APL_4_cell

June 12, 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,
                              Legend)
    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]:
```
from bokeh.palettes import Category20_4 as colors
```

[3]:
```
from Helper.numbers import rising_edge
```

[4]:
```
from scipy.interpolate import interp1d
```

[5]:
```
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
```

[6]:
```
def clean_vis_9010_diff(pd1,
                        pd2,
                        first=False,
                        filt_type='fft',
                        filter_param=.25,
                        plot=True,
                        filename=''):
    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]]

# Zero time vector relative to pd1 peak start
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 * 1e9

print('pd1 filtered ', end='')
pd1_inds, pd1_90_10_inds = nu.rising_edge(pd1_clean, t, first)
t -= t[pd1_inds[0]]

print('pd2 filtered ', end='')
pd2_inds, pd2_90_10_inds = nu.rising_edge(pd2_clean, t, first)

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]))

    pd1 = fig.line(
        t[pd1_inds[0] - 100:pd2_inds[1] * 2],
        pd1_clean[pd1_inds[0] - 100:pd2_inds[1] * 2],
        line_width=4,
        #                    legend_label='PD1'
    )
    pd2 = fig.line(
        t[pd1_inds[0] - 100:pd2_inds[1] * 2],
        pd2_clean[pd1_inds[0] - 100:pd2_inds[1] * 2],
        line_width=4,
        #                    legend_label='PD2',
        color='red')

    #     fig.line(t[pd1_inds[0]:pd1_inds[1]], pd1_clean[pd1_inds[0]:
↪pd1_inds[1]], color = 'red', alpha = 1)
    #     fig.line(t[pd2_inds[0]:pd2_inds[1]], pd2_clean[pd2_inds[0]:
↪pd2_inds[1]], color = 'blue', alpha = 1)

    minmax = fig.scatter(
        t[pd1_inds],
        pd1_clean[pd1_inds],
        color='black',
        size = 15
        # legend_label='peak min/max'
    )
    ten_90 = fig.scatter(
        t[pd1_90_10_inds],
        pd1_clean[pd1_90_10_inds],
        color='grey',
        size = 15
        #                           legend_label='10-90 points'
    )
    minmax = fig.scatter(
        t[pd2_inds],
        pd2_clean[pd2_inds],
        color='black',
        size = 15
        #                           legend_label='peak min/max'
    )
    ten_90 = fig.scatter(
        t[pd2_90_10_inds],
```

```
        pd2_clean[pd2_90_10_inds],
        color='grey',
        size = 15
        #                           legend_label='10-90 points'
    )

    fig.x_range = Range1d(t[pd1_inds[0]] - .5, t[pd2_inds[1]] * 1.3)
    fig.y_range = Range1d(-.1*pd1_clean[pd1_inds[1]], pd1_clean[pd1_inds[1]] *␣
↪1.05)
    fig.add_layout(
        Legend(items=[("PD1", [pd1]), ("PD2", [pd2]),
                      ("10-90 points", [ten_90]), ("peak min/max", [minmax])],
               orientation='horizontal'), 'above')
    #     fig.add_layout(
    #         Legend(items=[("10-90 points", [ten_90]), ("peak min/max",␣
↪[minmax])],
    #                orientation='horizontal'), 'above')
    if plot:
        show(fig)
    else:
        del fig
    if filename:
        style(fig)
        export_png(fig, filename=filename)

    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

[7]:
```
[[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']}
```

[8]:
```
fig = figure()
fig.line(squares.ts, squares.mV)
show(fig)
```

[9]:
```
approx_amplitude = squares.mV.max() - squares.mV.min()
fourier_coefs = np.fft.fft(squares.mV, n=len(squares)*16)
fourier_freqs = np.fft.fftfreq(
    len(squares.mV)*16, 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'
```

[9]:
```
(5.5999997545, 'mV', 92187.50253515632, 'Hz')
```

[10]:
```
fig = figure()
fig.line(fourier_freqs, np.abs(fourier_coefs))
show(fig)
```

0.2 Initial Optics Setup

Failed Data

[11]:
```
[[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']}
```

[12]:
```
[[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

[13]:
```
[[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']}
```

[14]:
```
[[pd1_cable, pd1_config], [pd2_cable, pd2_config]] = read_scope_csv(
    './Data/l4_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

First pass

[15]:
```
fig = figure()
"""
Create a time vector beginning at zero for alignment purposes.
"""
t = pd1.loc[pd1.ts >= 0, 'ts'] * 1e9
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%.
"""
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.xaxis.axis_label = 'time (ns)'
fig.yaxis.axis_label = 'mV'
fig.x_range = Range1d(-.4, 30)
style(fig)
show(fig)
export_png(fig, filename='l4_A.png')
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 1.999999718999999e-09
10-90 time 2.00000005299
10-90 time 2.0000000287189996
10-90 time 2.400000200000001
10-90 time 1.999999722
```

[16]:
```
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()
```

[16]:
```
(5.10000017006775, 0.20000017053124855)
```

Second Pass

[19]:
```
res = clean_vis_9010_diff(pd1, pd2, filename = '../Images/l4_C_a.png')
res.iloc[1] - res.iloc[0]
```

```
pd1 unfiltered 10-90 time 2.0000000712810003e-09
pd2 unfiltered 10-90 time 2.2000001799999994e-09
pd1 filtered 10-90 time 2.1372996649880314
pd2 filtered 10-90 time 2.187294978788736
pd1 peak halfmax: 4.6573077059290116
pd2 peak halfmax: 4.534987707144018
```

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

[20]:
```
res2 = clean_vis_9010_diff(pd1_cable, pd2_cable, filename = '../Images/l4_C_a.
 ↪png')
res1 = clean_vis_9010_diff(pd1, pd2, filename = '../Images/l4_C_b.png')
print('With extra cable time measurements')
print(np.round(res1, 2).to_latex())
print('Without extra cable time measurements')
print(np.round(res2, 2).to_latex())
```

```
pd1 unfiltered 10-90 time 2.000000079649e-09
pd2 unfiltered 10-90 time 2.400000155e-09
pd1 filtered 10-90 time 2.124800775971664
pd2 filtered 10-90 time 2.2997843692869075
pd1 peak halfmax: 4.671802877128309
pd2 peak halfmax: 4.607934941152604

pd1 unfiltered 10-90 time 2.0000000712810003e-09
pd2 unfiltered 10-90 time 2.2000001799999994e-09
pd1 filtered 10-90 time 2.1372996649880314
pd2 filtered 10-90 time 2.187294978788736
pd1 peak halfmax: 4.6573077059290116
pd2 peak halfmax: 4.534987707144018

With extra cable time measurements
\begin{tabular}{lrrrr}
\toprule
{} &  start &   end &    10 &    90 \\
\midrule
sig1 &   0.00 &  3.77 &  0.60 &  2.74 \\
sig2 &   4.89 &  9.15 &  5.89 &  8.07 \\
\bottomrule
\end{tabular}

Without extra cable time measurements
\begin{tabular}{lrrrr}
\toprule
{} &  start &    end &     10 &     90 \\
\midrule
sig1 &   0.00 &   4.04 &   0.89 &   3.01 \\
sig2 &   9.82 &  14.39 &  11.01 &  13.31 \\
\bottomrule
\end{tabular}
```

[21]:
```
dif = pd.DataFrame([(res1.iloc[1] - res1.iloc[0]),
                    (res2.iloc[1] - res2.iloc[0])],
                   index=['Same Cable', 'Different Cable'])
print(pd.DataFrame([dif.mean(1), dif.std(1)], index = ['value','std']).T.
 ↪to_latex())
```

```
difdif = dif.iloc[1] - dif.iloc[0]
difdif_val = ufloat(difdif.mean(), difdif.std())
'Added time due to the longer cable mean and standard deviation: ', nu.
 ↪print_unc(difdif_val)
```

```
\begin{tabular}{lrr}
\toprule
{} &      value &       std \\
\midrule
Same Cable      &   5.221386 &  0.225759 \\
Different Cable &  10.149048 &  0.237149 \\
\bottomrule
\end{tabular}

4.93 +- 0.06
```

[21]:
```
('Added time due to the longer cable mean and standard deviation: ',
 (4.93, 0.06, 2))
```

[22]:
```
cable_sol = 944.5e-3/difdif_val*10
nu.print_unc(cable_sol)[:2], '1e8 m/s'
```

```
1.92 +- 0.02
```

[22]:
```
((1.92, 0.02), '1e8 m/s')
```

[23]:
```
nu.print_unc(2.9979/cable_sol)[:2], 'effective refractive index'
```

```
1.56 +- 0.02
```

[23]:
```
((1.56, 0.02), 'effective refractive index')
```

0.3 D

0.3.1 Air

Setup Distances

[24]:
```
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
```

[24]:
```
1.6035+/-0.001
```

Loading Data

[25]:
```
[[pd1_air, air_config], [pd2_air, air_config]] = read_scope_csv(
    './Data/l4_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 1.9748148570174042
pd2 filtered 10-90 time 1.9498172005993304
pd1 peak halfmax: 3.1202149871449234
pd2 peak halfmax: 3.081017803009269
pd1 unfiltered 10-90 time 1.599999976308e-09
pd2 unfiltered 10-90 time 1.4000001199999998e-09
pd1 filtered 10-90 time 1.5998500107482125
pd2 filtered 10-90 time 1.5623535261212087
pd1 peak halfmax: 2.810015460089166
pd2 peak halfmax: 2.7181436958658733
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.4000001199999998e-09
pd1 filtered 10-90 time 1.3873699311955434
```

```
pd2 filtered 10-90 time 1.3623722747778944
pd1 peak halfmax: 2.758434222585292
pd2 peak halfmax: 2.639173990798222
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619419592
pd2 filtered 10-90 time 1.2873793055238858
pd1 peak halfmax: 2.751124486562226
pd2 peak halfmax: 2.6205435011029015
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619417474
pd2 filtered 10-90 time 1.2748804773149551
pd1 peak halfmax: 2.745941899166004
pd2 peak halfmax: 2.6160336792008194
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.312376961941959
pd2 filtered 10-90 time 1.2623816491060245
pd1 peak halfmax: 2.7882800931826375
pd2 peak halfmax: 2.6461592957973137
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3123769619419594
pd2 filtered 10-90 time 1.2623816491060245
pd1 peak halfmax: 2.817331764521846
pd2 peak halfmax: 2.6872963144808506
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.324875790150678
pd2 filtered 10-90 time 1.2748804773149551
pd1 peak halfmax: 2.8566951542880554
pd2 peak halfmax: 2.69734595394379
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598207
pd2 filtered 10-90 time 1.287379305524098
pd1 peak halfmax: 2.8234394625932477
pd2 peak halfmax: 2.7006649081305034
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598207
pd2 filtered 10-90 time 1.287379305524098
pd1 peak halfmax: 2.821605704513487
pd2 peak halfmax: 2.6935283082468096
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3623722747778937
```

```
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.8271424447015807
pd2 peak halfmax: 2.7045455251616883
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598205
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.8408514711621535
pd2 peak halfmax: 2.6980766889956493
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.8286899369458665
pd2 peak halfmax: 2.6922394669865355
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.83555353063396
pd2 peak halfmax: 2.6904082137337344
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.8463970488863355
pd2 peak halfmax: 2.687724254819696
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.8458900032031864
pd2 peak halfmax: 2.6803934083152905
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3373746183598203
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.8461330037208166
pd2 peak halfmax: 2.681174837835987
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.8443576837774427
pd2 peak halfmax: 2.6804950555069023
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
```

```
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.843879417267724
pd2 peak halfmax: 2.6834961471438774
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2873793055240972
pd1 peak halfmax: 2.8333307897114612
pd2 peak halfmax: 2.688182647704908
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687514
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.815135585436362
pd2 peak halfmax: 2.683855642317423
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687514
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.818655526791413
pd2 peak halfmax: 2.6807484294817727
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.349873446568751
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.8202093447074725
pd2 peak halfmax: 2.6924678702087377
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687514
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.8161046039160134
pd2 peak halfmax: 2.6951823128181958
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687514
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.8161046039160134
pd2 peak halfmax: 2.6951823128181958
```

[26]:

	param	mean	std
0	0.05	5.265131	0.027714
1	0.07	5.283880	0.034419
2	0.09	5.283880	0.027714
3	0.11	5.287004	0.042077
4	0.13	5.283880	0.034419
5	0.15	5.283880	0.034419

```
6    0.17  5.299503  0.047867
7    0.19  5.421367  0.277871
8    0.21  5.383870  0.211467
9    0.23  5.355748  0.155609
10   0.25  5.393244  0.231256
11   0.27  5.424491  0.292412
12   0.29  5.343249  0.132472
13   0.31  5.337000  0.135772
14   0.33  5.321376  0.097552
15   0.35  5.333875  0.120480
16   0.37  5.333875  0.111501
17   0.39  5.324501  0.103572
18   0.41  5.318251  0.091563
19   0.43  5.321376  0.090347
20   0.45  5.308877  0.064037
21   0.47  5.321376  0.080599
22   0.49  5.321376  0.073856
23   0.51  5.315127  0.062390
24   0.53  5.315127  0.062390
```

[27]:
```
air_firstpeak_times = clean_vis_9010_diff(
    pd1_air, pd2_air, False, filter_param=.51, plot=True, filename = '../Images/
 ↪l4_D_a.png')
print(np.round(air_firstpeak_times, 2).to_latex())
```

```
pd1 unfiltered 10-90 time 1.39999996058e-09
pd2 unfiltered 10-90 time 1.2000001e-09
pd1 filtered 10-90 time 1.3498734465687514
pd2 filtered 10-90 time 1.2998781337330279
pd1 peak halfmax: 2.8161046039160134
pd2 peak halfmax: 2.6951823128181958
```

```
\begin{tabular}{lrrrr}
\toprule
{} &  start &   end &    10 &    90 \\
\midrule
sig1 &    0.0 &  2.47 &  0.67 &  2.02 \\
sig2 &    5.4 &  7.79 &  5.97 &  7.27 \\
\bottomrule
\end{tabular}
```

[28]:
```
air_timeshift = error_opt.iloc[22, 1:]
air_timeshift = ufloat(air_timeshift['mean'], air_timeshift['std'])
nu.print_unc(air_timeshift)[:2], 'air timeshift in ns'
```

```
5.32 +- 0.07
```

[28]:
```
((5.32, 0.07), 'air timeshift in ns')
```

[29]:
```
air_sol = air_length_diff/air_timeshift
nu.print_unc(air_sol)[:2], '1e8 m/s air SoL'
```

```
0.301 +- 0.004
```

[29]:
```
((0.301, 0.004), '1e8 m/s air SoL')
```

Refractive Index

[30]:
```
nu.print_unc(.299792458/air_sol)[:2], 'air refractive index'
```

```
0.99 +- 0.01
```

[30]:
```
((0.99, 0.01), 'air refractive index')
```

0.3.2 Glass

Cable Length

[31]:
```
FO_legth = ufloat(2065.5, .5)*1e-3
```

Loading Data

[32]:
```
[[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

[33]:
```
glass_peak_times = clean_vis_9010_diff(pd1_glass, pd2_glass, False, 'fft', .6,␣
 ↪filename = '../Images/l4_D_b.png')

print(np.round(glass_peak_times, 2).to_latex())
glass_timeshift = (glass_peak_times.iloc[1] - glass_peak_times.iloc[0])
glass_timeshift = ufloat(glass_timeshift.mean(), glass_timeshift.std())
nu.print_unc(glass_timeshift)[:2], 'ns glass time shift'
```

```
pd1 unfiltered 10-90 time 1.40000003748e-09
pd2 unfiltered 10-90 time 1.2000000999999992e-09
pd1 filtered 10-90 time 1.3623722623851662
```

```
pd2 filtered 10-90 time 1.2998781219086588
pd1 peak halfmax: 2.9804844136337993
pd2 peak halfmax: 2.8430695099190104

\begin{tabular}{lrrrr}
\toprule
{} &  start &    end &     10 &     90 \\
\midrule
sig1 &   0.00 &   2.59 &   0.76 &   2.12 \\
sig2 &  15.81 &  18.06 &  16.30 &  17.60 \\
\bottomrule
\end{tabular}

15.6 +- 0.2
```

[33]:
```
((15.6, 0.2), 'ns glass time shift')
```

Refractive Index

[34]:
```
FO_shift = glass_timeshift - air_timeshift
```

[35]:
```
nu.print_unc(FO_shift)[:2], 'ns glass vs air time shift'
```

```
10.3 +- 0.2
```

[35]:
```
((10.3, 0.2), 'ns glass vs air time shift')
```

[36]:
```
glass_sol = FO_legth / FO_shift*10
nu.print_unc(glass_sol)[:2], '1e8 m/s SoL in FO cable'
```

```
2.01 +- 0.03
```

[36]:
```
((2.01, 0.03), '1e8 m/s SoL in FO cable')
```

[37]:
```
nu.print_unc(2.9979/glass_sol)[:2], 'refractive index of glass'
```

```
1.49 +- 0.03
```

[37]:
```
((1.49, 0.03), 'refractive index of glass')
```

0.3.3 Water

Water length

[38]:
```
water_length = ufloat(614.0, .5)*1e-3
```

Loading Data

[39]:
```
[[pd1_water, water_config], [pd2_water, water_config]] = read_scope_csv(
    './Data/l4_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

[40]:
```
water_peak_times = clean_vis_9010_diff(pd1_water, pd2_water, False, 'fft', 1.6,␣
 ↪filename = '../Images/l4_D_c.png')

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 1.362372308639518
pd2 filtered 10-90 time 1.312376994560986
pd1 peak halfmax: 2.9765729693314276
pd2 peak halfmax: 2.8945223371989934
```

[40]:
```
6.724369743559947+/-0.054001111113103815
```

[41]:
```
print(np.round(water_peak_times, 2).to_latex())
```

```
\begin{tabular}{lrrrr}
\toprule
{} &  start &   end &    10 &    90 \\
\midrule
sig1 &    0.0 &  2.55 &  0.71 &  2.07 \\
sig2 &    6.8 &  9.25 &  7.44 &  8.75 \\
\bottomrule
\end{tabular}
```

[42]:
```
nu.print_unc(water_air_timeshift)[:2], 'ns air and water timeshift'
```

```
6.72 +- 0.05
```

[42]:
```
((6.72, 0.05), 'ns air and water timeshift')
```

Refractive Index

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

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

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

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

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

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

[43]:
```
time_not_water = air_timeshift * (air_length_diff-2*water_length)/
 ↪air_length_diff
water_timeshift = water_air_timeshift - time_not_water
print(nu.print_unc(water_timeshift)[:2], 'ns timeshift water only')

water_speed = 2*water_length/water_timeshift
print(nu.print_unc(water_speed*10)[:2], 'm/s SoL in water')
nu.print_unc(.29979/water_speed)[:2], 'refractive index of water'
```

```
5.48 +- 0.06
(5.48, 0.06) ns timeshift water only
2.24 +- 0.02
(2.24, 0.02) m/s SoL in water
1.34 +- 0.01
```

[43]:
```
((1.34, 0.01), 'refractive index of water')
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: