week1

June 15, 2023

Notebook file for Week 1

Import libraries. The nmrbase folder needs to be located within the folder containing this Jupyter notebook

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import nmrbase.expbase as expbase
import nmrbase.expdta as expdta
```

Tuning response experiment

[ ]:
```
filename = r"../DIRECTORY/FILENAME"    #Defines the path to the data file. Here␣
↪the path is relative to the current folder.
```

[ ]:
```
a = expbase.expbase()                        # create an experiment base object (a)
a.load(filename)                             # load data into a
f1 = plt.figure()
ax1 = f1.subplots()              # create axes for a figure

a.plottm(ax1,1)                               # plot the time domain data of the␣
↪first scan from the data set
```

[ ]:
```
## ADDITIONAL TASKS:
## include statements such as ax1.set_ylabel() to label axes
## use statements to change appearance such as font size, etc.
## Adjust horizontal axis limits to display only the ring-down signal with␣
↪statements such as ax1.set_xlim([0,1]). You will need to select appropriate␣
↪values.

f1    # show the figure again with modifications
```

[ ]:
```
f2=plt.figure()
ax2=f2.subplots()

a.pproc['ftmin']=0                           # time interval for Fourier transform␣
↪(0 = all data)
a.pproc['ftmax']=0
```

```
a.pproc['ffmin']=0                          # frequency interval for spectrum␣
 ↪display (0 = all data)
a.pproc['ffmax']=0
a.proc()                                 # calculate Fourier transform of the data␣
 ↪in a

a.plotfrq(ax2,1)                         # plot the frequency domain data of the␣
 ↪first scan from the data set

## TASKS:
## change parameter "ftmin" and "ftmax" to select the time interval␣
 ↪corresponding to the ringdown for the Fourier transform
## change parameter "ffmin" and "ffmax" to select the frequency range of the␣
 ↪tuning response signal
## include statements to label axes
## use statements to change appearance such as font size, etc.
## Adjust horizontal axis limits to display the spectrum of the ring-down signal
```

[ ]:
```
i=np.argmax(a.frq[0].dta)      # find data point index of largest peak in␣
 ↪spectrum
f=a.frq[0].ind_to_x(i)         # find frequency corresponding to largest peak in␣
 ↪spectrum
print("Tuning frequency: {:.2f} Hz".format(f))     # this statement prints the␣
 ↪identified frequency in a nice format
```

[ ]:
```
# plot figure again with line indicating maximum peak position. Notice this␣
 ↪works only if previous parameters have been correctly selected.
ax2.axvline(x = f, color = 'r', label = 'Larmor Frequency')

f2   # show the figure again with the line included
```

Pulse length calibration. NOTE: To analyze and report multiple datasets, simply copy the necessary blocks of code and change the name of the veriables.

[ ]:
```
filename2 = r"../DIRECTORY/FILENAME"         # path of data file
```

[ ]:
```
a2 = expbase.expbase()
a2.load(filename2)                          # load data

f3=plt.figure()
ax3=f3.subplots()
a2.plottm(ax3,5)
```

[ ]:
```
## TASKS:
## fine tune with ax3.set_xlim and ax3.set_ylim parameters to zoom in on the␣
 ↪echo
```

```
f3   # display adapted figrue
```

[ ]:
```
a2.pproc['digfmin']=1500     # set appropriate digital filter parameters
a2.pproc['digfmax']=3500
a2.digfilt()                 # perform digital filter operation
```

[ ]:
```
f4=plt.figure()
ax4=f4.subplots()
a2.plottm(ax4,5)   # plot the 5th scan (the 1st scan doesn't have signal␣
 ↪because the pulse is too short)
```

[ ]:
```
## TASKS:
## fine tune with ax4.set_xlim and ax4.set_ylim parameters to zoom in on the␣
 ↪echo

f4   # display adapted figrue
```

[ ]:
```
f5=plt.figure()
ax5 = f5.subplots()

a2.pproc['ftmin']=0                     # time interval for Fourier transform␣
 ↪(0 = all data)
a2.pproc['ftmax']=0
a2.pproc['ffmin']=0                     # frequency interval for spectrum␣
 ↪display (0 = all data)
a2.pproc['ffmax']=0
a2.proc()                            # calculate Fourier transform of the data␣
 ↪in a
## this will use the digitally filtered data from before. Instead, the original␣
 ↪data can be processed by loading it again.

a2.plotfrq(ax5,5)                    # plot the frequency domain data of the␣
 ↪first scan from the data set

## TASKS:
## change parameter "ftmin" and "ftmax" to select the time interval␣
 ↪corresponding to the ringdown for the Fourier transform
## change parameter "ffmin" and "ffmax" to select the frequency range of the␣
 ↪tuning response signal
## include statements to label axes
## use statements to change appearance such as font size, etc.
## Adjust horizontal axis limits to display the spectrum of the ring-down signal
```

[ ]:
```
# This section is to integrate the peaks in the NMR spectra to determine pulse␣
 ↪length
f5=plt.figure()
ax5=f5.subplots()
```

```
a2.pproc['intmin']=0      # set correct frequency range for integration
a2.pproc['intmax']=0

a2.integrate()               # perform integration

#find the starting pulselength and increment, then set the correct x-axis
dx=a2.pinc["inc"][0]
x0=a2.p["p1"]
print('x0 =',x0,'s , dx =',dx,'s')          # x0 is the starting pulse length,␣
↪and dx is the increment
a2.idt.x0=x0
a2.idt.dx=dx

a2.idt.plot(ax5,disp=[0])                   # disp=0 plots only the NMR signal␣
↪in trace 0
ax5.axvline(x = 0.001, color = 'r', label = '90° pulse length') # display␣
↪vertical line at 90 degree pulse length

## TASKS:
## change the intmin and intmax parameters to select the correct frequency␣
↪range for integration
## use set_xlabel and set_ylabel to set the labels to get a publication quality␣
↪figure
## adjust appearance of figure as needed
## change and report the "x" parameter to indicate the pulse length for 90°␣
↪pulse with a vertical line
```