APL_4_cell

June 11, 2021

```
[1]: if True:
         import numpy as np
         import pandas as pd

         # Add lab library
         import sys
         sys.path.insert(0, '/home/trevormjs/Documents/Science/APL/Lab')

         #----------------------------------------------------------------------#
         #                        matplotlib plotting                           #
         #----------------------------------------------------------------------#
         import matplotlib.pyplot as plt
         import matplotlib as mpl
         from jupyterthemes import jtplot
         from Helper.plotting import my_graph
         # Edit the font, font size, and axes width
         # mpl.rcParams['font.family'] = 'Avenir'
         plt.rcParams['font.size'] = 24
         plt.rcParams['axes.linewidth'] = 2
         jtplot.style(theme='monokai', context='notebook', ticks=False, grid=False)

         #----------------------------------------------------------------------#
         #                           bokeh plotting                             #
         #----------------------------------------------------------------------#
         from bokeh.plotting import figure, show, output_notebook
         from bokeh.themes import Theme
         from bokeh.io import curdoc, export_png
         from bokeh.models import Range1d, Label, ColumnDataSource, LabelSet
         from Helper.plotting import style
         output_notebook()
         # curdoc().theme = Theme(filename="../Helper/theme.yml")

         #----------------------------------------------------------------------#
         #                       error and unit handling                        #
         #----------------------------------------------------------------------#
         from uncertainties import ufloat
         import Helper.numbers as nu
```

```
    from Helper.record import Measurement, Unit

    %load_ext autoreload
    %autoreload 2
```

[2]:
```
def read_scope_csv(path, n_sigs = 1):
    ret = []
    for n in range(n_sigs):
        config = pd.read_csv(path, header=None, usecols=[1+6*n, 2+6*n]).loc[:2]
        config = [record_length, sample_interval, trigger_point] = [
            [float(val), unit] for val, unit in zip(config[1+6*n],␣
 ↪config[2+6*n])]
        config= {
            'record_length': record_length,
            'sample_interval': sample_interval,
            'trigger_point': trigger_point
        }
        display(config)
        data = pd.read_csv(path, header = None, usecols=[3+6*n, 4+6*n])
        data.columns = ['ts', 'mV']
        ret.append([data, config])
    return ret
```

[44]:
```
def clean_vis_9010_diff(pd1, pd2, first = False, filt_type='fft', filter_param=.
 ↪25, plot=True):
    fig = figure(height = 400)
    if filt_type == 'fft':
        pd1_clean = pd.Series(nu.fft_filter(pd1.mV, filter_param))
        pd2_clean = pd.Series(nu.fft_filter(pd2.mV, filter_param))
    elif filt_type == 'avg':
        pd1_clean = pd.Series(nu.moving_avg_filter(pd1.mV, filter_param))
        pd2_clean = pd.Series(nu.moving_avg_filter(pd2.mV, filter_param))
    else:
        raise ValueError('No filter match')

    print('pd1 unfiltered ', end = '')
    pd1_start, pd1_end = nu.rising_edge(pd1_clean, pd1.ts, first)[0]
    print('pd2 unfiltered ', end = '')
    pd2_start, pd2_end = nu.rising_edge(pd2_clean, pd2.ts, first)[0]

    # timerange = [pd1_start-25, pd2_end + 150]
    # pd1_clean = pd1_clean[timerange[0]:timerange[1]]
    # pd2_clean = pd2_clean[timerange[0]:timerange[1]]

    t = pd1.ts - pd1.ts[pd1_start]  # [timerange[0]:timerange[1]]
    t_interp = np.linspace(t.min(), t.max(), len(pd2_clean)*16)
    interp1 = interp1d(t, pd1_clean, kind='cubic')
    pd1_clean = pd.Series(interp1(t_interp))

    interp2 = interp1d(t, pd2_clean, kind='cubic')
    pd2_clean = pd.Series(interp2(t_interp))

    t = t_interp

    print('pd1 filtered ', end = '')
    pd1_inds, pd1_90_10_inds = nu.rising_edge(pd1_clean, t, first)
    fig.scatter(t[pd1_inds], pd1_clean[pd1_inds])
    fig.scatter(t[pd1_90_10_inds], pd1_clean[pd1_90_10_inds], color='skyblue')

    print('pd2 filtered ', end = '')
    pd2_inds, pd2_90_10_inds = nu.rising_edge(pd2_clean, t, first)
    fig.scatter(t[pd2_inds], pd2_clean[pd2_inds])
    fig.scatter(t[pd2_90_10_inds], pd2_clean[pd2_90_10_inds], color='skyblue')


    small_t = t[pd1_inds[0]-50:pd1_inds[0] + (pd1_inds[1]-pd1_inds[0])*2+100]
    small_pd1 = pd1_clean[pd1_inds[0]-50:pd1_inds[0] + (pd1_inds[1]-pd1_inds[0])*2+100]

    peaks, b, b = nu.peak(small_pd1, 1, 0, 0, 0)
    widths, _, _, _ = nu.peak_widths(small_pd1, peaks,)
    print('pd1 peak halfmax:', widths[0] * (t[1]-t[0]))


    small_t = t[pd2_inds[0]-50:pd2_inds[0] + (pd2_inds[1]-pd2_inds[0])*2+100]
    small_pd2 = pd2_clean[pd2_inds[0]-50:pd2_inds[0] + (pd2_inds[1]-pd2_inds[0])*2+100]

    peaks, b, b = nu.peak(small_pd2, 1, 0, 0, 0)
    widths, _, _, _ = nu.peak_widths(small_pd2, peaks,)
    print('pd2 peak halfmax:', widths[0] * (t[1]-t[0]))


    fig.line(t[pd1_inds[0]-100:pd2_inds[1]*2], pd1_clean[pd1_inds[0]-100:pd2_inds[1]*2], legend_label='PD1')
    fig.line(t[pd1_inds[0]-100:pd2_inds[1]*2], pd2_clean[pd1_inds[0]-100:pd2_inds[1]*2], legend_label='PD2', color='red')

    fig.line(t[pd1_inds[0]:pd1_inds[1]], pd1_clean[pd1_inds[0]:pd1_inds[1]], color = 'green', alpha = 1)
    fig.line(t[pd2_inds[0]:pd2_inds[1]], pd2_clean[pd2_inds[0]:pd2_inds[1]], color = 'green', alpha = 1)
```

```python


    fig.x_range = Range1d(t[pd1_inds[0]] - 5e-10, t[pd2_inds[1]]*1.3)
    if plot:
        show(fig)
    else:
        del fig

    return pd.DataFrame({
        'start': t[[pd1_inds[0], pd2_inds[0]]],
        'end': t[[pd1_inds[1], pd2_inds[1]]],
        '10': t[[pd1_90_10_inds[0], pd2_90_10_inds[0]]],
        '90': t[[pd1_90_10_inds[1], pd2_90_10_inds[1]]]
    }, index=['sig1', 'sig2'])
```

0.1 Test Pulse, Scope, and Laser Diode

[4]:

```python
[[squares, squares_config]] = read_scope_csv('./Data/l4_A_1_data.csv')
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [3.99999989e-09, 's'],
 'trigger_point': [4640.00039, 'Samples']}
```

[5]:

```python
fig = figure()
fig.line(squares.ts, squares.mV)
show(fig)
```

[6]:

```python
approx_amplitude = squares.mV.max() - squares.mV.min()
fourier_coefs = np.fft.fft(squares.mV, n=len(squares)*8)
fourier_freqs = np.fft.fftfreq(
    len(squares.mV)*8, squares_config['sample_interval'][0]
)

pulse_frequency = fourier_freqs[
    10:len(fourier_freqs)//2][
    fourier_coefs[10:len(fourier_freqs)//2].argmax()
]

approx_amplitude, 'mV', pulse_frequency, 'Hz'
```

[6]:

```
(5.5999997545, 'mV', 90625.00249218756, 'Hz')
```

[7]:

```python
fig = figure()
fig.line(fourier_freqs, np.abs(fourier_coefs))
show(fig)
```

0.2 Initial Optics Setup

Failed Data

[8]:
```
[[pd1, pd1_config], [pd2, pd2_config]] = read_scope_csv(
    './Data/l4_C_data.csv', 2)

fig = figure()
fig.line(pd1.ts, pd1.mV, legend_label='PD1')
fig.line(pd1.ts, pd2.mV, legend_label='PD2', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [1.9999999e-09, 's'],
 'trigger_point': [5020.00004, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [1.9999999e-09, 's'],
 'trigger_point': [5020.00004, 'Samples']}
```

[9]:
```
[[pd1, pd1_config], [pd2, pd2_config]] = read_scope_csv(
    './Data/l4_C_data3.csv', 2)

fig = figure()
fig.line(pd1.ts, pd1.mV, legend_label='PD1')
fig.line(pd1.ts, nu.fft_filter(pd2.mV, .17), legend_label='PD2', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4834.00032, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4834.00032, 'Samples']}
```

0.2.1 Loading Data

[10]:
```
[[pd1, pd1_config], [pd2, pd2_config]] = read_scope_csv(
    './Data/l4_C_data4.csv', 2)

fig = figure()
fig.line(pd1.ts, pd1.mV, legend_label='PD1')
fig.line(pd1.ts, pd2.mV, legend_label='PD2', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4932.80043, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4932.80043, 'Samples']}
```

[11]:

```
[[pd1_cable, pd1_config], [pd2_cable, pd2_config]] = read_scope_csv(
    './Data/14_C_c_data.csv', 2)

fig = figure()
fig.line(pd1_cable.ts, pd1_cable.mV, legend_label='PD1_cable')
fig.line(pd1_cable.ts, pd2_cable.mV, legend_label='PD2_cable', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4923.20015, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4923.20015, 'Samples']}
```

0.2.2 Rising Edge analysis

[12]:

```
from bokeh.palettes import Category20_4 as colors
```

[13]:

```
from Helper.numbers import rising_edge
```

First pass

[14]:

```
fig = figure()

"""
Create a time vector beginning at zero for alignment purposes.
"""
t = pd1.loc[pd1.ts >= 0, 'ts']
t.index = range(len(t))

"""
Filter all curves such that they are identical. This is really for
ease of processing but also because they really should be identical,
and any differences must represent some error.
"""
pd1_cable_clean = pd.Series(nu.fft_filter(pd1_cable.mV, .44))
pd2_cable_clean = pd.Series(nu.fft_filter(pd2_cable.mV, .34))

pd1_clean = pd.Series(nu.fft_filter(pd1.mV, .34))
pd2_clean = pd.Series(nu.fft_filter(pd2.mV, .34))

"""
Caclulate the indeces of the rising edge for all signals, and the
locations of 10% and 90%.
"""
```

```python
pd1_cable_inds, pd1_cable_90_10_inds = rising_edge(
    pd1_cable_clean, pd1_cable.ts)
pd1_inds, pd1_90_10_inds = rising_edge(
    pd1_clean, pd1_cable.ts)

pd2_cable_inds, pd2_cable_90_20_inds = rising_edge(
    pd2_cable_clean, pd2_cable.ts)
pd2_inds, pd2_90_20_inds = rising_edge(
    pd2_clean, pd2_cable.ts)

"""
Align each PD1 curve to the beginning of its rising edge.
Align each PD2 curve to the beginning of the rising edge of
the corresponding PD1.
"""
zi_pd1_cable = pd1_cable_clean[pd1_cable_inds[0]:]
zi_pd1 = pd1_clean[pd1_inds[0]:]

zi_pd2_cable = pd2_cable_clean[pd1_cable_inds[0]:]
zi_pd2 = pd2_clean[pd1_inds[0]:]

"""
Normalize each signal on the scale [0, 1].
"""
zi_pd1_cable -= zi_pd1_cable.min()
zi_pd2_cable -= zi_pd2_cable.min()
zi_pd1_cable /= zi_pd1_cable.max()
zi_pd2_cable /= zi_pd2_cable.max()

zi_pd1 -= zi_pd1.min()
zi_pd2 -= zi_pd2.min()
zi_pd1 /= zi_pd1.max()
zi_pd2 /= zi_pd2.max()

"""
Plot each pair of PD1 and PD2 signals, and calculate
their locations of etc.
"""
results = {}
for sig, name, color in zip(
    [zi_pd1_cable, zi_pd1, zi_pd2_cable, zi_pd2],
    ['PD1_cable', 'PD1', 'PD2_cable', 'PD2'],
    colors
):
    sig.index = range(len(sig))
    fig.line(t,
             sig[:len(t)],
             legend_label=name,
             color=color,
             line_width=1.6)
    times = rising_edge(sig[:len(t)], t)
    times = times[0] + times[1]
    results.update({name:[t[i] for i in times]})

fig.x_range = Range1d(-.4e-9, 3e-8)
show(fig)
results = pd.DataFrame(results, index = ['start','stop','10','90']).T
```

```
10-90 time 2.000000079649e-09
10-90 time 2.000000180569e-09
10-90 time 2.400000155e-09
10-90 time 1.9999997189999999e-09
10-90 time 2.00000005299e-09
10-90 time 2.000000028719e-09
10-90 time 2.4000002e-09
10-90 time 1.9999997220000004e-09
```

[15]:
```
diffs = pd.DataFrame(results.T[['PD2_cable', 'PD2']].values -
                     results.T[['PD1_cable', 'PD1']].values,
                     index=results.columns,
                     columns=['cable_diff', 'no-cable_diff'])
(diffs.cable_diff-diffs['no-cable_diff']).mean(), (diffs.
 ↪cable_diff-diffs['no-cable_diff']).std()
```

[15]:
```
(5.100000170067751e-09, 2.00000170531249e-10)
```

Second Pass

[18]:
```
from scipy.interpolate import interp1d
```

[19]:
```
res = clean_vis_9010_diff(pd1, pd2)
res.iloc[1] - res.iloc[0]
```

```
pd1 unfiltered 10-90 time 2.0000007128100003e-09
pd2 unfiltered 10-90 time 2.2000001799999994e-09
pd1 filtered 10-90 time 2.1372996649880312e-09
pd2 filtered 10-90 time 2.1872949787887363e-09
pd1 peak halfmax: 4.657307705924974e-09
pd2 peak halfmax: 4.534987707140087e-09
```

[19]:
```
start    4.887042e-09
end      5.374496e-09
10       5.287004e-09
90       5.337000e-09
dtype: float64
```

```
[20]: res2 = clean_vis_9010_diff(pd1_cable, pd2_cable)
      res1 = clean_vis_9010_diff(pd1, pd2)
      (res2.iloc[1] - res2.iloc[0])-(res1.iloc[1] - res1.iloc[0])
```

```
pd1 unfiltered 10-90 time 2.000000079649e-09
pd2 unfiltered 10-90 time 2.400000155e-09
pd1 filtered 10-90 time 2.124800775971664e-09
pd2 filtered 10-90 time 2.2997843692869078e-09
pd1 peak halfmax: 4.671802877125571e-09
pd2 peak halfmax: 4.607934941149904e-09

pd1 unfiltered 10-90 time 2.0000000712810003e-09
pd2 unfiltered 10-90 time 2.2000001799999994e-09
pd1 filtered 10-90 time 2.1372996649880312e-09
pd2 filtered 10-90 time 2.1872949787887363e-09
pd1 peak halfmax: 4.657307705924974e-09
pd2 peak halfmax: 4.534987707140087e-09
```

```
[20]: start    4.937037e-09
      end      4.974533e-09
      10       4.837046e-09
      90       4.962035e-09
      dtype: float64
```

```
[21]: difdif = ((res2.iloc[1] - res2.iloc[0])-(res1.iloc[1] - res1.iloc[0]))
      'Added time due to the longer cable mean and standard deviation: ', difdif.
       ↪mean(), difdif.std()
```

```
[21]: ('Added time due to the longer cable mean and standard deviation: ',
       4.927662827207844e-09,
       6.23898963113883e-11)
```

```
[22]: 944.5e-3/difdif.mean()
```

```
[22]: 191673016.82756993
```

```
[23]: 2.9979e8/191673016
```

```
[23]: 1.5641220984387285
```

0.3 D

0.3.1 Air

Setup Distances

```
[67]: air_pd2_path = ufloat(917.5, .5) + ufloat(730.0, .5) + ufloat(124.5, .5)
      air_pd1_path = ufloat(168.5, .5)
      air_length_diff = air_pd2_path - air_pd1_path
      air_length_diff *= 1e-3 # convert to meters
```

```
air_length_diff
```

[67]:

```
1.6035+/-0.001
```

Loading Data

[68]:

```
[[pd1_air, air_config], [pd2_air, air_config]] = read_scope_csv(
    './Data/14_D_1_b.csv', 2)
pd1_air.loc[pd1_air.ts > 6e-9, 'mV'] = 0
pd2_air.loc[pd2_air.ts > 1.1e-8, 'mV'] = 0

fig = figure()

fig.line(pd1_air.ts, pd1_air.mV, legend_label='pd1_air')
fig.line(pd1_air.ts, pd2_air.mV, legend_label='pd2_air', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4961.80024, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4961.80024, 'Samples']}
```

Time Shift

[26]:

```
error_opt = pd.DataFrame(columns=['param', 'mean', 'std'])
error_opt.param = np.arange(.05, .55, .02)


def fun(row):
    filter_param = row.param
    air_firstpeak_times = clean_vis_9010_diff(
        pd1_air, pd2_air, False, filter_param=filter_param, plot=False)
    firstpeak_air_vals = (
        air_firstpeak_times.iloc[1] - air_firstpeak_times.iloc[0])
    row[['mean', 'std']] = firstpeak_air_vals.mean(), firstpeak_air_vals.std()
    return row


error_opt = error_opt.apply(fun, axis=1)

error_opt
```

```
pd1 unfiltered 10-90 time 2.000000007766e-09
pd2 unfiltered 10-90 time 1.8000001499999996e-09
pd1 filtered 10-90 time 1.974814857017404e-09
pd2 filtered 10-90 time 1.949817200599331e-09
pd1 peak halfmax: 3.1202149871265297e-09
```

```
pd2 peak halfmax: 3.081017802991106e-09
pd1 unfiltered 10-90 time 1.599999976308e-09
pd2 unfiltered 10-90 time 1.4000001199999998e-09
pd1 filtered 10-90 time 1.5998500107482124e-09
pd2 filtered 10-90 time 1.5623535261212086e-09
pd1 peak halfmax: 2.8100154600981597e-09
pd2 peak halfmax: 2.718143695874573e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.4000001199999998e-09
pd1 filtered 10-90 time 1.3873699311955434e-09
pd2 filtered 10-90 time 1.3623722747778938e-09
pd1 peak halfmax: 2.7584342225941204e-09
pd2 peak halfmax: 2.6391739908066693e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619419592e-09
pd2 filtered 10-90 time 1.287379305523886e-09
pd1 peak halfmax: 2.751124486571032e-09
pd2 peak halfmax: 2.620543501111289e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619417474e-09
pd2 filtered 10-90 time 1.2748804773149553e-09
pd1 peak halfmax: 2.7459418991498162e-09
pd2 peak halfmax: 2.6160336791853975e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619419592e-09
pd2 filtered 10-90 time 1.2623816491060246e-09
pd1 peak halfmax: 2.7882800931662e-09
pd2 peak halfmax: 2.6461592957817144e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619419592e-09
pd2 filtered 10-90 time 1.2623816491060246e-09
pd1 peak halfmax: 2.8173317645052374e-09
pd2 peak halfmax: 2.6872963144650088e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3248757901506781e-09
pd2 filtered 10-90 time 1.2748804773149553e-09
pd1 peak halfmax: 2.856695154297199e-09
pd2 peak halfmax: 2.6973459539524236e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598206e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8234394626022846e-09
```

```
pd2 peak halfmax: 2.700664908139148e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598206e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.821605704522518e-09
pd2 peak halfmax: 2.693528308255431e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3623722747778938e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8271424447106297e-09
pd2 peak halfmax: 2.704545525170345e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598206e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8408514711712463e-09
pd2 peak halfmax: 2.6980766890042853e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8286899369549205e-09
pd2 peak halfmax: 2.6922394669951524e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8355353306430353e-09
pd2 peak halfmax: 2.690408213742346e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8463970488954458e-09
pd2 peak halfmax: 2.6877242548282983e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.8458900032122955e-09
pd2 peak halfmax: 2.6803934083238696e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598206e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.8461330037299262e-09
```

```
pd2 peak halfmax: 2.681174837844569e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8443576837865465e-09
pd2 peak halfmax: 2.6804950555154816e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.8438794172768265e-09
pd2 peak halfmax: 2.6834961471524664e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2873793055240978e-09
pd1 peak halfmax: 2.83333078972053e-09
pd2 peak halfmax: 2.6881826477135116e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.815135585419767e-09
pd2 peak halfmax: 2.6838556423016014e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.8186555267747963e-09
pd2 peak halfmax: 2.6807484294659694e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.820209344716499e-09
pd2 peak halfmax: 2.6924678702173555e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.816104603899412e-09
pd2 peak halfmax: 2.6951823128023075e-09
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.816104603899412e-09
```

```
pd2 peak halfmax: 2.6951823128023075e-09
```

[26]:

```
    param          mean           std
0    0.05  5.265131e-09  2.771435e-11
1    0.07  5.283880e-09  3.441909e-11
2    0.09  5.283880e-09  2.771435e-11
3    0.11  5.287004e-09  4.207733e-11
4    0.13  5.283880e-09  3.441909e-11
5    0.15  5.283880e-09  3.441909e-11
6    0.17  5.299503e-09  4.786687e-11
7    0.19  5.421367e-09  2.778706e-10
8    0.21  5.383870e-09  2.114667e-10
9    0.23  5.355748e-09  1.556092e-10
10   0.25  5.393244e-09  2.312565e-10
11   0.27  5.424491e-09  2.924120e-10
12   0.29  5.343249e-09  1.324719e-10
13   0.31  5.337000e-09  1.357720e-10
14   0.33  5.321376e-09  9.755227e-11
15   0.35  5.333875e-09  1.204803e-10
16   0.37  5.333875e-09  1.115014e-10
17   0.39  5.324501e-09  1.035720e-10
18   0.41  5.318251e-09  9.156334e-11
19   0.43  5.321376e-09  9.034673e-11
20   0.45  5.308877e-09  6.403744e-11
21   0.47  5.321376e-09  8.059887e-11
22   0.49  5.321376e-09  7.385598e-11
23   0.51  5.315127e-09  6.238990e-11
24   0.53  5.315127e-09  6.238990e-11
```

[27]:

```
firstpeak_air_vals = (error_opt.iloc[1] - error_opt.iloc[0])
air_firstpeak_timeshift = ufloat(firstpeak_air_vals.mean(), firstpeak_air_vals.
 ↪std())
nu.print_unc(air_firstpeak_timeshift)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-aff279429905> in <module>
----> 1 firstpeak_air_vals = (air_firstpeak_times.iloc[1] - air_firstpeak_times
 ↪iloc[0])
      2 air_firstpeak_timeshift = ufloat(firstpeak_air_vals.mean(),␣
 ↪firstpeak_air_vals.std())
      3 nu.print_unc(air_firstpeak_timeshift)

NameError: name 'air_firstpeak_times' is not defined
```

[48]:
```
air_firstpeak_times = clean_vis_9010_diff(
    pd1_air, pd2_air, False, filter_param=.51, plot=True)
air_firstpeak_times
```

```
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687513e-09
pd2 filtered 10-90 time 1.2998781337330285e-09
pd1 peak halfmax: 2.816104603899412e-09
pd2 peak halfmax: 2.6951823128023075e-09
```

[48]:
```
             start           end            10            90
sig1 -8.047023e-11  2.394298e-09  5.944665e-10  1.944340e-09
sig2  5.319024e-09  7.706300e-09  5.893970e-09  7.193848e-09
```

[53]:
```
air_timeshift = error_opt.iloc[22, 1:]
air_timeshift = ufloat(air_timeshift['mean'], air_timeshift['std'])
air_timeshift
```

[53]:
```
5.3213761099689765e-09+/-7.385598377043428e-11
```

Refractive Index

[71]:
```
nu.print_unc(299792458/(air_length_diff/air_timeshift))
```

```
0.99 +- 0.01
```

[71]:
```
(0.99, 0.01, 2)
```

0.3.2 Glass

Cable Length

[72]:
```
FO_legth = ufloat(2065.5, .5)*1e-3
```

Loading Data

[31]:
```
[[pd1_glass, glass_config], [pd2_glass, glass_config]] = read_scope_csv(
    './Data/l4_D_2_b.csv', 2)

fig = figure()
pd1_glass.loc[pd1_glass.ts > 6e-9, 'mV'] = 0
pd2_glass.loc[pd2_glass.ts > 2.1e-8, 'mV'] = 0
fig.line(pd1_glass.ts, pd1_glass.mV, legend_label='pd1_glass')
fig.line(pd1_glass.ts, pd2_glass.mV, legend_label='pd2_glass', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4936.00015, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4936.00015, 'Samples']}
```

Timeshift

[62]:
```
glass_peak_times = clean_vis_9010_diff(pd1_glass, pd2_glass, False, 'fft', .2)

glass_timeshift = (glass_peak_times.iloc[1] - glass_peak_times.iloc[0])
glass_timeshift = ufloat(glass_timeshift.mean(), glass_timeshift.std())
glass_timeshift
```

```
pd1 unfiltered 10-90 time 1.40000003748e-09
pd2 unfiltered 10-90 time 1.4000001999999994e-09
pd1 filtered 10-90 time 1.374871090480383e-09
pd2 filtered 10-90 time 1.3123769500038768e-09
pd1 peak halfmax: 3.09791354391202e-09
pd2 peak halfmax: 2.951953305442787e-09
```

[62]:
```
1.5832890489690345e-08+/-6.442556879730129e-10
```

Refractive Index

[73]:
```
FO_shift = glass_timeshift - air_timeshift
```

[75]:
```
nu.print_unc(2.998e8/(FO_legth / FO_shift))
```

```
1.53 +- 0.09
```

[75]:
```
(1.53, 0.09, 2)
```

0.3.3 Water

Water length

[76]:
```
water_length = ufloat(614.0, .5)*1e-3
```

Loading Data

[36]:
```
[[pd1_water, water_config], [pd2_water, water_config]] = read_scope_csv(
    './Data/14_D_3_b.csv', 2)

pd1_water.loc[pd1_water.ts > 6e-9, 'mV'] = 0
pd2_water.loc[pd2_water.ts > 1.25e-8, 'mV'] = 0

fig = figure()
fig.line(pd1_water.ts, pd1_water.mV, legend_label='pd1_water')
fig.line(pd1_water.ts, pd2_water.mV, legend_label='pd2_water', color='red')
show(fig)
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4959.80049, 'Samples']}
```

```
{'record_length': [10000.0, 'Points'],
 'sample_interval': [2.0000000185e-10, 's'],
 'trigger_point': [4959.80049, 'Samples']}
```

Time Shift

[77]:
```
water_peak_times = clean_vis_9010_diff(pd1_water, pd2_water, False, 'fft', 1.6)

water_air_timeshift = (water_peak_times.iloc[1] - water_peak_times.iloc[0])
water_air_timeshift = ufloat(water_air_timeshift.mean(), water_air_timeshift.
 ↪std())
water_air_timeshift
```

```
pd1 unfiltered 10-90 time 1.19999996896e-09
pd2 unfiltered 10-90 time 1.4000001139999998e-09
pd1 filtered 10-90 time 1.3623723086395181e-09
pd2 filtered 10-90 time 1.3123769945609862e-09
pd1 peak halfmax: 2.9765729693274294e-09
pd2 peak halfmax: 2.8945223371951055e-09
```

[77]:
```
6.724369743559948e-09+/-5.400111111310354e-11
```

Refractive Index

$$T_A = time\ of\ flight\ diff\ air$$

$$T_W = time\ of\ flight\ in\ water$$

$$D_A = distance\ of\ air\ flight\ path$$

$$D_W = distance\ of\ water\ flight\ path$$

$$T_{A-W} = T_A \cdot \frac{D_A - D_W}{D_A}$$

$$T_W = T_A - T_{A-W}$$

[80]:
```
time_not_water = air_timeshift * (air_length_diff-2*water_length)/
 ↪air_length_diff
water_timeshift = water_air_timeshift - time_not_water

water_speed = 2*water_length/water_timeshift
(2.998e8/water_speed)
```

[80]:
```
1.3374388559231725+/-0.013860026931971947
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: