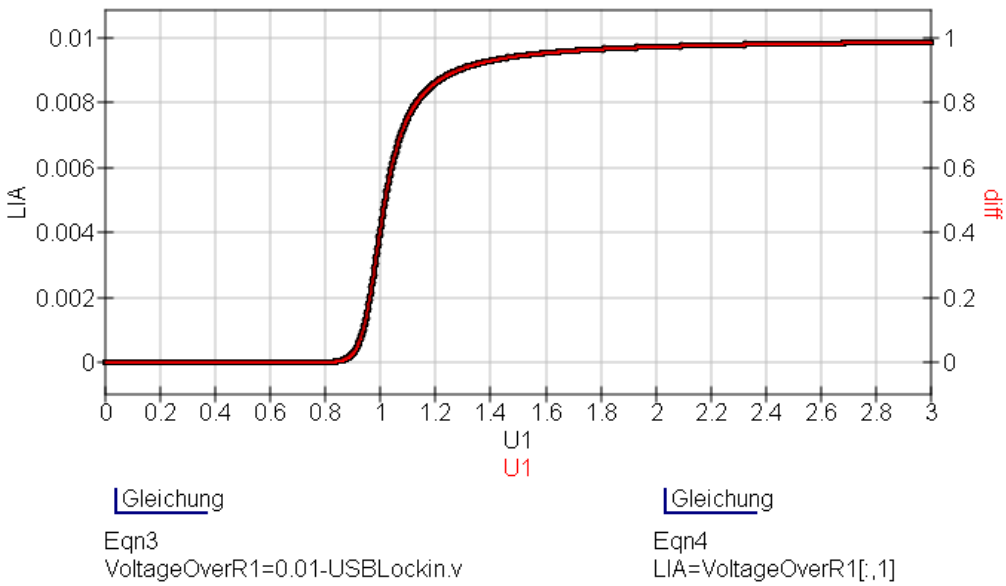


Figure 51: qucs diagram

One finds a nice accordance between these two signals.

AC/DC Coupling

Diagram 1 (In):

- DC
- Ampl = 0 V

Diagram 2 (R):

- AC
- f = 100 kHz
- Ampl = 0.01 V
- τ = 10 ms

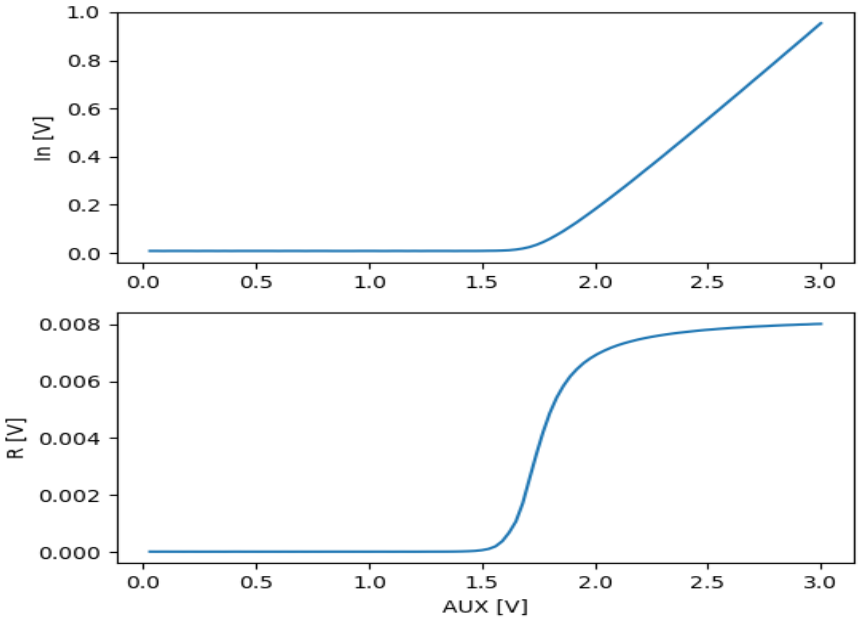
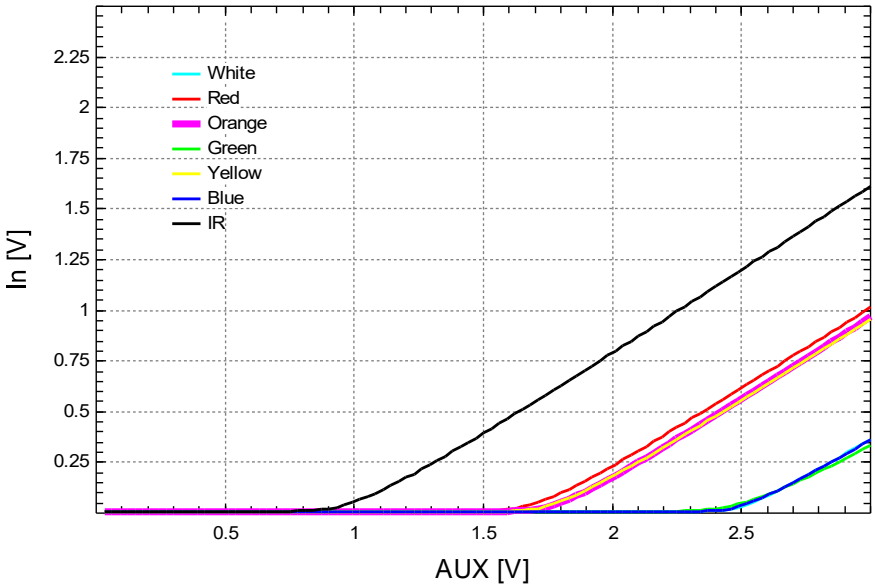
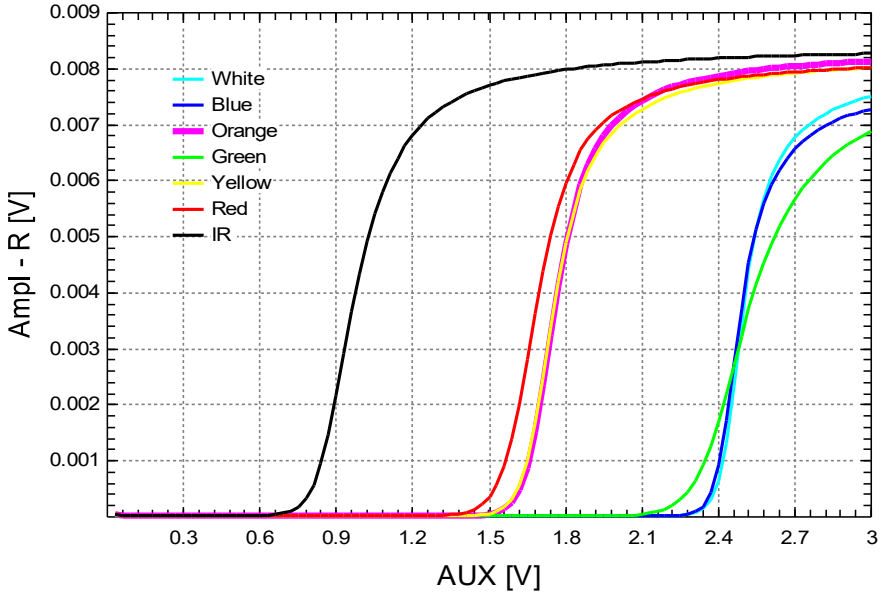


Figure 52: Python Plot



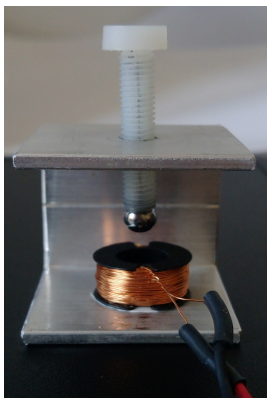


Using this set-up to investigate the threshold voltages of different light emitting diodes as shown in the two graphs above, is an excellent teaching example for students to understand the way lockin amplifiers operate.

9.4.5 Multiple Devices

The application **Multiple_Devices.py** is an simple example that demonstrates how to handle with multiple lockin devices in parallel. In this example we have 2 devices. The output of one device is connected to the input of the other device and vise versa.
If both devices are operating on the same frequency, the function GetLockinChannel(2) applied to device 2 detects the output amplitude of device 1 and vice versa.

9.4.6 Experiment: Distance Measurement



This example experiment bases on the fact, that a ferromagnetic metal brought into the centre of a coil changes the inductance of the coil.
The schematics in Fig. 54 shows that the reference output of the lockin amplifier generates an alternating current in the coil and thus an alternating magnetic field. The series resistor R2 is used to measure this current.
The metal sphere is approached from the top. It is centrally mounted to a screw with a 0.8 mm /turn threat, so that 25 turn end up to be 20 mm distance.

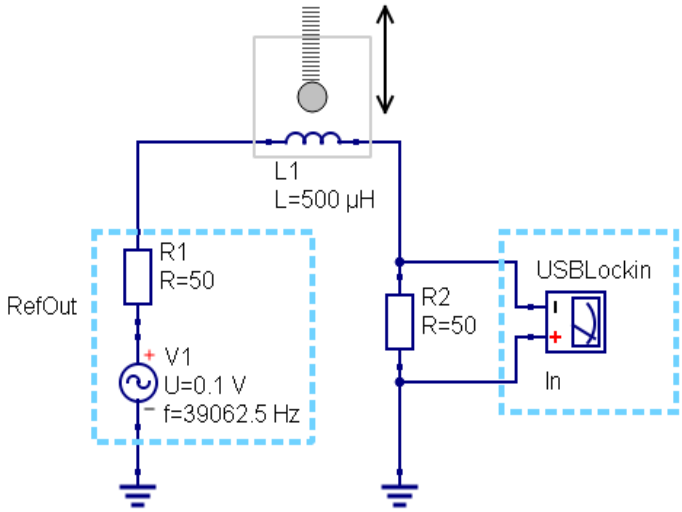


Figure 54: Circuit diagram of the distance measurement qucs/Distance.sch

Figure 53: Coil connected to the USBLockin

The operation frequency for this experiment is chosen to be 39.0625 kHz, just high enough that the inductive current amplitude can be easily measured vs. distance:

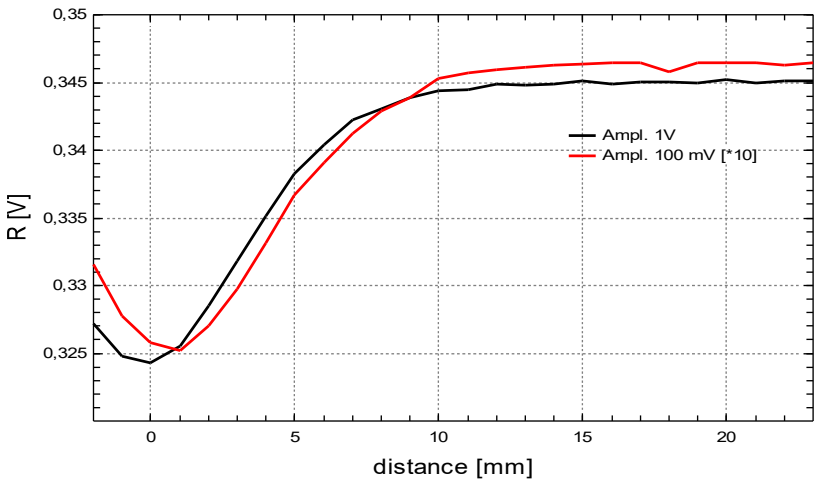


Figure 55: Curve diagram showing the Amplitude over the distance. Amplitude = 1V / 0.1 V, τ = 20 ms, f = 39062.5 Hz

10 REMOTE CONTROL IN SCILAB

Note: Scilab and DLL-functions will not run properly, when the test software "lockin.exe" never has been started as Administrator.

10.1 FILES AND LOCATIONS

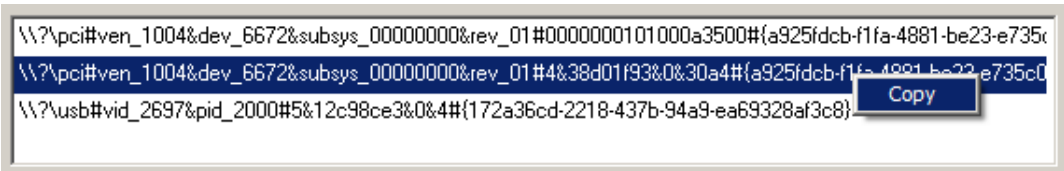
The DLL provided for Scilab has been translated with different settings than the DLL used for Python or Labview. Its named differently ("LockinSciLab.dll") and for 32-bit or 64-bit versions, it can be found in the folder "..:\DLL\SciLab\Win32" or "F:\DLL\SciLab\Win32" on the USB stick, respectively. Copy the correct DLL version (32/64 Bit) to the library path of Scilab or into the directory your Scilab files will be hold. In our examples, SciLab 5.5.2 is used.

10.2 CALL DLL FUNCTIONS IN SCILAB

The *.sci file 'DLLFunctions.sci' hold all DLL functions and includes the DLL "LockinSciLab.dll". This file needs to be in the same folder like the other SciLab files. In order to include this file, call exec(DLLFunctions.sci), which allows multiple access to DLL functions.
A DLL function call in SciLab looks like this:

```
call('SetLockInFreq', 10000, 1, 'd', 'out', [1,1], 1, 'd');
```

If several lockin devices are connected, the function SetUsedDevice(DeviceID) allows to define, which device is actively reacting on the following command. The parameter "DeviceID" can be copied from the Devices list inside the lockin.exe:



10.3 SCILAB EXAMPLE FILES

10.3.1 SPECTRUM ANALYSER

The spectrum analyser application in Scilab is called **Sweep_Frequency_LIA.py**.
In order to test it, connect a signal from an external signal generator to the input and sweep the internal reference over this external frequency.
For the example in Fig. 57, a 2mV rms 100 kHz signal was connected to the input. The detection bandwidth was set to 100 Hz (10 ms) so that the result is best visualized in the frequency range of f_{centre} ± 200 Hz.
Providing an 10 kHz square wave at the input, this applications shows all upper odd harmonics of the base frequency (30 kHz, 50 kHz , ..) as well.

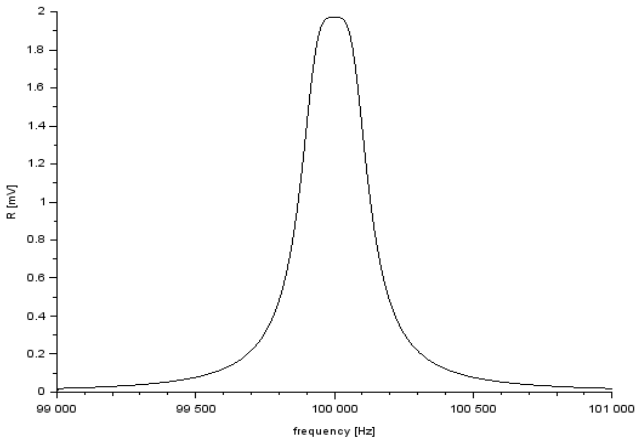


Figure 56: **Time Constant** *= 10 ms, Senderfrequenz externer Generator 100 kHz*

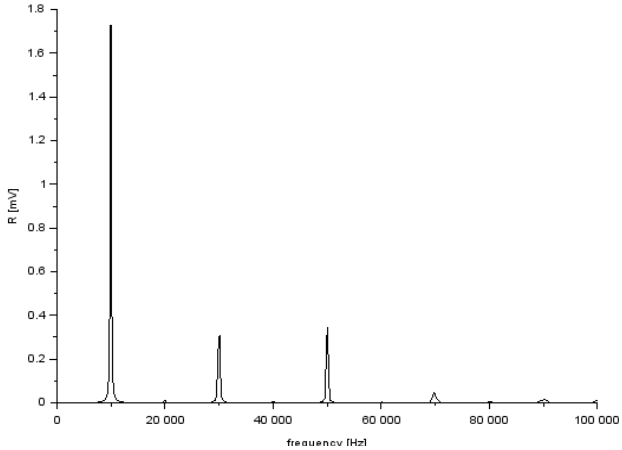
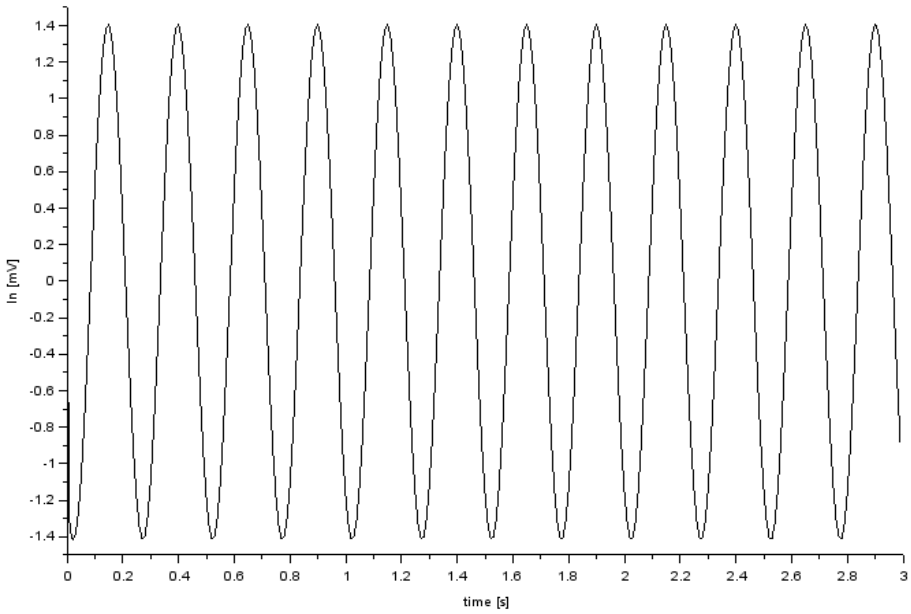


Figure 57: see ***see Fourier Transformation*** *of an Square Signal, f = 10kHz*

10.3.2 Transient Recorder

Another usefull application is the data acquisition vs. time demonstrated in the example **Transient_Recorder.sce**. In order to test it, connect input and output of the lockin amplifier and set the internal reference to a low frequency. Then, record the dc value of this output by displaying the signal In. Fig. 58 shows the result that also makes clear, how rms rated ac voltages measured or output with the lockin amplifier translate into peak-to-peak values.



- *Figure 58: Input connected to output and output of the lockin amplifier set to 4 Hz with 1 Vrms. Udc (= In) recorded versus time, showing that 1 V rms equals 2.8 Vpp amplitude.*

10.3.3 Network analyser

Network analysers detect amplitude and phase of a signal in the frequency space and plot them in the Bode Plot, with a double-logarithmic scale for amplitude and frequency. This type of data display is realized in **Network_Analyzer.sce**. The example in Fig. 59 shows the result, if a 50 kHz, 200 mV signal is provided at the lockin amplifiers input.

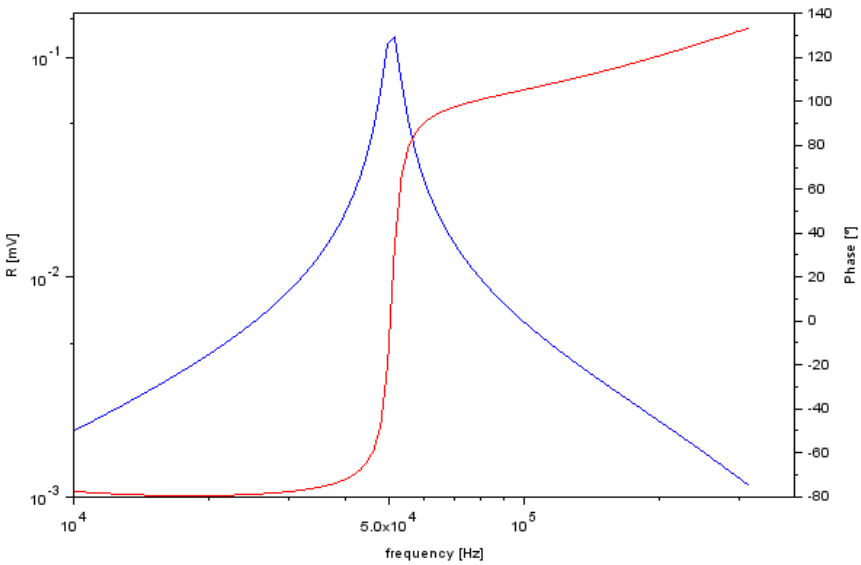
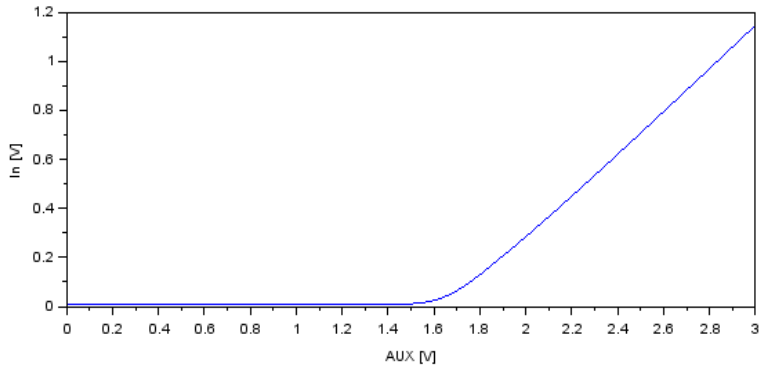


Figure 59: Bode plot of a 50 kHz, 20 mV oscillation analyzded with the lockin amplifier.

See also: **Measurement Resonance**

10.3.4 Parametric Sweep (Only USBLockin)

With the example **Parameter_sweep.py**, a linear voltage ramp is created at an AUX Channel using the DLL function SetLockinAUX(Channel, Input, Voltage). "Channel" defines which Aux output is used. "Input" defines which signal will be displayed: In → 0; X → 10; Y → 11; R → 12; Phi → 13.



Basically, this examples shows how to investigate I(V)- and dI/dV-curves with a lockin amplifier. Two graphs display In(AUX) and R(AUX) as result. Fig. 56 Shown the result one obtains if a diode is taken as device under test, which is quite illustrative due to the non-linear I(V) curve of a diode.

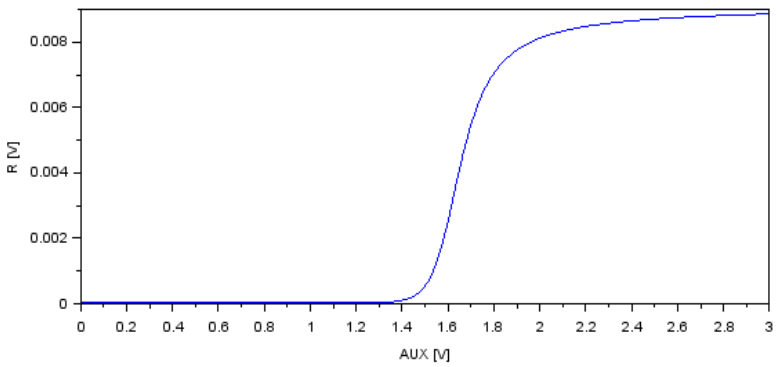


*Figure 60: Output graphs of the Parameter_Sweep.sce taken with a diode as device under test. For better understanding of the circuitry, please refer to "**Diode AUX** ".*

11 EXAMPLE: ELECTRICAL FORCE MICROSCOPE

In this chapter, Electrical Force Microscopy (EFM) is briefly presented as an example for the application of lock-in amplifiers.
EFM is one of the detection technique used to enhance the capabilities of the well-established atomic force microscopy (AFM). Its special aim is to detect electrical forces to learn something about the electrical properties of a surface, for example about the local distribution of surface potentials on electronic devices, work functions, surface charges or different dopant concentrations in semiconducting materials.

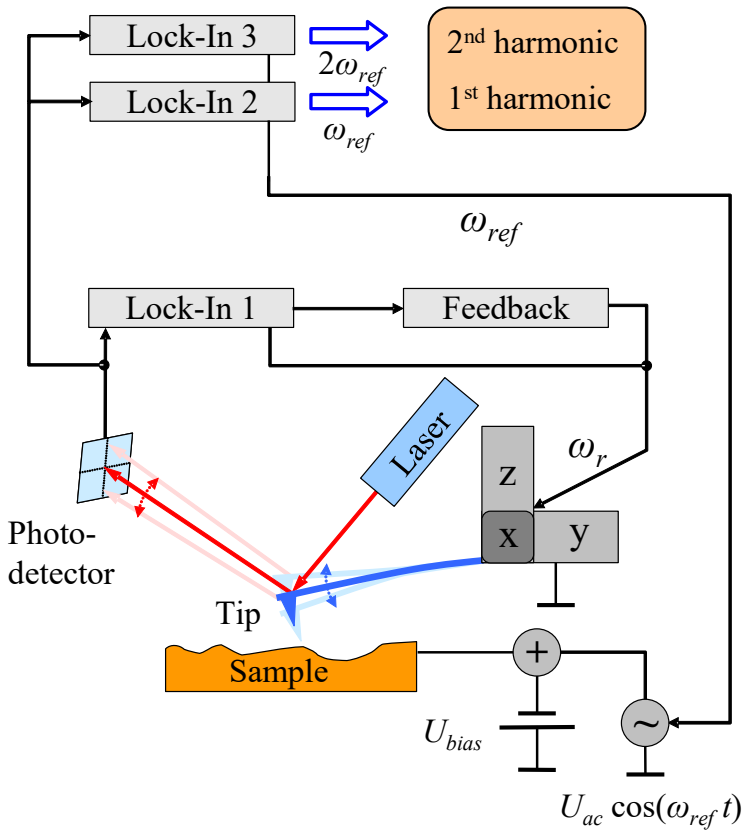


Figure 61: Schematic diagram of the EFM experimental setup.

The schematic diagram sketches an AFM operated the non-destructive dynamic non-contact mode. The cantilever (with the metallic tip) is oscillating in its 1st mechanical Eigenmode ωr perpendicularly to the surface. For position sensing of the cantilever, a laser beam deflection detection with a position-sensitive photo-detector is employed. A lock-in amplifier analyses the detector signal at the cantilever resonance frequency ωr and passes the determined amplitude value to a feedback control system that re-adjusts the tip-sample distance. The required displacement of the z-piezo can be recorded as topography signal.
In addition to the topography, EFM detects electrical forces. These forces are proportional to the derivative of the capacitance *C* of the tip-sample arrangement with respect to the tip-sample distance *z* and proportional to the potential difference *U* squared:

$$F_{el} = -\frac{1}{2}\frac{\mathrm{d}C}{\mathrm{d}z}U^2 \quad . \qquad (5)$$

A possible voltage dependency of *C* is neglected in this consideration.

The voltage U contains a direct voltage part U_{DC} and an alternating voltage part U_{AC}:

$$U = U_{DC} + U_{AC} \cdot \cos(\omega_{ref} t) \quad . \qquad (6)$$

U_{DC} consists of an additional applied bias voltage and, what is especially interesting from the physical point of view, potential differences caused by different electronic work functions and charges. For separating the impact of these electrical forces from other forces (e.g. van der Waals forces), U has to be modulated at the frequency ω_{ref}. As a result of this, the photo-detector signal is modulated at this frequency as well.

Inserting equation (6) in (5) and using power-reduction formulae of trigonometric functions one can expect a force between tip and sample at the frequency ω_{ref} as well as at $2 \cdot \omega_{ref}$. Figure 62 confirms this prediction: It shows the frequency spectrum of the cantilever oscillation with clear signals at the mechanical excitation frequency ω_r and at the frequencies of the first and second harmonic of the electrical excitation.

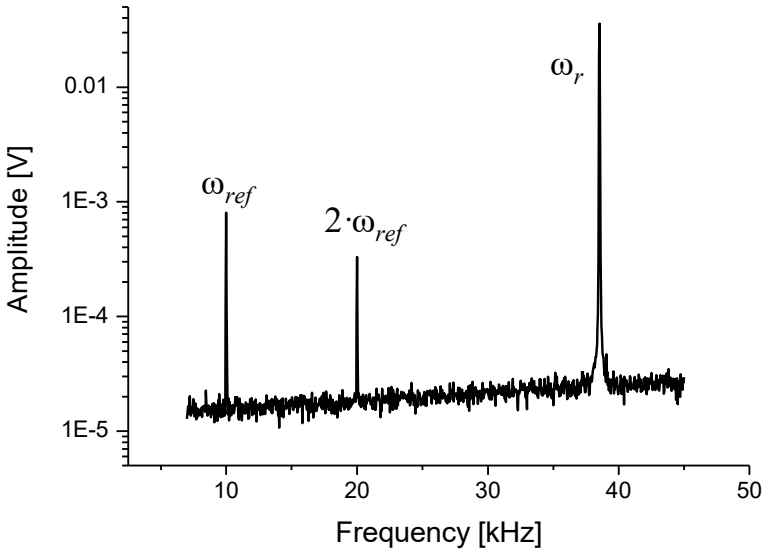


Figure 62: Frequency spectrum of the oscillating cantilever

Now two additional lock-in amplifiers (see Figure 61) can be used to analyse the photo-detector signal at the frequency ω_{ref} and $2 \cdot \omega_{ref}$ simultaneously. The measured amplitudes of the signals are proportional to the strength of the electrical force components.

12 REVISION HISTORY

12.1 DRIVER REVISIONS

Download the newest driver and software here:
https://www.anfatec.de/support/01_lockin/lockin_amplifier_support.html

June 2018

- 0.7.0.8
  This is the first driver that acts as exclusive driver. Only one application can access the driver at a time.

January 2023

- 0.7.0.12
  Improved overload handling

12.2 HARDWARE REVISIONS

December 2017 – all S/Ns -100 and up

- internal board was changed with the aim to improve serviceability

March 2018 – all S/Ns -200 and up

- minor changes on internal board

January 2023

- changes in overload handling

12.3 CHANGES IN THE MANUAL

December 2017

- added description for output scaling

May 2018
general changes in the manual concerning:

- updated specification
- updated installation routine
- updated Images 4 and 9
- new images Fig. 10
- inclusion of PID description
- correction in the description of filters

August 2018

- Python Remote
- SciLab Remote

June 2022

- DLL Description

September 2022

- command “SetLockInTimeBase” corrected p. 36

January 2023

- DLL Function “GetLockinStatus” was extended