Laser Cooling and Trapping

February 27, 2018

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import scipy.optimize as optimize
        from scipy.optimize import curve_fit
        from scipy import signal

        def log_growth(t, V_0, tau, t_0, V_offset):
            return V_0*(1-np.exp(-(t - t_0)/tau)) + V_offset
```

1 13 Feb 2017

SETUP

Going through equipment manuals and ensuring proper setup of equipment * Removed RF output to RF input cable between laser controller and laser servo. The equipment manual states that this is for peak lock only. * Connected Temp Servo input/output as per recommendation of manual for greater long term stability. * Tuned lasers using controller until trapping laser is on redshifted side of ^{87}Rb D_2 $5^2P_{3/2}$ $F' = 3$ line, and pump was on redshifted side of ^{87}Rb D_1 $5^2P_{3/2}$ $F' = 2$ line. * Locked lasers using the servo by adjusting the DC offset until the desired portion of each line was at $0\,V$, and then locked the lasers. * Confirmed the lasers are not hopping by ensuring the LD OUT line remained less than $50\,mV$.

2 15 Feb 2017

SETUP * Performed steps mentioned in previous section * We adjusted the mirrors on the table to ensure the beams all crossed in the center of the chamber. Because the beams are expanded, it is not as difficult to adjust, but we made sure the return beam reflected directly backwards as much as possible by checking for laser light around the perimeter of the primary director. * This Cold Quanta MOT has a different back panel than the manual. KEEP ION PUMP ON. We adjusted the dispenser current to approximately $3.3\,mA$ in keeping with the Cold Quanta manual. However, there are no controls for the coils. * We found a power supply controlling the coils, and turned it on. We adjusted the current and voltage of the power supply for the coils to $1\,V$ and $1\,mA$. * After turning on the dispenser for approximately 30 s, and toggling the polarization of the trapping beams, we noticed a glowing light on the CCTV observing the trap. * We blocked one of the trapping beams and noticed the light went away. After removing the block, the light came back over the course of several seconds. Jun suggested that this is a way to determine the stability of

our trap. * by adjusting the current and voltage of the coil power supply, we noticed that as we lowered these values, the ions flattened out perpindicular to the coil axis. * Over the next few days, we will devise several experiments and observations to make so we can have fun with this experiment * Is there a quantitative way to measure the brightness of the trap? Doing so could allow us to measure the filling time of the trap

3 20 Feb 2017

SETUP * We performed the same setup as before and immediately obtained a MOT. We will now build an imaging system to collect the light from the trapped atoms and measure the intensity of this light vs time. * Our MOT is $\approx$ 5.1 cm from the nearest we can place the lens. We use the below program to determine the optimal placement of lens and collector in the $\approx$ 220 mm available to us. * We found a 25 mm focal length lens, which should optimize the location of the image to the space constraints of our table.

```
In [3]: def lenseq(f,xo):
            """
            Given a lens focal length and location (giving xo), returns the image location xi
            """
            return xo*f/(xo-f)

        print(lenseq(25, 51))
```

```
49.03846153846154
```

- So our image should be approximately 5 cm from the lens.

4 22 Feb 2017

- It is difficult to obtain data from our photodetector, so we placed a camera in the place of the detector to optically inspect the what the photodetector is 'seeing'.
- We noticed that there is a great deal of light coming directly from the mirrors. These mirrors have a reasonably large angular separation from the MOT, so we need to place a tube arount the detector to prevent these sources from entering the detector. We MacGyver'd a thin cardboard package and rolled it into a cylinder to achieve this blocking.
- Furthermore, we noticed that there is a significant amount of light from the laser scattering off the walls of the rubidium cell. We need something to limit the 'close-angle' light from entering the detector. We found an iris that we place close to the lens so we can limit the amount of scattering light that can enter the detector.
- It is difficult to 'aim' the photodetector to make sure it is 'seeing' the MOT. It took a great deal of adjustment to get the detector centered on the image.
  - In order to make sure we were seeing only the MOT, we would adjust the position with the MOT on, all while watching the signal from the detector.
  - Once we found a maximum, we would block all beams from entering the MOT area to ensure that the signal dropped, as the MOT should fail and so should any other laser light sources.

  - We then waited for the MOT to reform (on the camera), and then shut off the coils. This will make the MOT fail, but any other laser light should still be incident upon the detector.
  - If we see the signal drop after turning off the anit-Helmholz coils, and then slowly build back up, we know we are imaging the MOT. If no change, then we are primarily imaging the scattered light passing through the lens, and we know the detector is not on the line defined by the MOT and the center of the lens.
  - We readjusted the position of the detector until we obtained the signal behavior we desired.

- We can now take data from the oscilloscope.

- We are limited in the amount of time we have to perform experiments on the MOT, so we choose to characterize the population of the trap by determining the characteristic fill time for the MOT. By cycling the power on the coils, we can make the MOT fail, and then immediately begin repopulating. We expect this curve to have the form of a logistic growth curve, per lab manual equation (1):

$$N(t) = N_0 \left(1 - e^{-t/\tau}\right)$$

  Since the population should be proportional to the intensity of the light from the trap, the voltage from the photodetector should follow the logistic growth curve:

$$V(t) = V_0 \left(1 - e^{-t/\tau}\right)$$

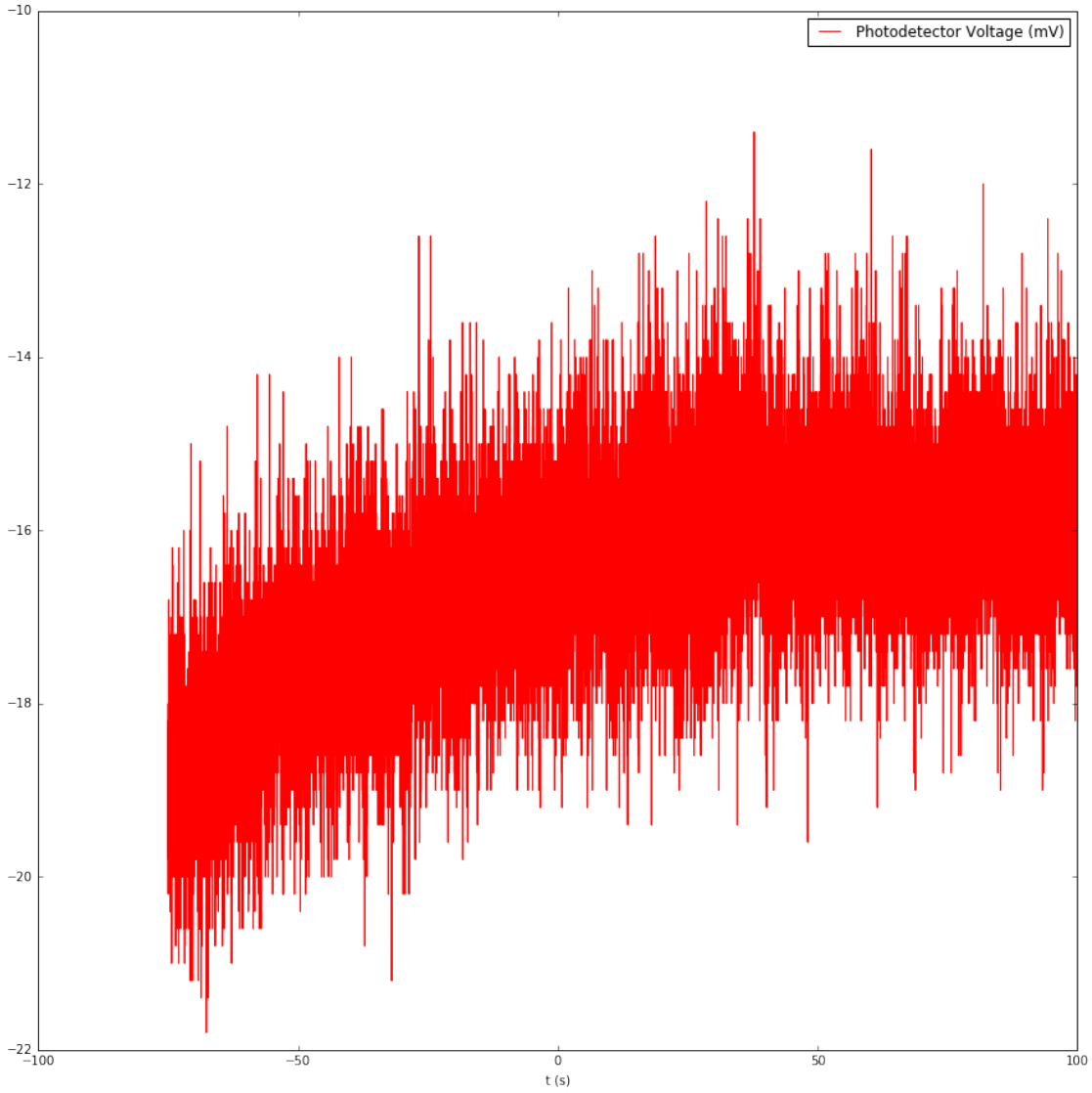
- We perform the curve fitting below

```
In [4]: class MOT_data:
            """
            Given a datafile, produces a curve of best fit to a logistic curve.
            The class automatically parses the data and creates a raw plot of the data.
            Methods include using a low pass digital filter on the data, fitting the data with v
            and plotting the refined data.
            """

            def __init__(self, filepath, x_units, y_units, x_mult_factor, y_mult_factor, title):

                self.data = pd.read_csv(filepath, skiprows = 21, names = [x_units, y_units])

                self.x_units = x_units
                self.y_units = y_units
                self.x_mult_factor = x_mult_factor
                self.y_mult_factor = y_mult_factor
                self.title = title

                self.data[self.data.columns[0]] *= self.x_mult_factor
                self.data[self.data.columns[1]] *= self.y_mult_factor

            def plot_data(self, plotdata_list, color_list=['r','r','b','r', 'g', 'r', 'c','r', '
                self.fig = plt.figure()
```

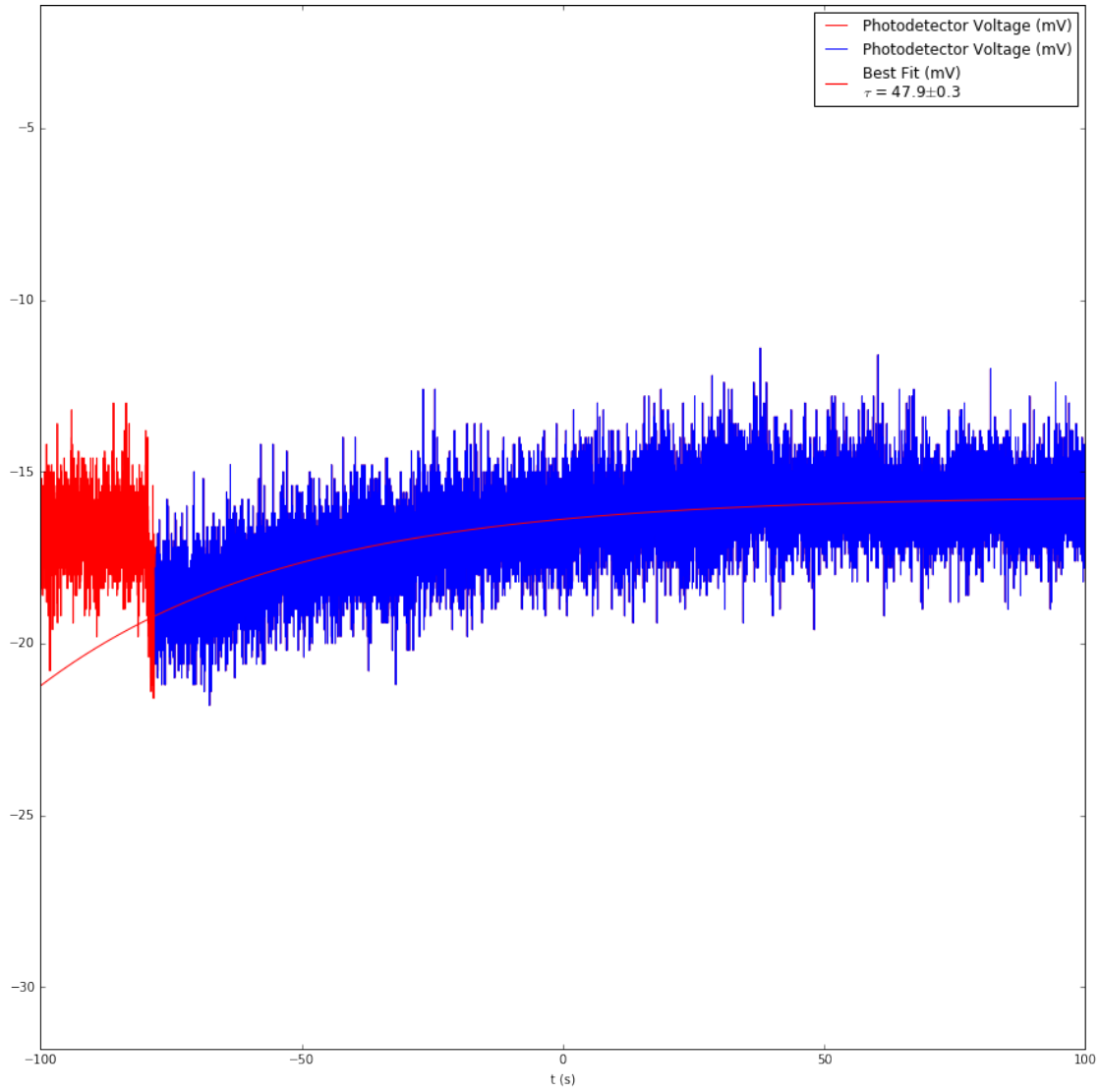
```
        self.ax = plt.subplot(111)
        for i in np.arange(len(plotdata_list)):
            plotdata_list[i].plot(kind = 'line', x = plotdata_list[i].columns[0], y = pl
                                  ax = self.ax, label = plotdata_list[i].columns[1], col
        self.ax.set_xlim(self.data[self.data.columns[0]].min(), self.data[self.data.colu
        self.ax.legend(loc = 'best')
        self.fig.set_figheight(15)
        self.fig.set_figwidth(15)
        if set_lim:
            self.ax.set_ylim(self.data[self.data.columns[1]].min()-10, self.data[self.da
        plt.show()

    def zoom(self, dataset, lbound, rbound):
        asarray = np.array(dataset['t (s)'])
        l_idx = np.searchsorted(asarray,lbound)
        r_idx = np.searchsorted(asarray,rbound)

        return dataset[l_idx:r_idx]


    def filter_data_butter(self, butterworth_order, cutoff_freq, pad):
        filtered_data = self.data.copy()
        b, a = signal.butter(butterworth_order, cutoff_freq)
        filtered_data[filtered_data.columns[1]] = signal.filtfilt(b,a, filtered_data[fil
                                                                  padlen = pad)
        return filtered_data

    def filter_data_gust(self, ellip_order, max_ripple, min_attenuation, cutoff_freq):
        filtered_data = self.data.copy()
        b, a = signal.ellip(ellip_order, max_ripple, min_attenuation, cutoff_freq)
        filtered_data[filtered_data.columns[1]] = signal.filtfilt(b,a, filtered_data[fil
                                                                  method = "gust")
        return filtered_data

    def logistic_fit(self, data, initial_guess):
        """
        Fits data to the curve V(t) = V_0 * (1-np.exp(-(t-t_0)/tau)) + V_offset
        Requires intial guess to be of the form [V_0, tau, t_0, V_offset]
        data must have two columns: first column is time, second in voltage
        """
        best_vals, covar = curve_fit(log_growth, data[data.columns[0]], data[data.column
        dTau = best_vals[1]
        errTau = np.sqrt(np.diag(covar))[1]

        return best_vals, covar

    def make_bestfit(self, function, x_data, fit_vals, val_err):
        report_Tau, report_Err = self.reported_values(fit_vals[1], np.sqrt(np.diag(val_e
```

```
            xmin = self.data[self.data.columns[0]].min()
            xmax = self.data[self.data.columns[0]].max()
            x_data = np.linspace(xmin, xmax, 1000)
        #         bestfit_data = pd.DataFrame([[x_data] , [function(x_data, *fit_vals)],\
        #                                      columns = [self.data.columns[0], 'Best Fit (mV)'])
            bestfit_data = pd.DataFrame({self.data.columns[0]:x_data})
            bestfit_data[ 'Best Fit (mV)\n$ \\tau $ = {}$\pm${}'.format(report_Tau, report_E

            return bestfit_data


        def reported_values(self, value, error):
            a = int(np.floor(np.log10(np.abs(error))))
            report_Err = round(error, -a)
            report_Val = round(value, -a)

            return report_Val, report_Err

        def best_w_data(self, data_list, best_guess_list, color_list = ['r','b','r', 'g', 'r
            """
            Given a list of data and the associated best guesses for the fit,\
            plots the curves and best fits for those plots
            """
            data_w_best = []
        #         data_w_best = [test.data]
            for i in np.arange(len(data_list)):
                data_w_best.append(data_list[i])
                if i != 0:
                    best_vals, val_err = self.logistic_fit(data_list[i], best_guess_list[i])
                    data_w_best.append(self.make_bestfit(log_growth, data_list[i][data_list[

            self.plot_data(data_w_best,color_list, set_lim=True)
```

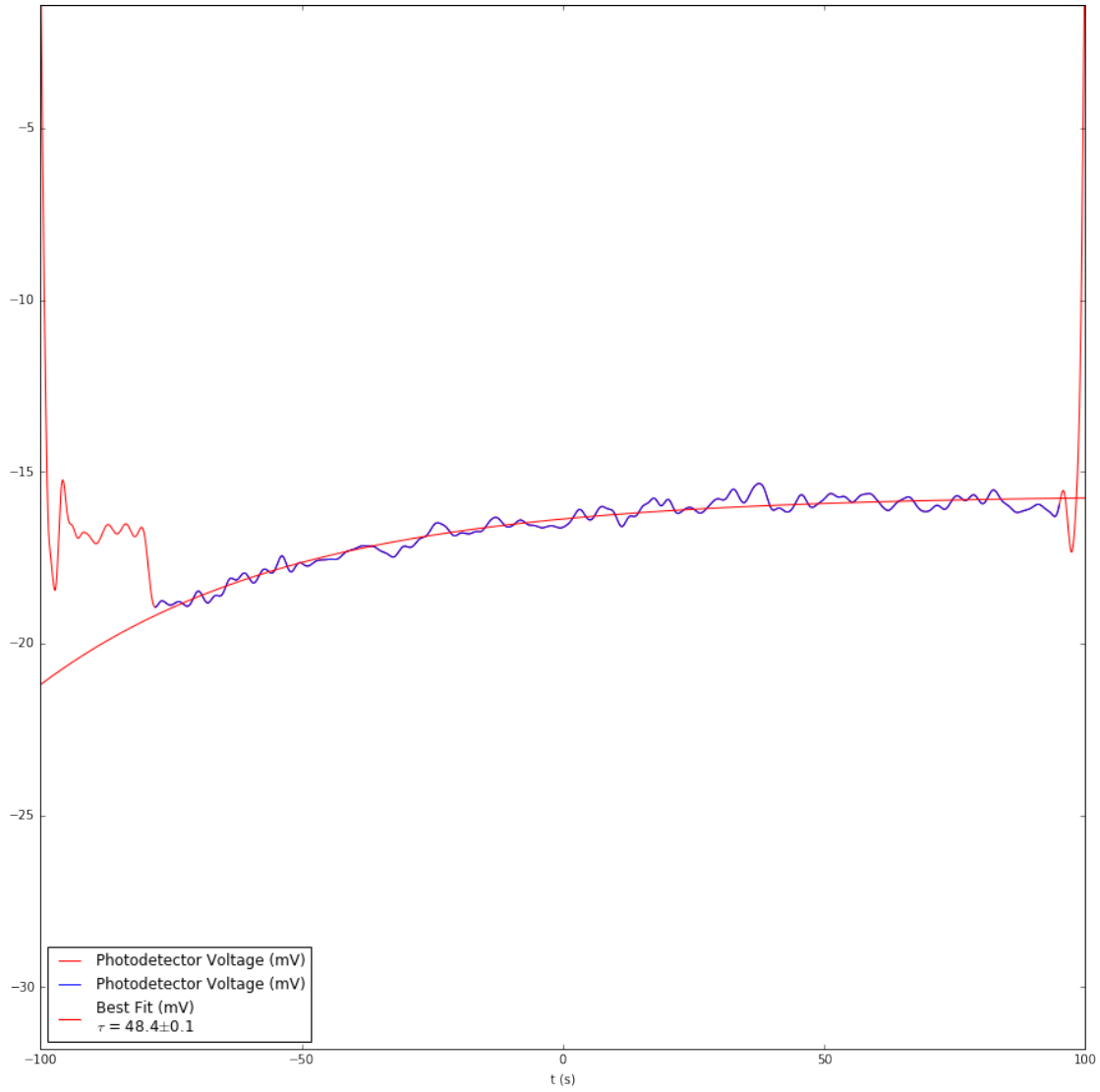
In [5]:
```
test0 = MOT_data('./Data/22FEB2018/tek0000CH3.csv', 't (s)', 'Photodetector Voltage (mV)
# test0.plot_data([test0.data])
test0.plot_data([test0.zoom(test0.data,-75,100)])
```



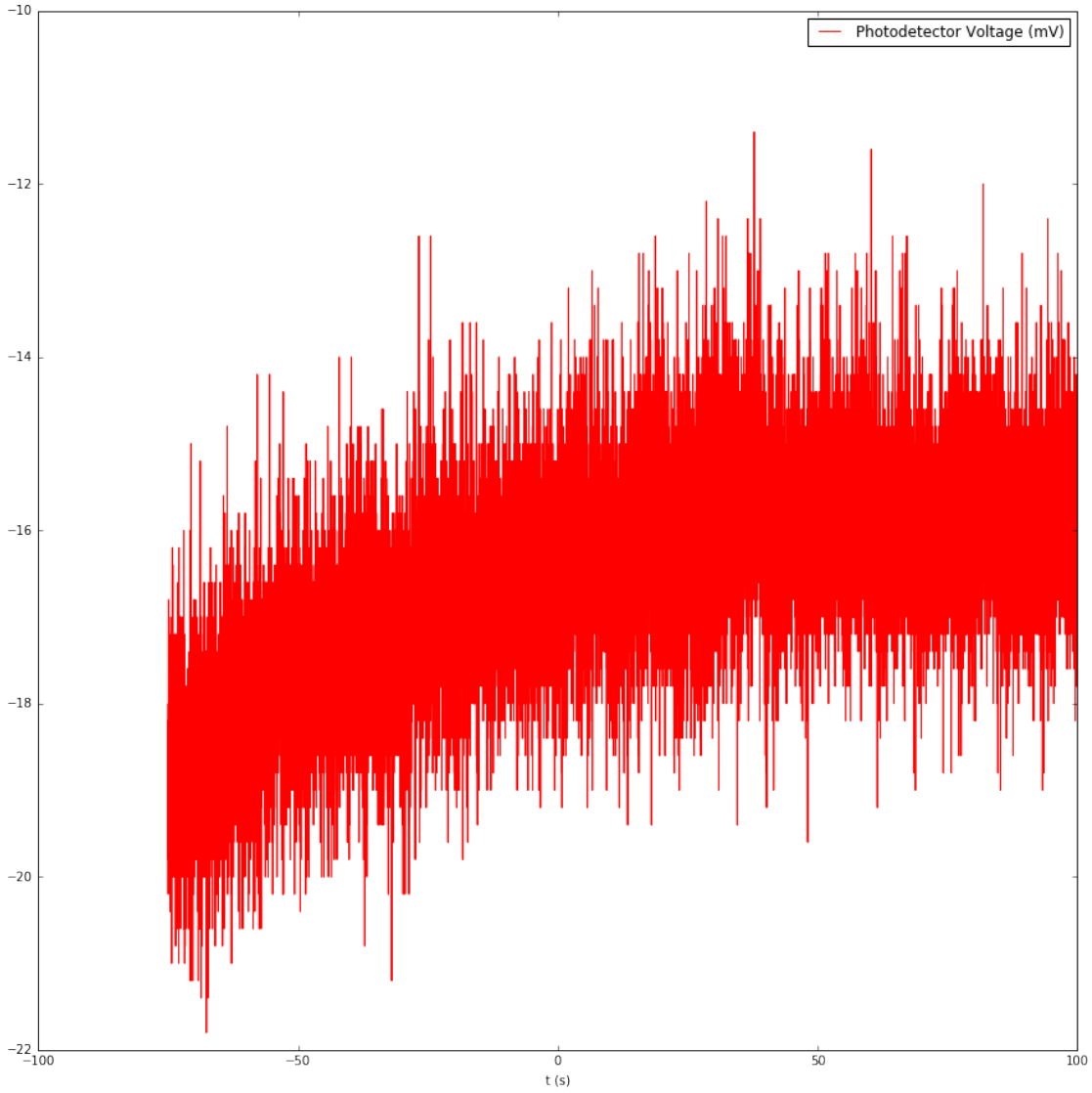

```
In [6]: test0.best_w_data([test0.data, test0.zoom(test0.data, -78,100)] ,\
                          [[],[3, 75., -80.,-20]])
```



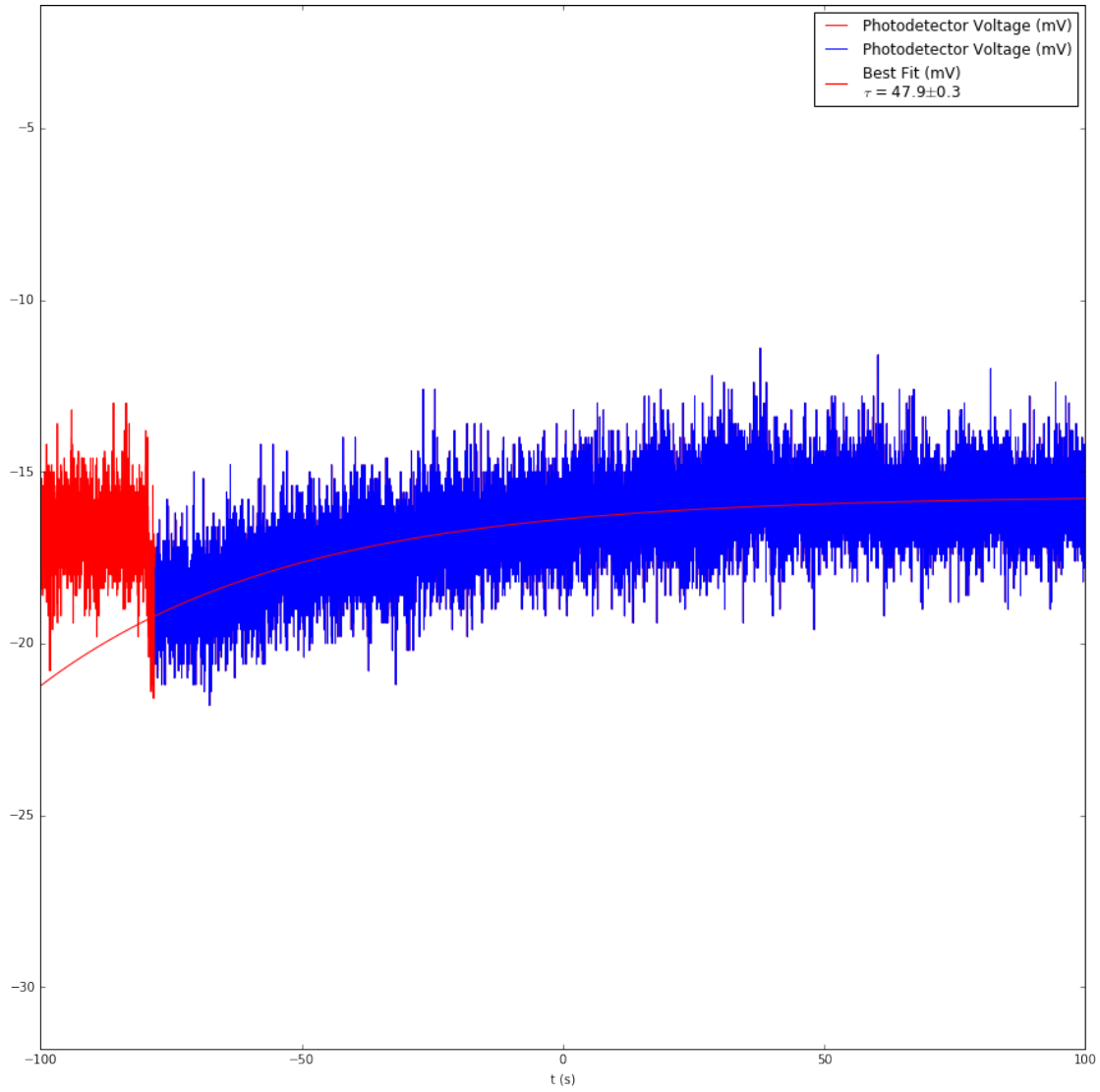

```
In [7]: test0.best_w_data([test0.filter_data_gust(5,0.001,120,.001),test0.zoom(test0.filter_data
                           [[],[3, 75., -80.,-20]])
```



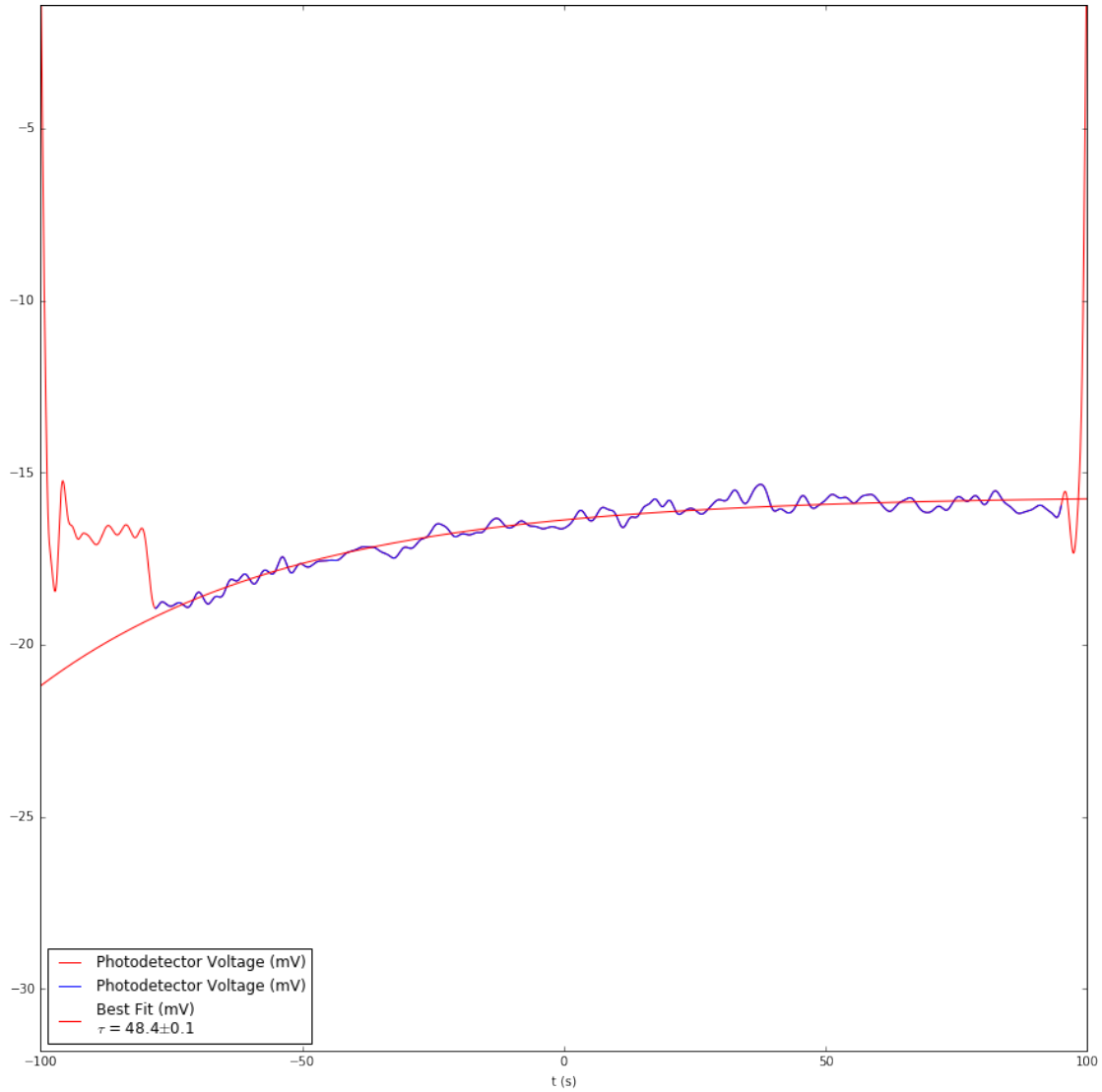

```
In [8]: test1 = MOT_data('./Data/22FEB2018/tek0001CH3.csv', 't (s)', 'Photodetector Voltage (mV)
        # test1.plot_data([test1.data])
        test1.plot_data([test1.zoom(test1.data,-75,100)])
```



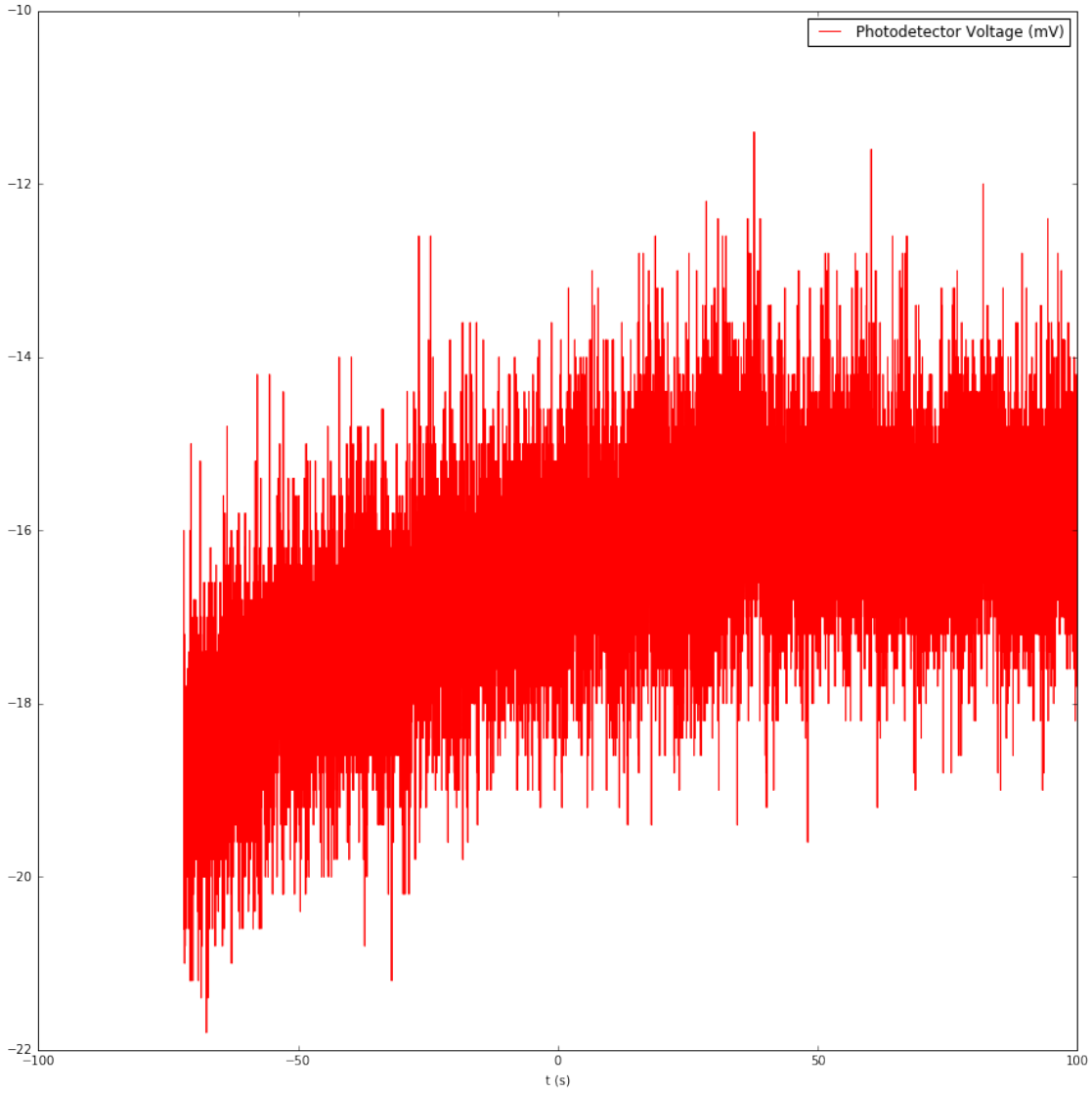

```
In [9]: test1.best_w_data([test1.data, test1.zoom(test1.data, -78,100)] ,\
                          [[],[3, 75., -80.,-20]])
```



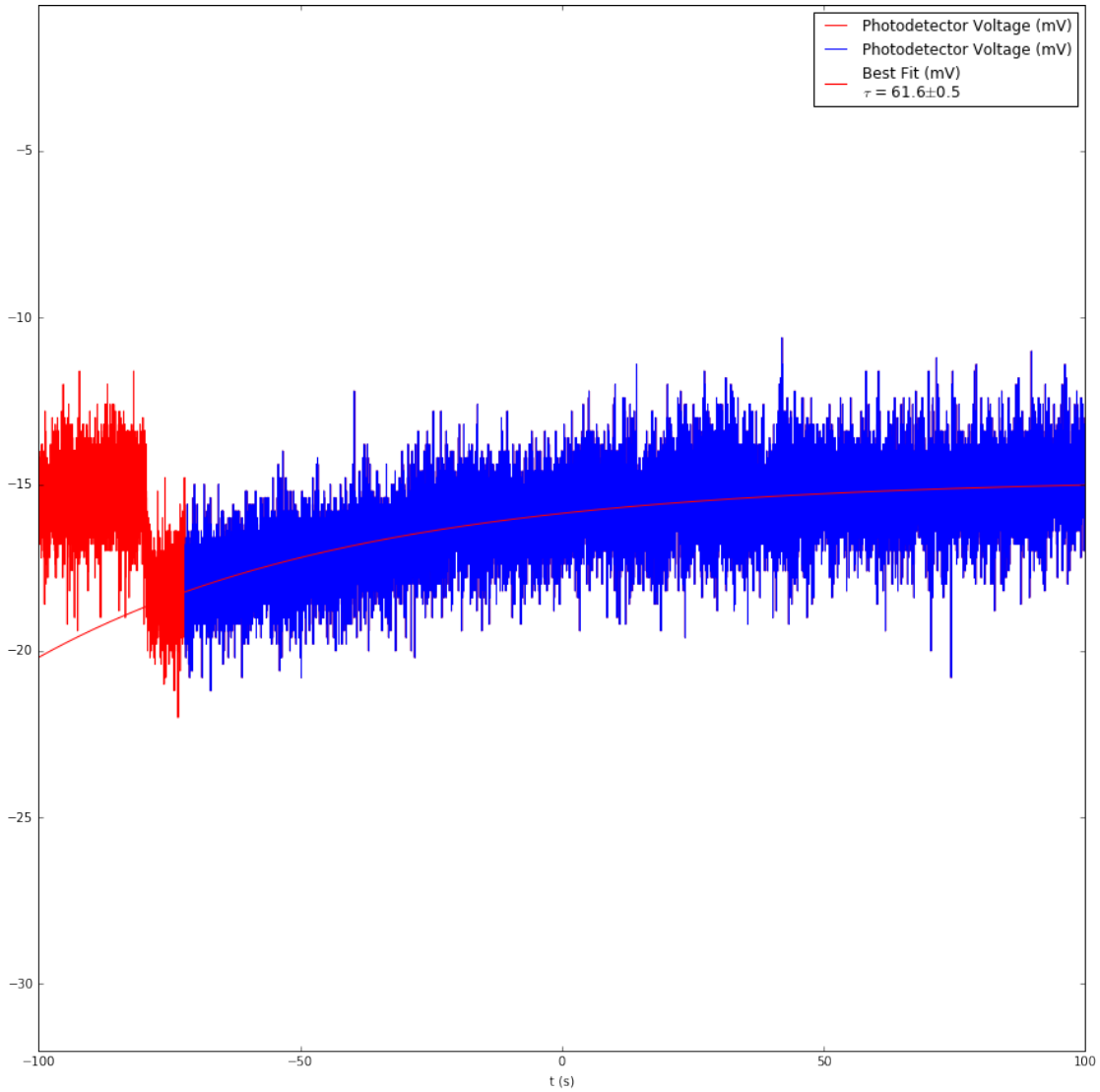

```
In [10]: test1.best_w_data([test1.filter_data_gust(5,0.001,120,.001),test1.zoom(test1.filter_dat
                            [[],[3, 75., -80.,-20]])
```



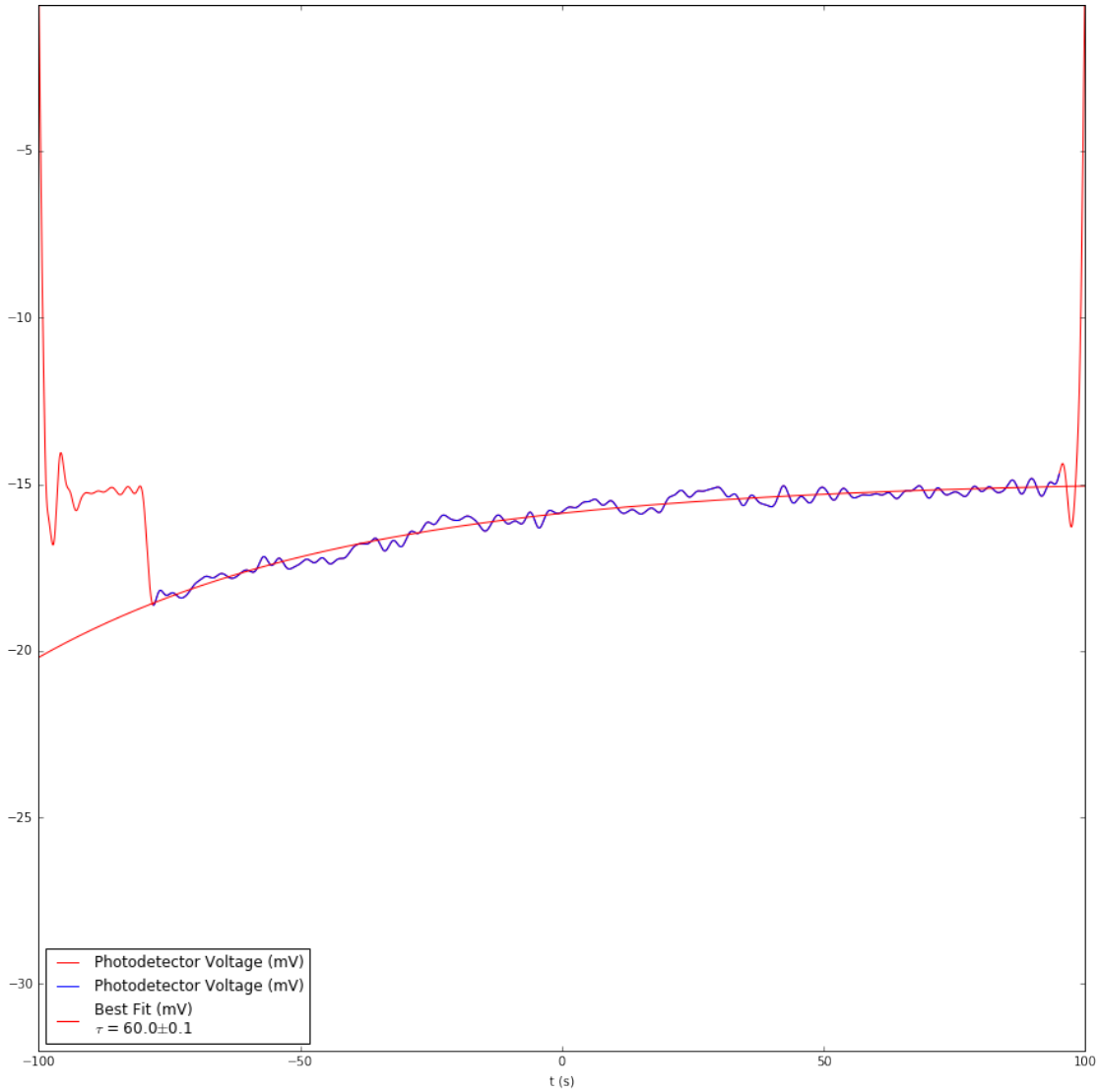

```
In [11]: test3 = MOT_data('./Data/22FEB2018/tek0003CH3.csv', 't (s)', 'Photodetector Voltage (mV
         # test3.plot_data([test3.data])
         test3.plot_data([test3.zoom(test0.data,-72,100)])
```



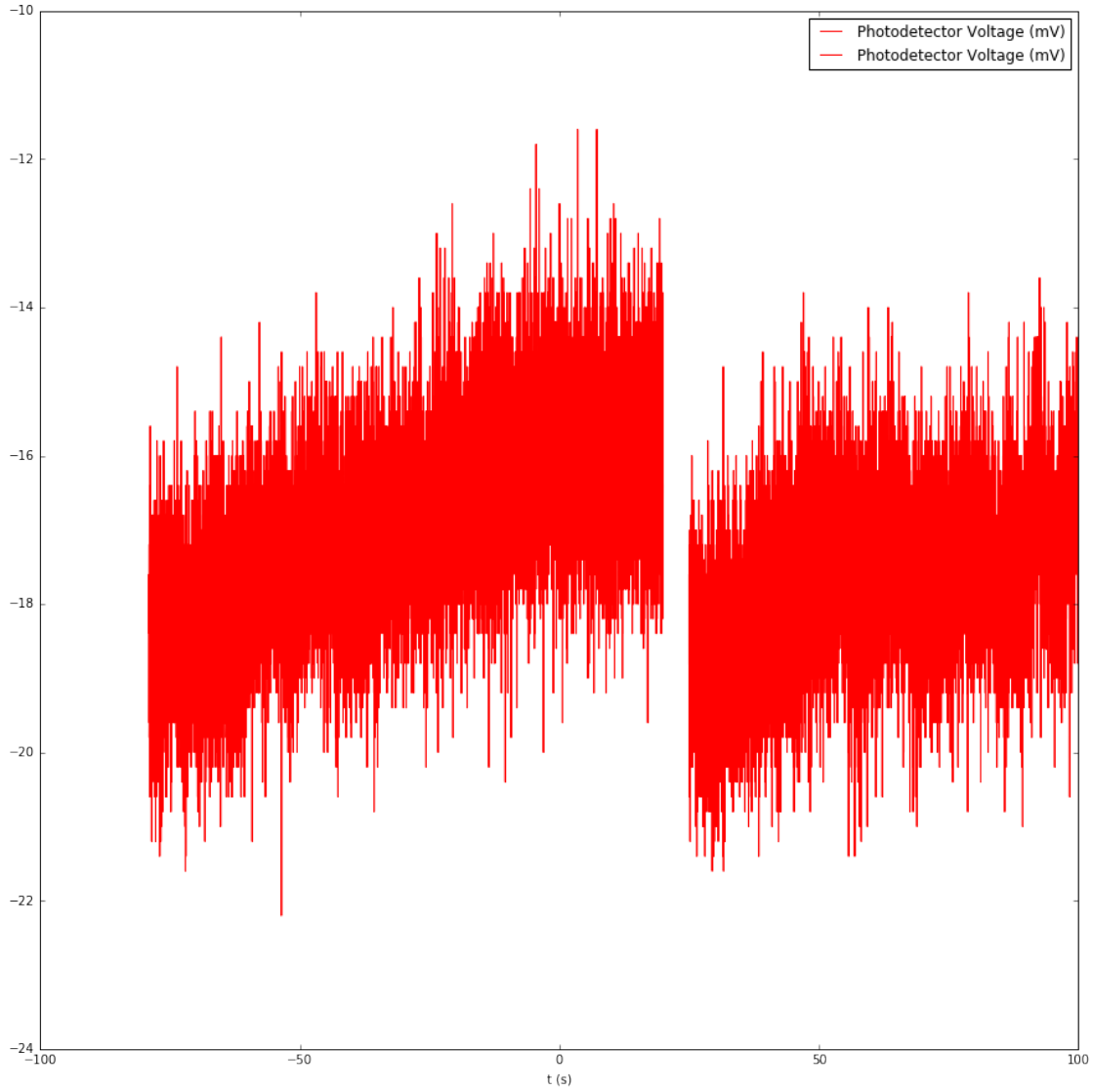

```
In [12]: test3.best_w_data([test3.data, test3.zoom(test3.data, -72,100)] ,\
                           [[],[3, 75., -70.,-19]])
```



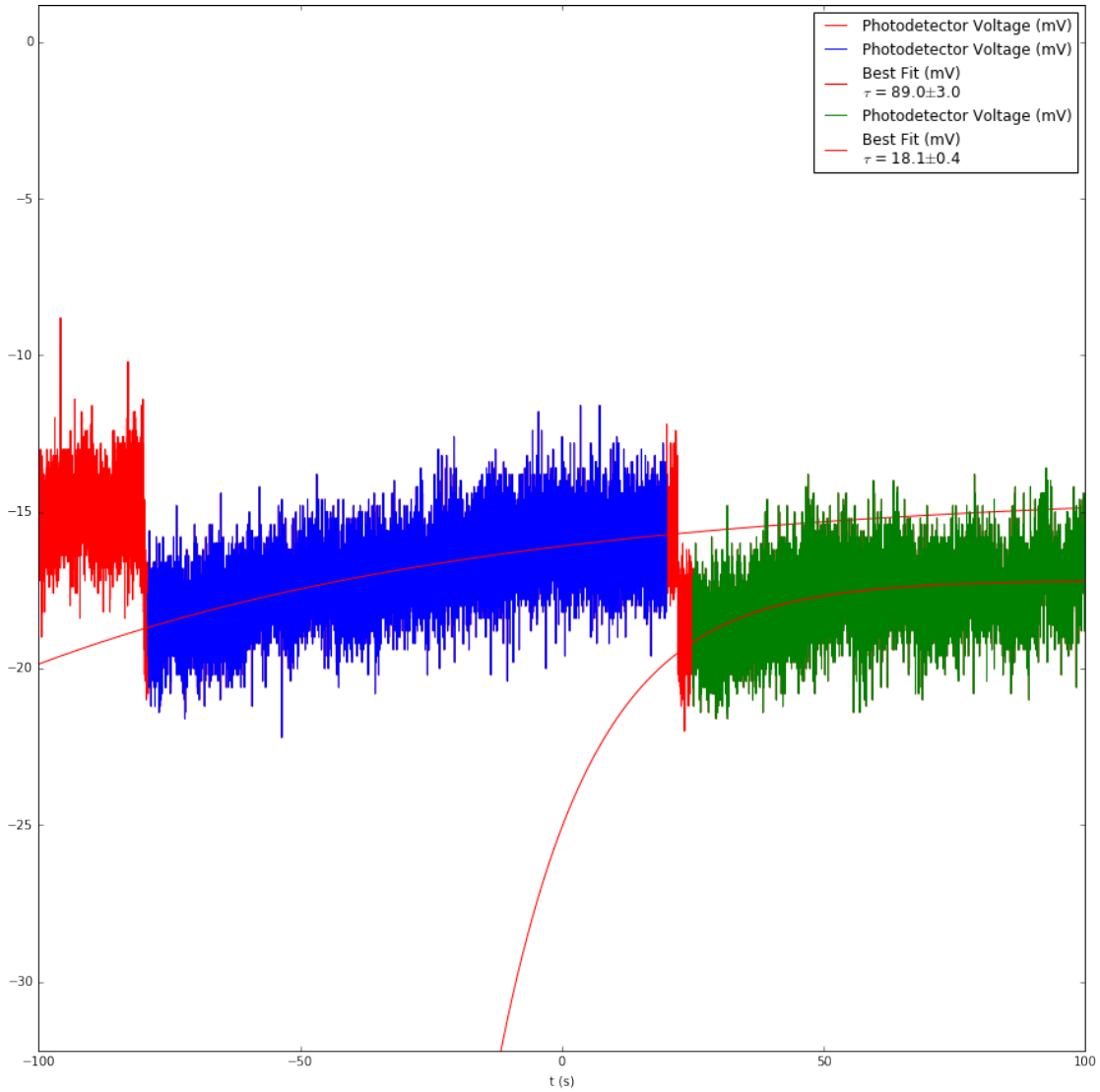

```
In [13]: test3.best_w_data([test3.filter_data_gust(5,0.001,120,.001),test1.zoom(test3.filter_dat
                            [[],[3, 75., -70.,-19]])
```



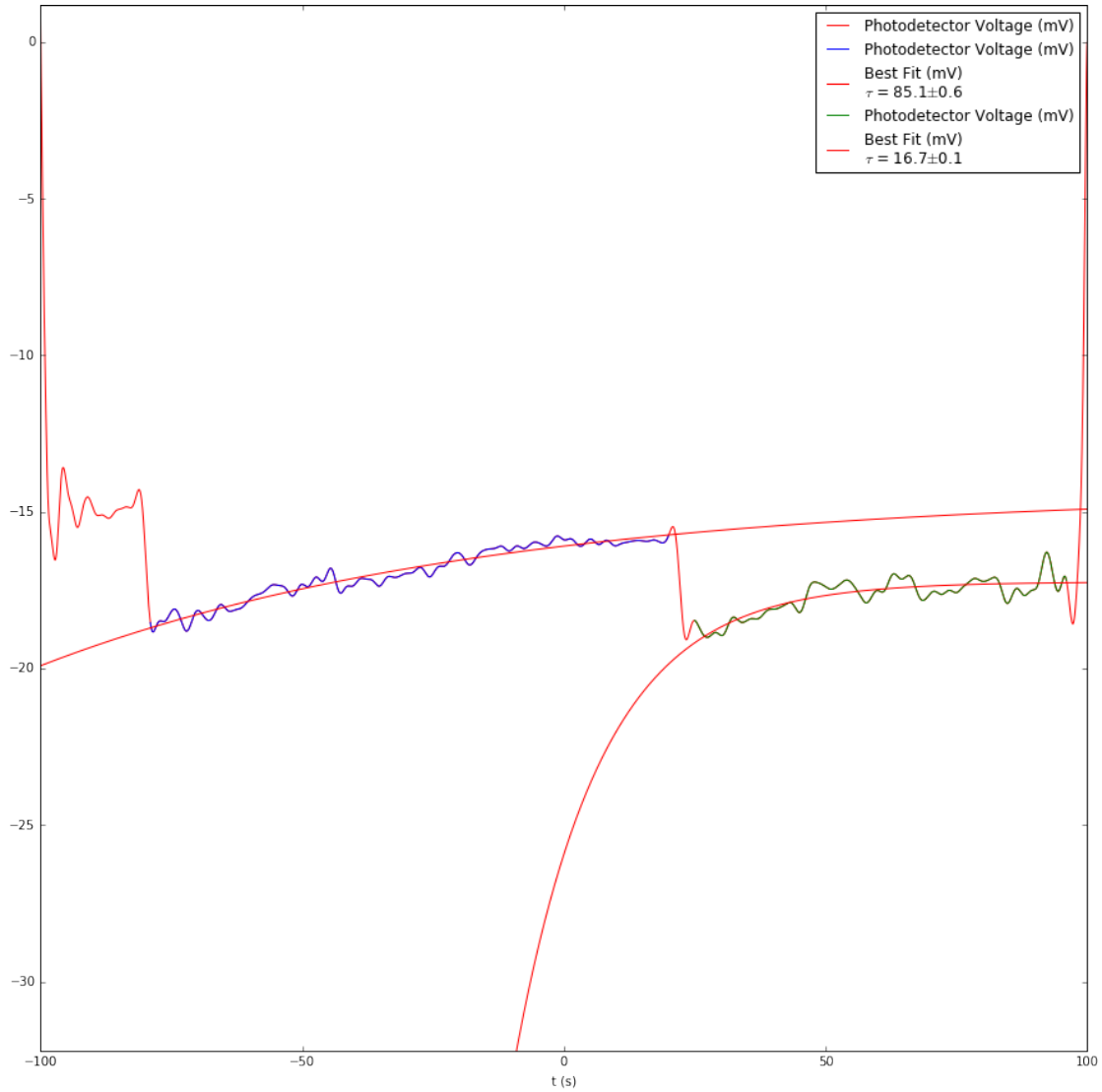

```
In [14]: test4 = MOT_data('./Data/22FEB2018/tek0004CH3.csv', 't (s)', 'Photodetector Voltage (mV
         # test4.plot_data([test4.data])
         test4.plot_data([test4.zoom(test4.data,-79,20),test4.zoom(test4.data,25,100)])
```



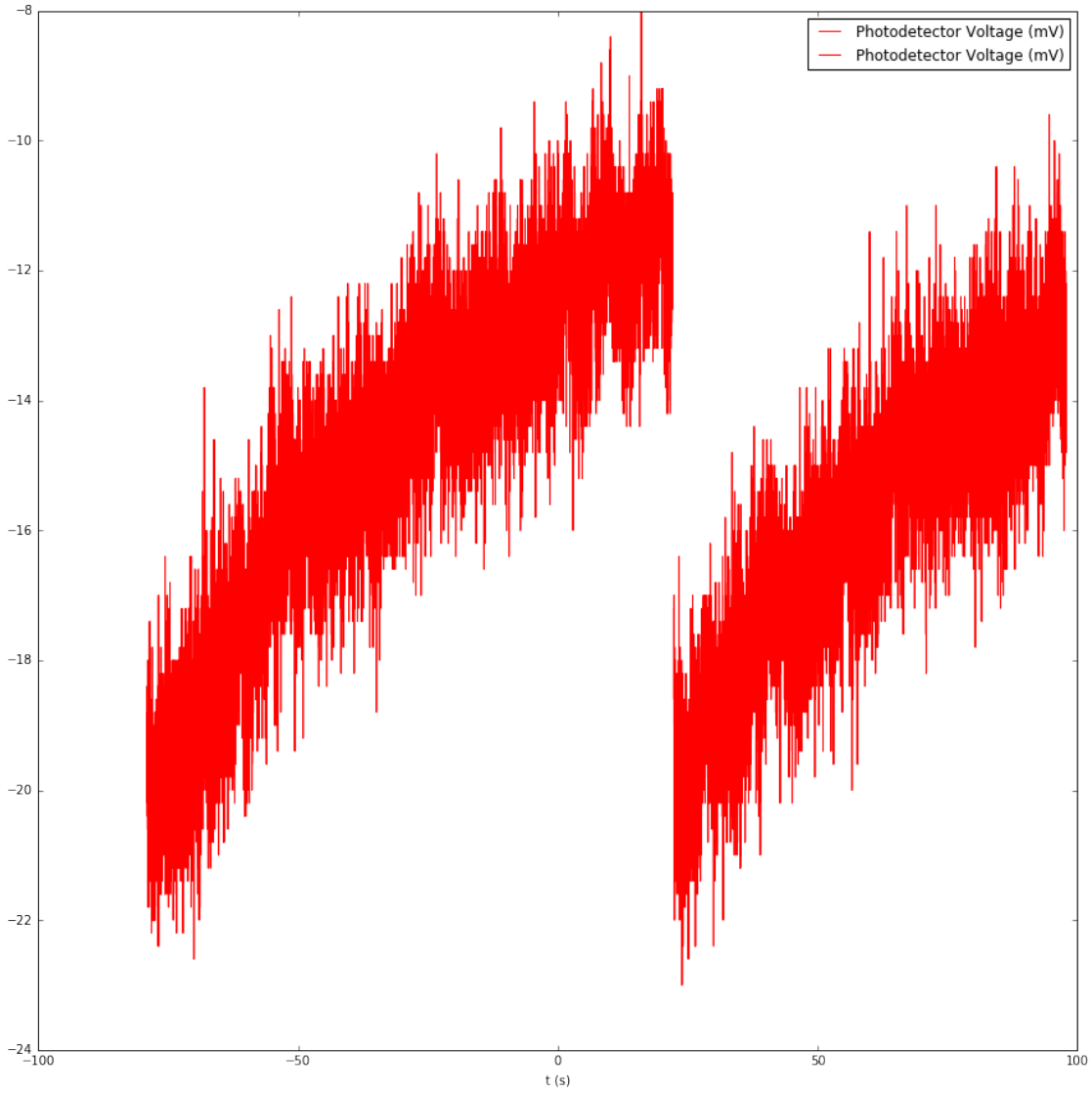

```
In [15]: test4.best_w_data([test4.data, test4.zoom(test4.data,-79,20), test4.zoom(test4.data,25,
                            [[],[3, 35., -80.,-19], [3, 35., 20.,-19]])
```



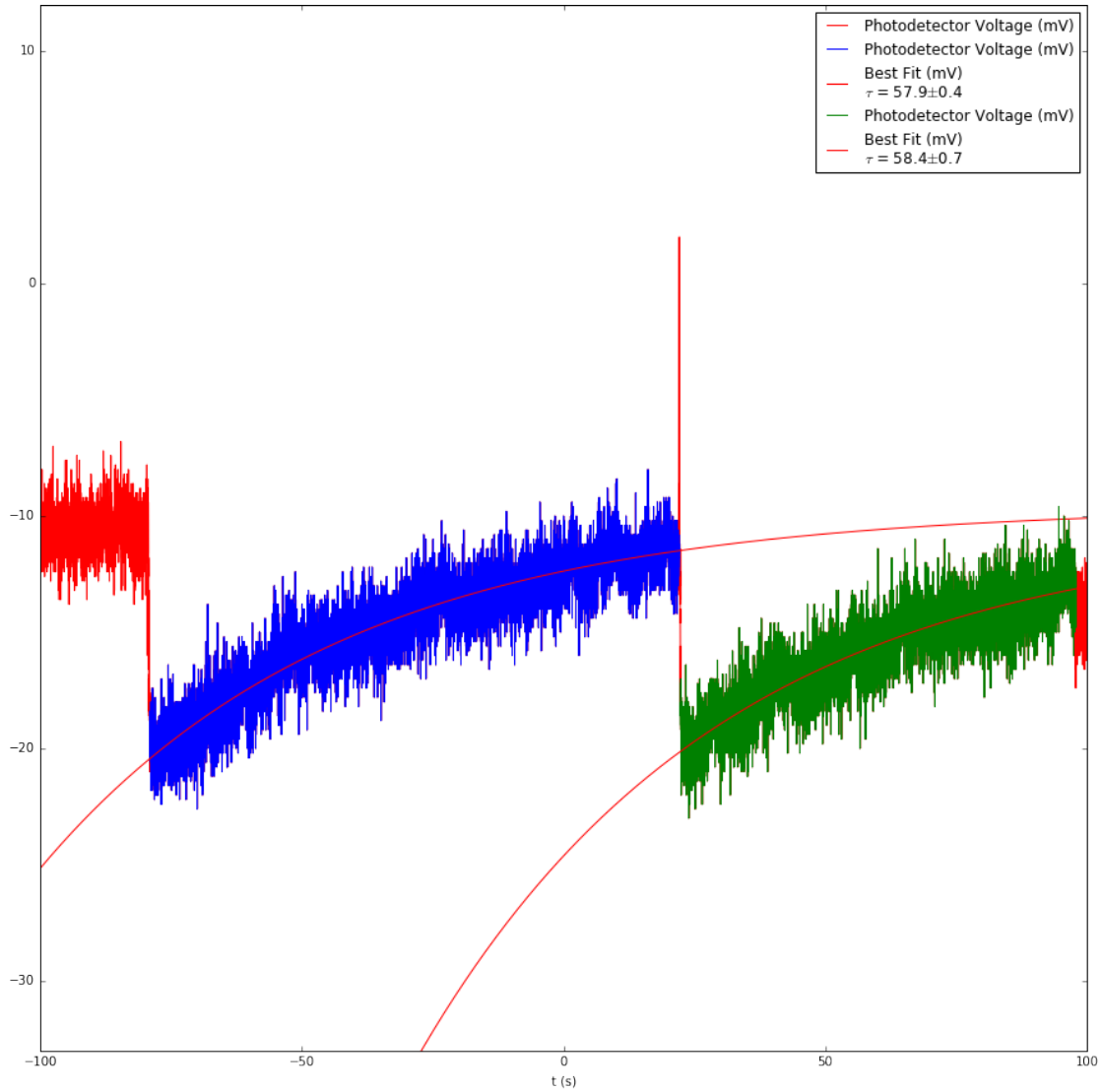

```
In [16]: test4.best_w_data([test4.filter_data_gust(5,0.001,120,.001),test4.zoom(test4.filter_dat
                            test4.zoom(test4.filter_data_gust(5,0.001,120,.001),25,96)] ,\
                           [[],[3, 35., -80.,-19], [3, 35., 20.,-19]])
```



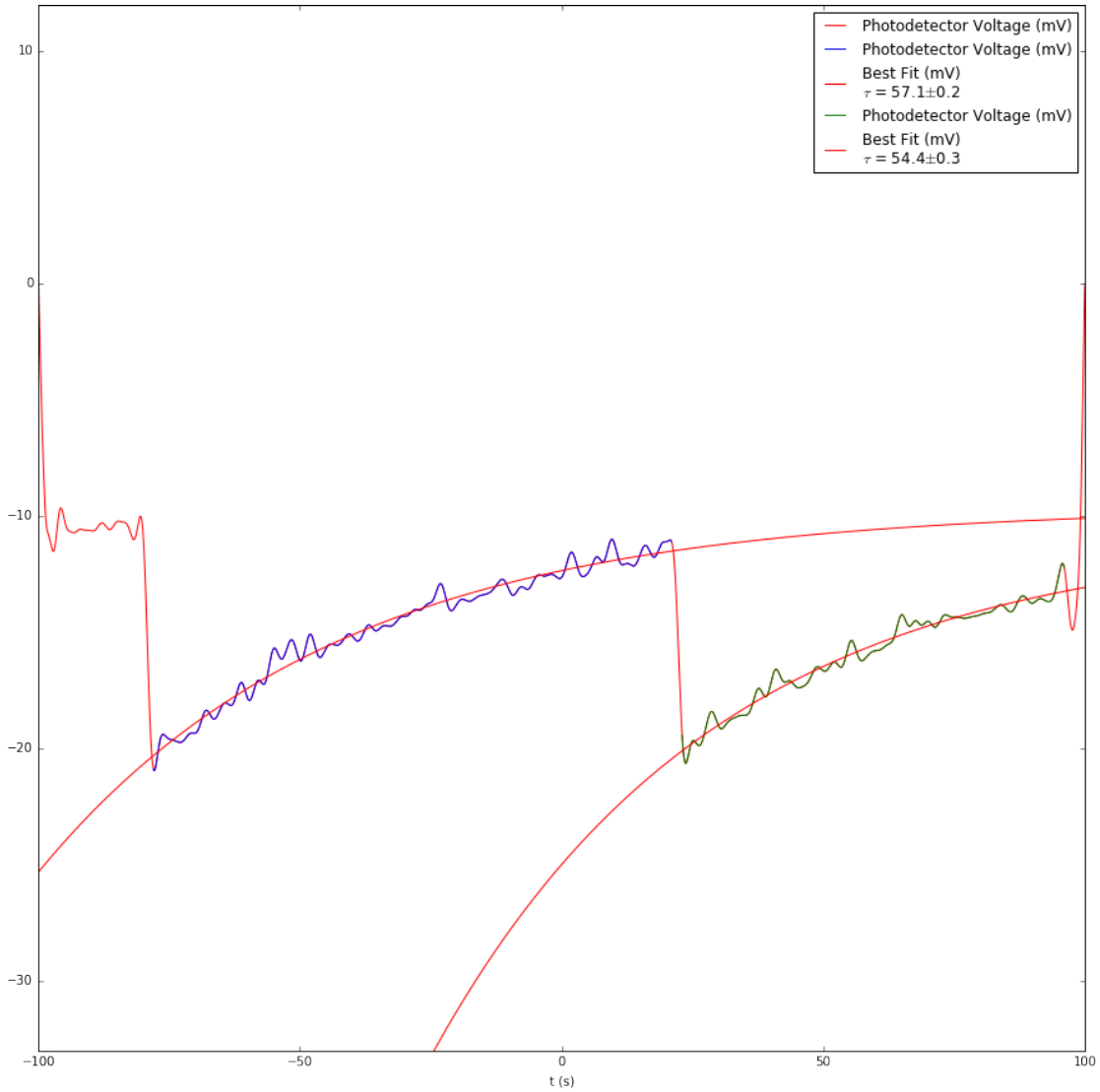

```
In [17]: test5 = MOT_data('./Data/22FEB2018/tek0005CH3.csv', 't (s)', 'Photodetector Voltage (mV
         test5.plot_data([test5.zoom(test5.data, -79,22), test5.zoom(test5.data,22.35,97.8)])
```



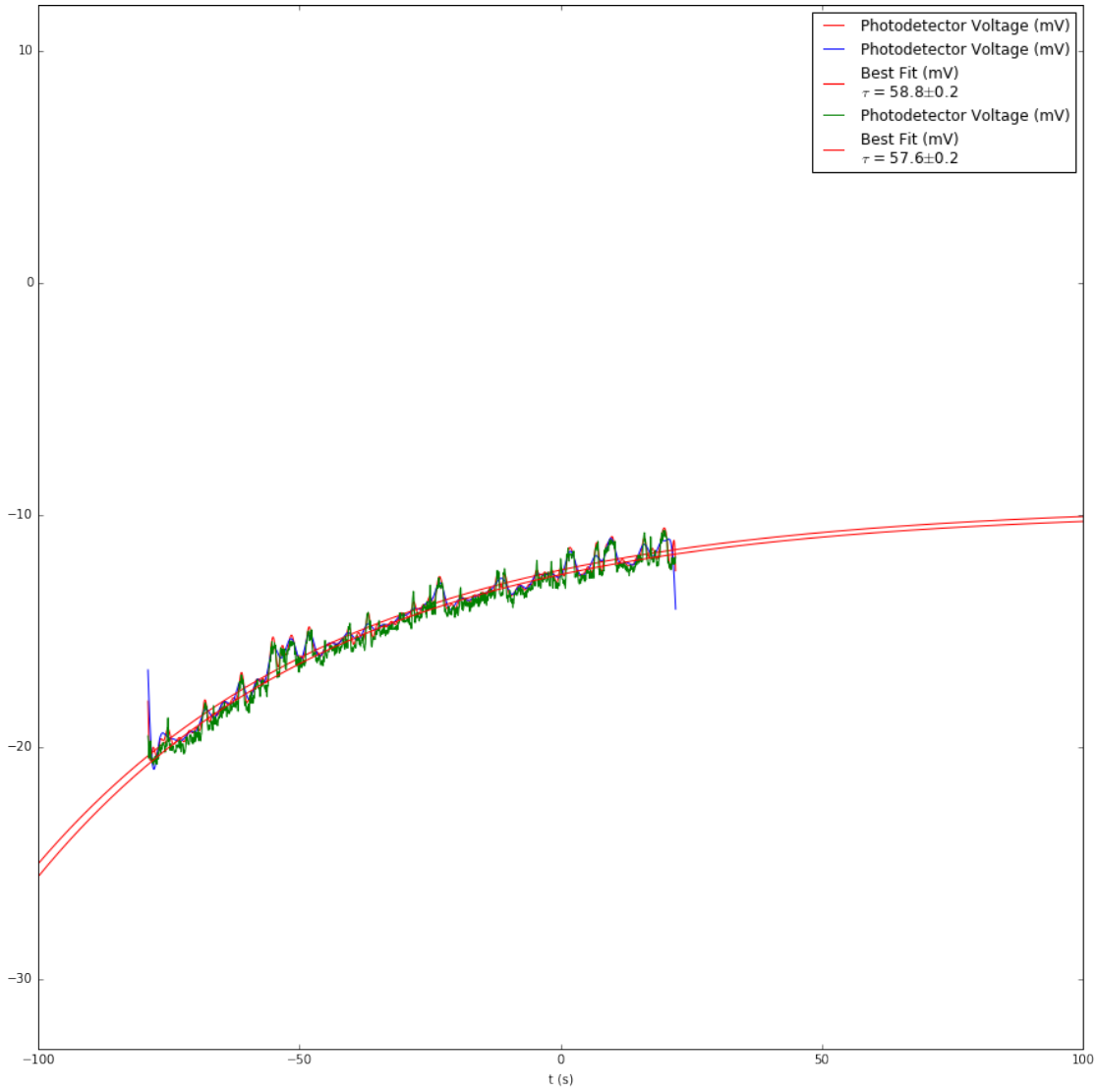

```
In [18]: test5.best_w_data([test5.data, test5.zoom(test5.data, -79,22), test5.zoom(test5.data,22
                            [[],[10, 50., -80.,-20], [10, 50., 20.,-20]])
```



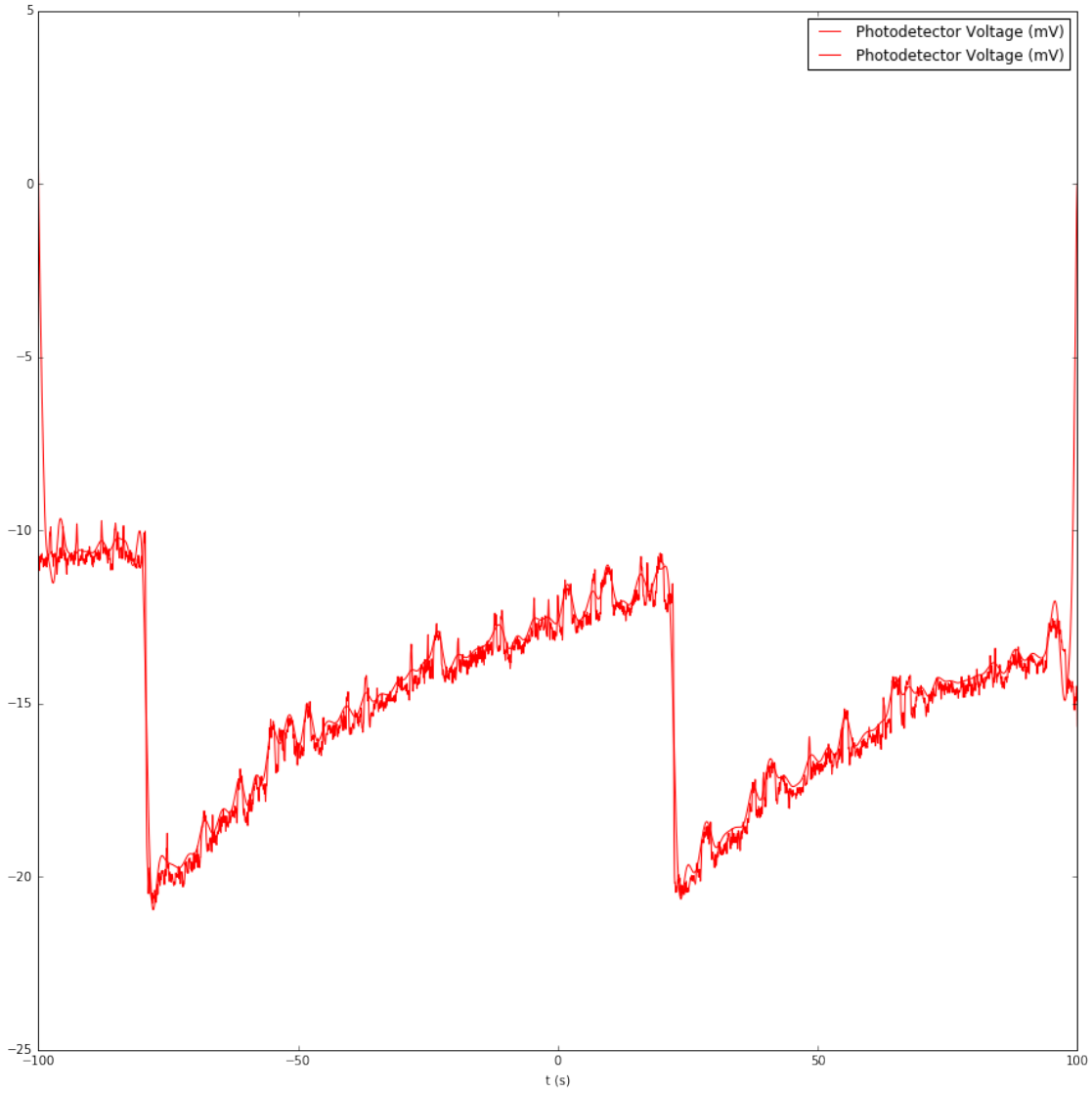

```
In [19]: test5.best_w_data([test5.filter_data_gust(5,0.001,120,.001),test5.zoom(test5.filter_dat
                            test5.zoom(test5.filter_data_gust(5,0.001,120,.001),23,96)] ,\
                           [[],[10, 50., -80.,-20], [10, 50., 20.,-20]])
```



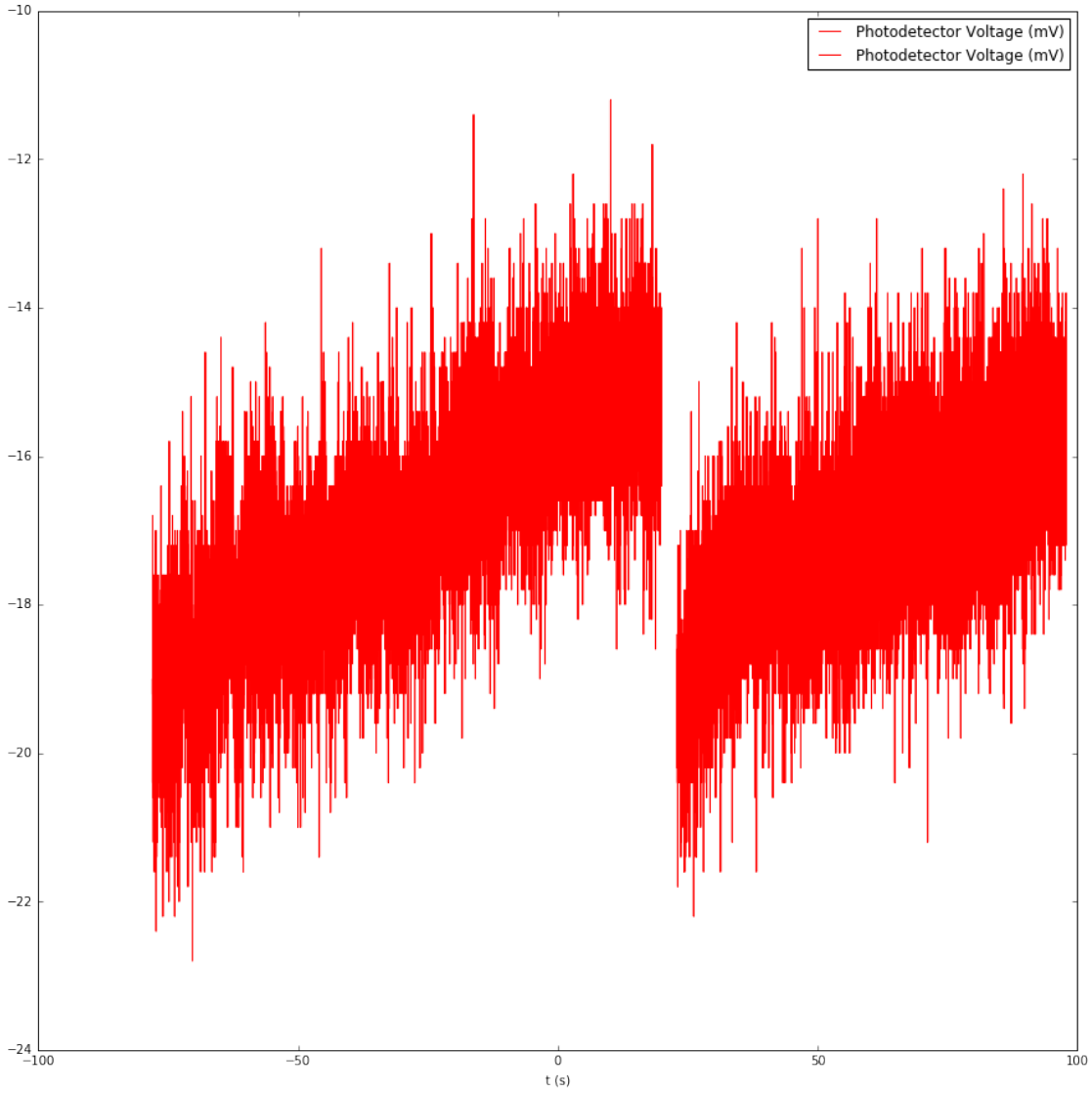
In [20]:
```
# Compare fits of the different filters on each peak
test5.best_w_data([test5.zoom(test5.filter_data_gust(5,0.001,120,.002), -79,22),\
                   test5.zoom(test5.filter_data_gust(5,0.001,120,.001), -79,22),\
                   test5.zoom(test5.filter_data_butter(10,.025, 150), -79,22)],\
                  [[10, 50., -80.,-20], [10, 50., -80.,-20],[10, 50., -80.,-20]])
```



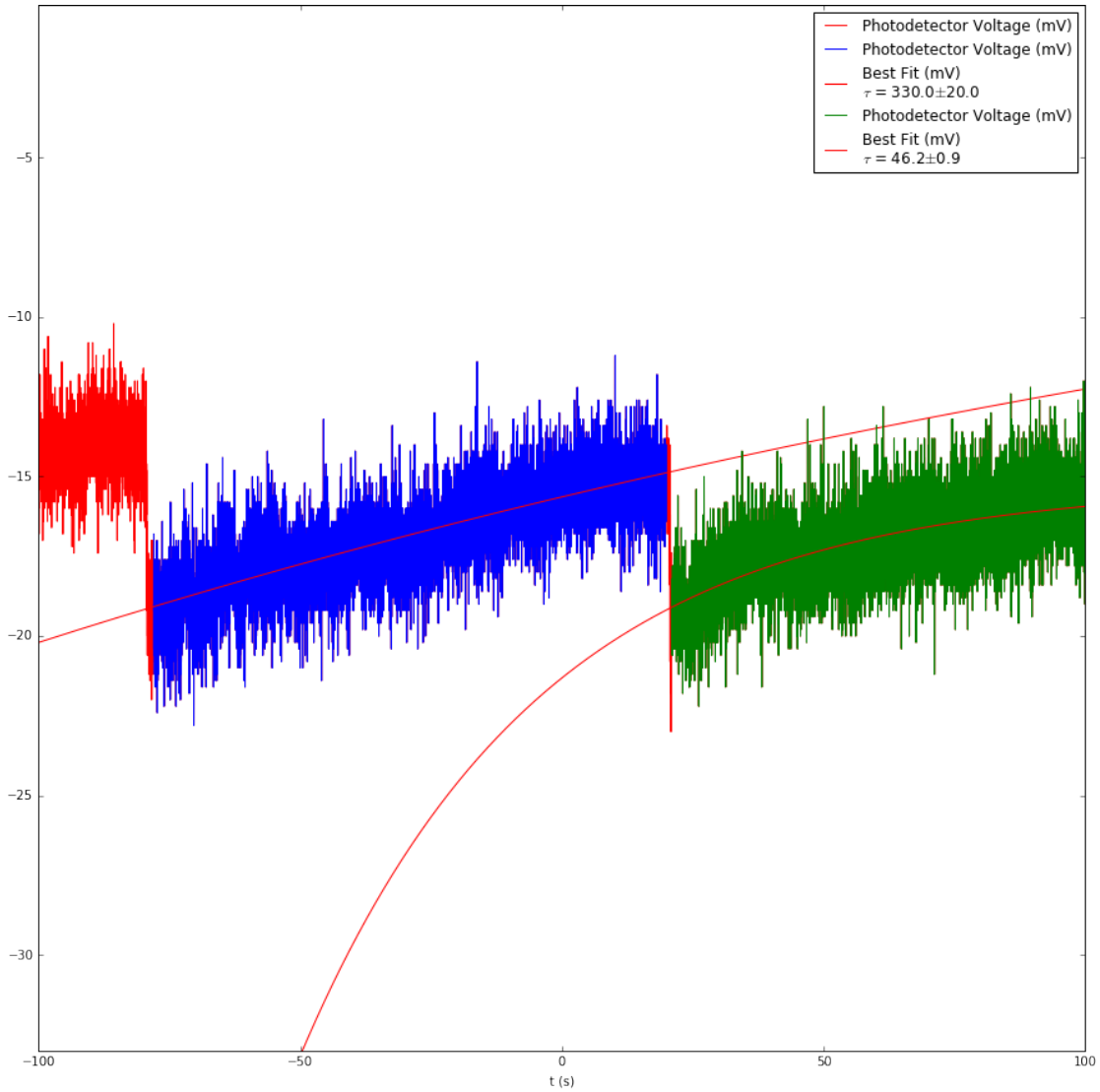
In [21]: test5.plot_data([test5.filter_data_butter(10,.025, 150), test5.filter_data_gust(5,0.001



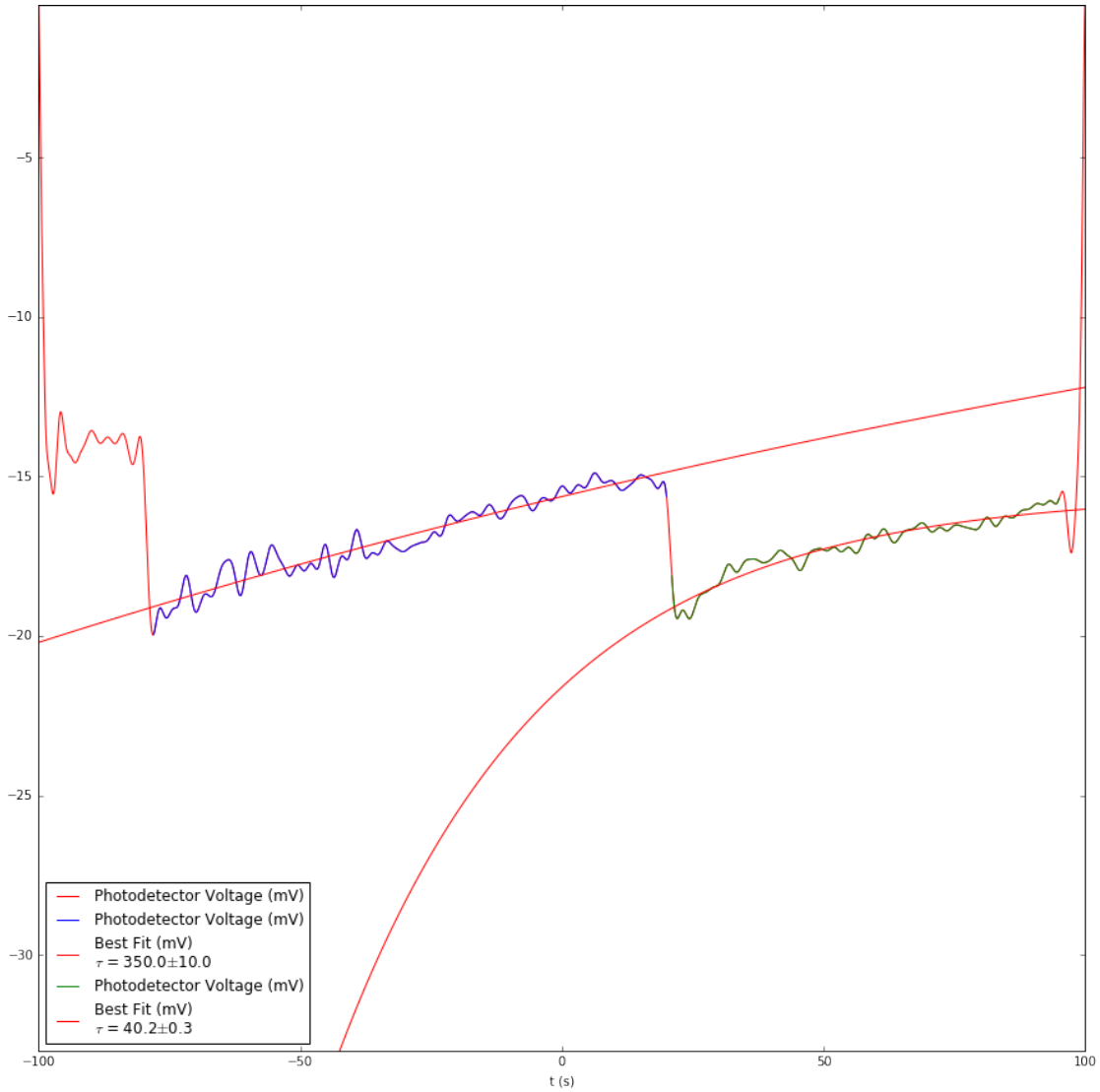

```
In [22]: test6 = MOT_data('./Data/22FEB2018/tek0006CH3.csv', 't (s)', 'Photodetector Voltage (mV
         # test6.plot_data([test6.data])#test6.zoom(test6.data,22.35,97.8)
         test6.plot_data([test6.zoom(test6.data, -78,20), test6.zoom(test6.data,23,97.8)])
```



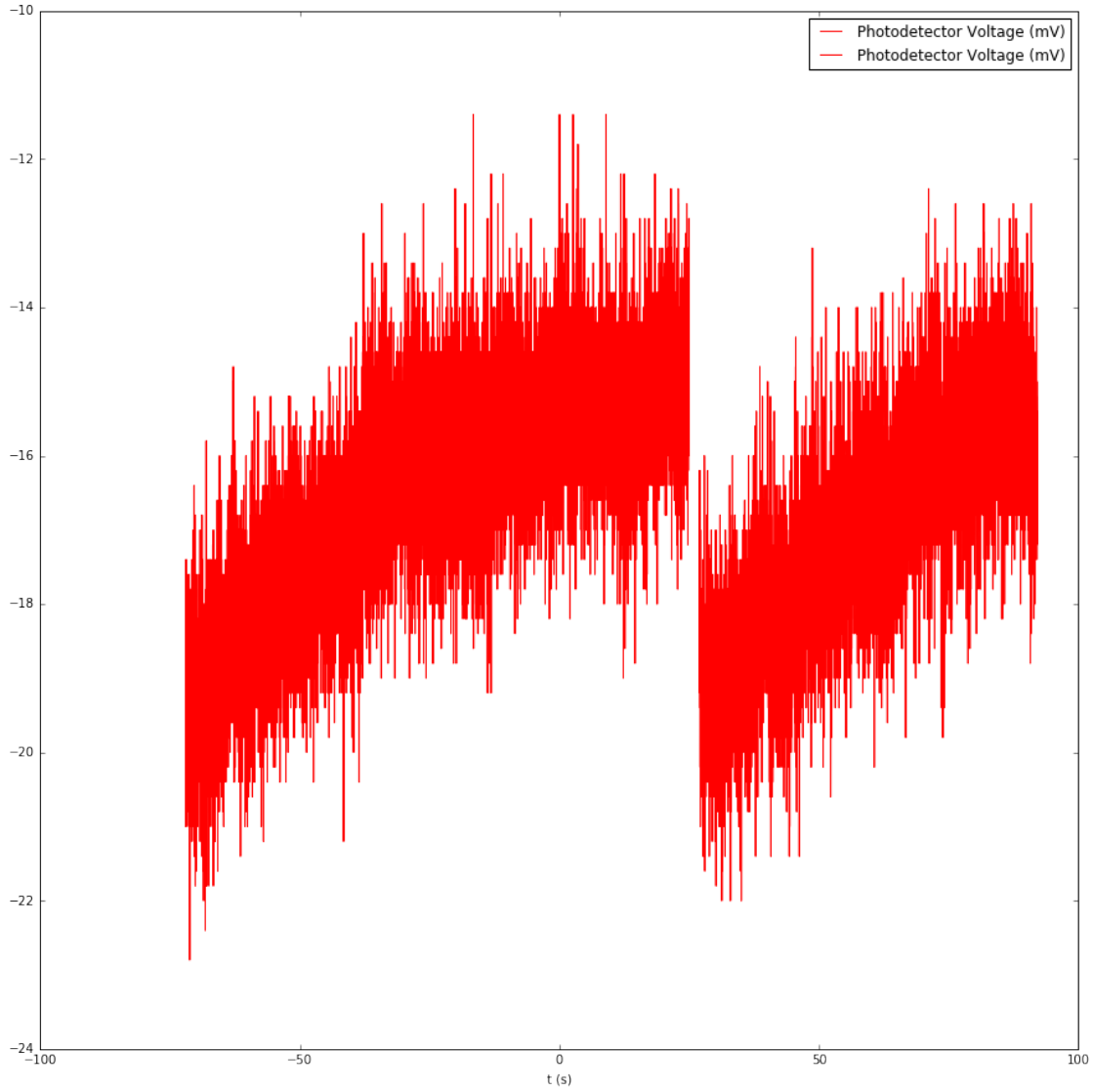

```
In [23]: test6.best_w_data([test6.data, test6.zoom(test6.data,-78,20), test6.zoom(test6.data,21,
                            [[],[5, 50., -78.,-20], [5, 50., 21.,-20]])
```



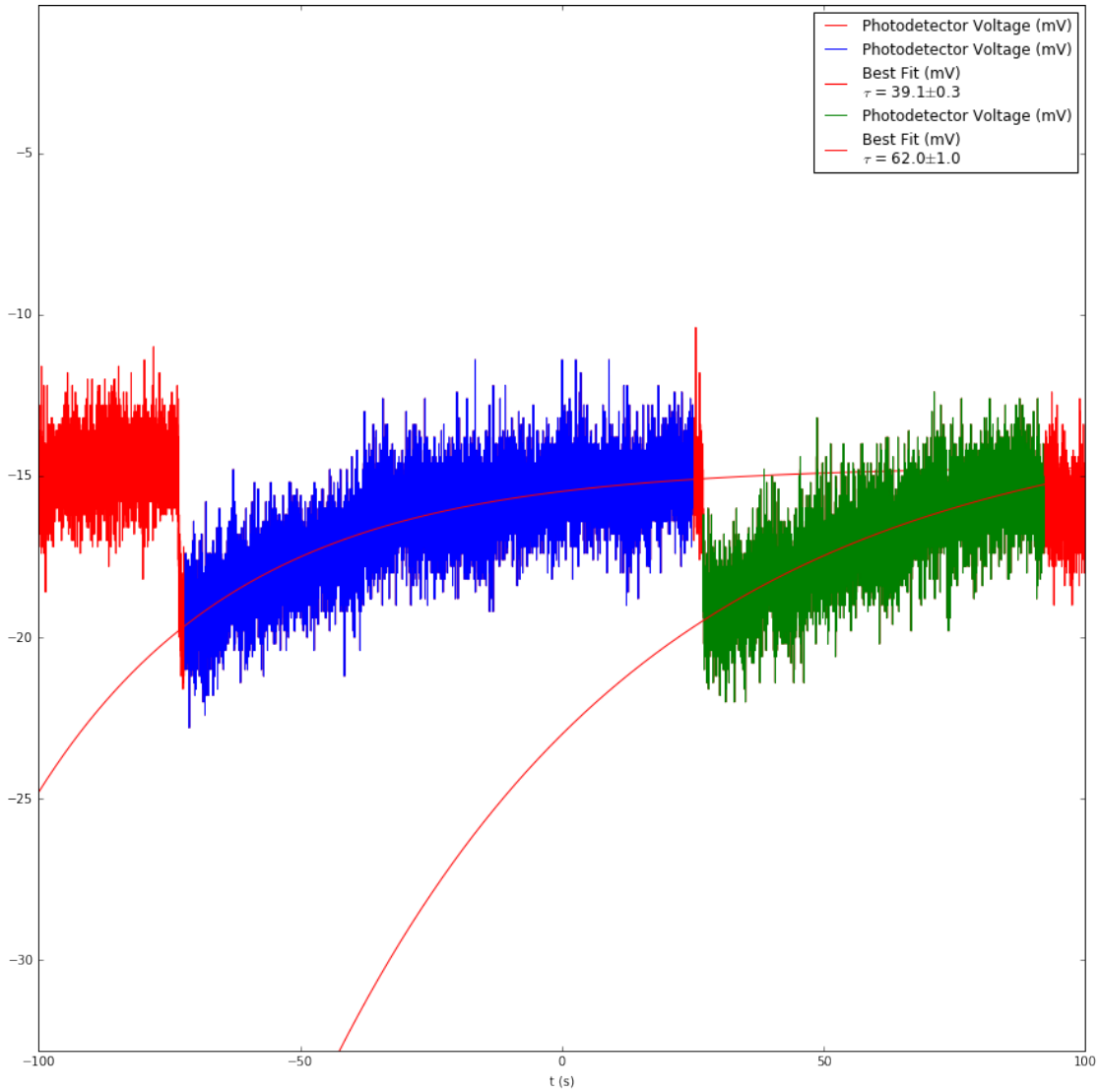

```
In [24]: test6.best_w_data([test6.filter_data_gust(5,0.001,120,.001),test6.zoom(test6.filter_dat
                            test6.zoom(test6.filter_data_gust(5,0.001,120,.001),21,95)] ,\
                           [[],[5, 50., -78.,-20], [5, 50., 21.,-20]])
```



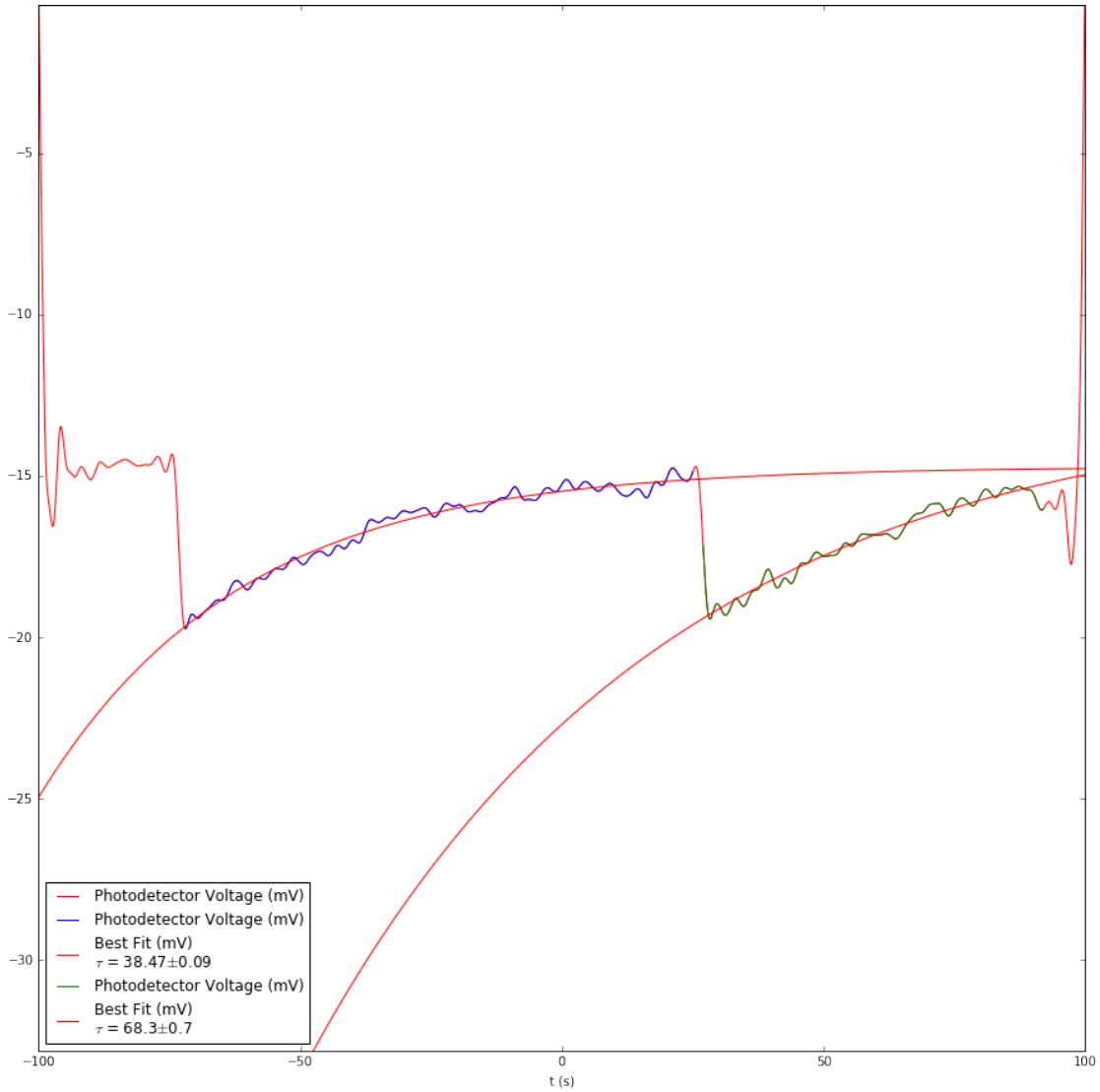
In [25]:
```
test7 = MOT_data('./Data/22FEB2018/tek0007CH3.csv', 't (s)', 'Photodetector Voltage (mV
# test7.plot_data([test7.data])#test6.zoom(test6.data,22.35,97.8)
test7.plot_data([test7.zoom(test7.data, -72,25), test7.zoom(test7.data,27,92)])
```



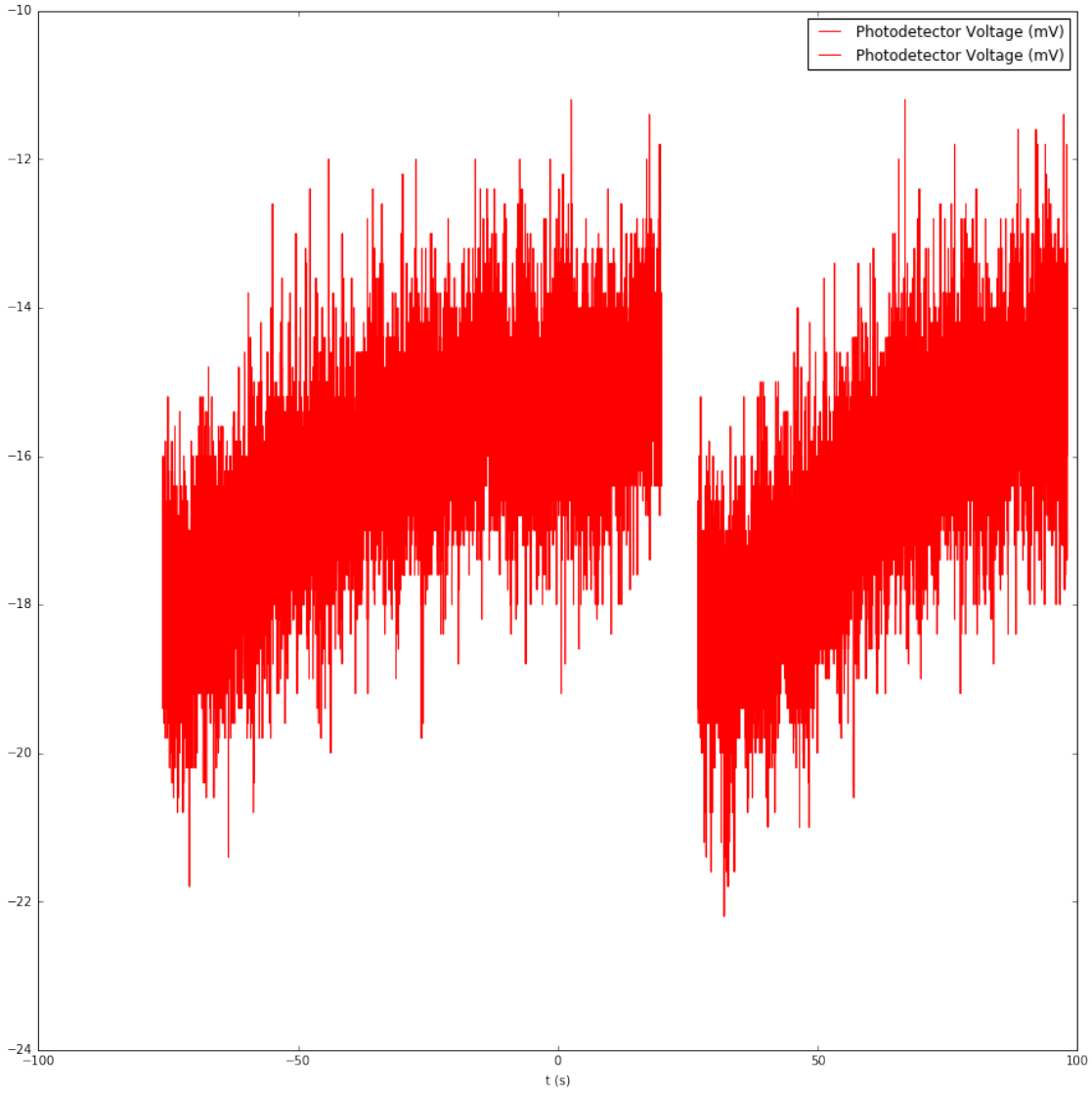

```
In [26]: test7.best_w_data([test7.data, test7.zoom(test7.data,-72,25), test7.zoom(test7.data,27,
                            [[],[5, 50., -78.,-20], [5, 50., 21.,-20]])
```



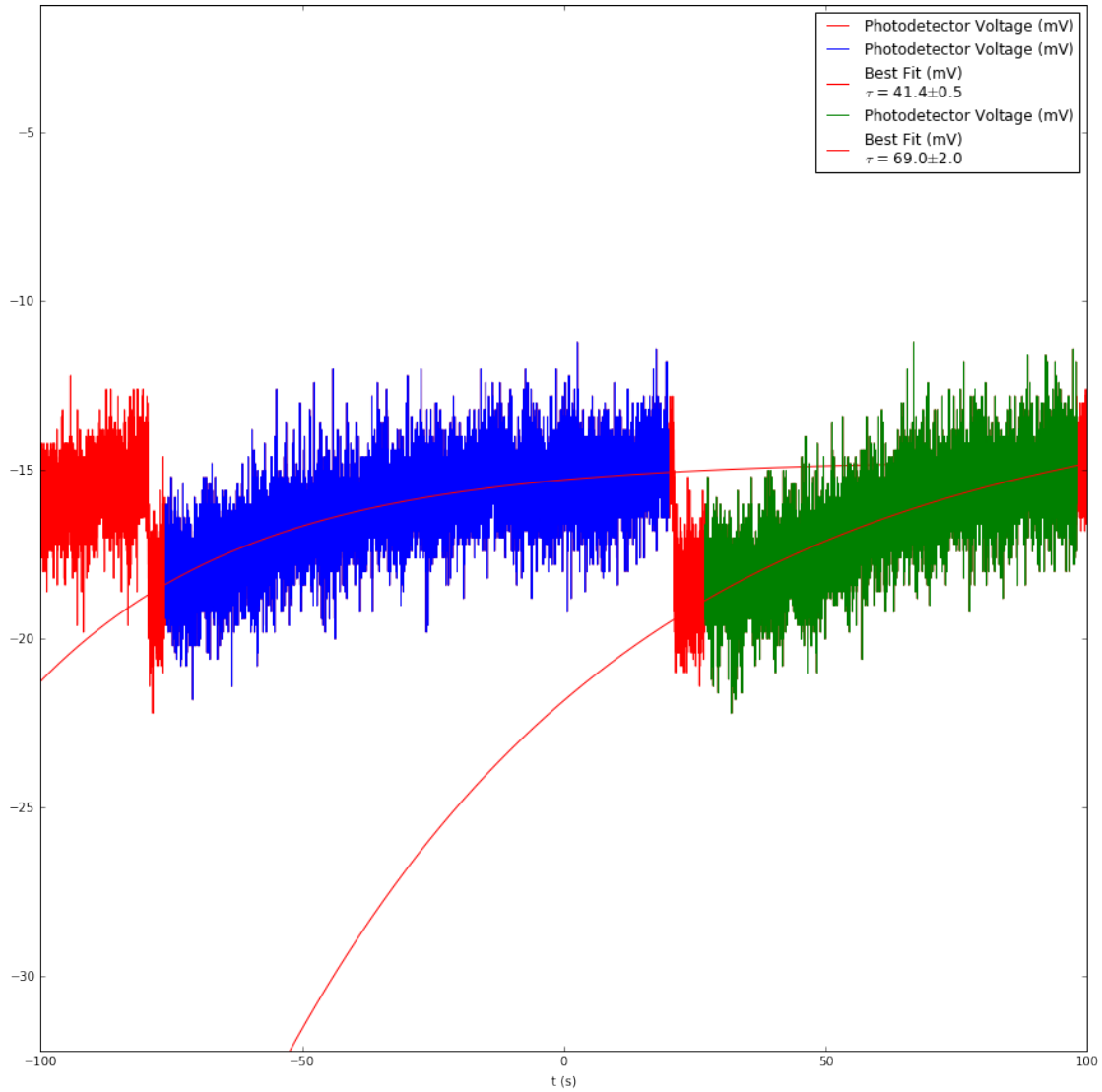

```
In [27]: test7.best_w_data([test7.filter_data_gust(5,0.001,120,.001),test7.zoom(test7.filter_dat
                            test7.zoom(test7.filter_data_gust(5,0.001,120,.001),27,92)] ,\
                           [[],[5, 50., -78.,-20], [5, 50., 21.,-20]])
```



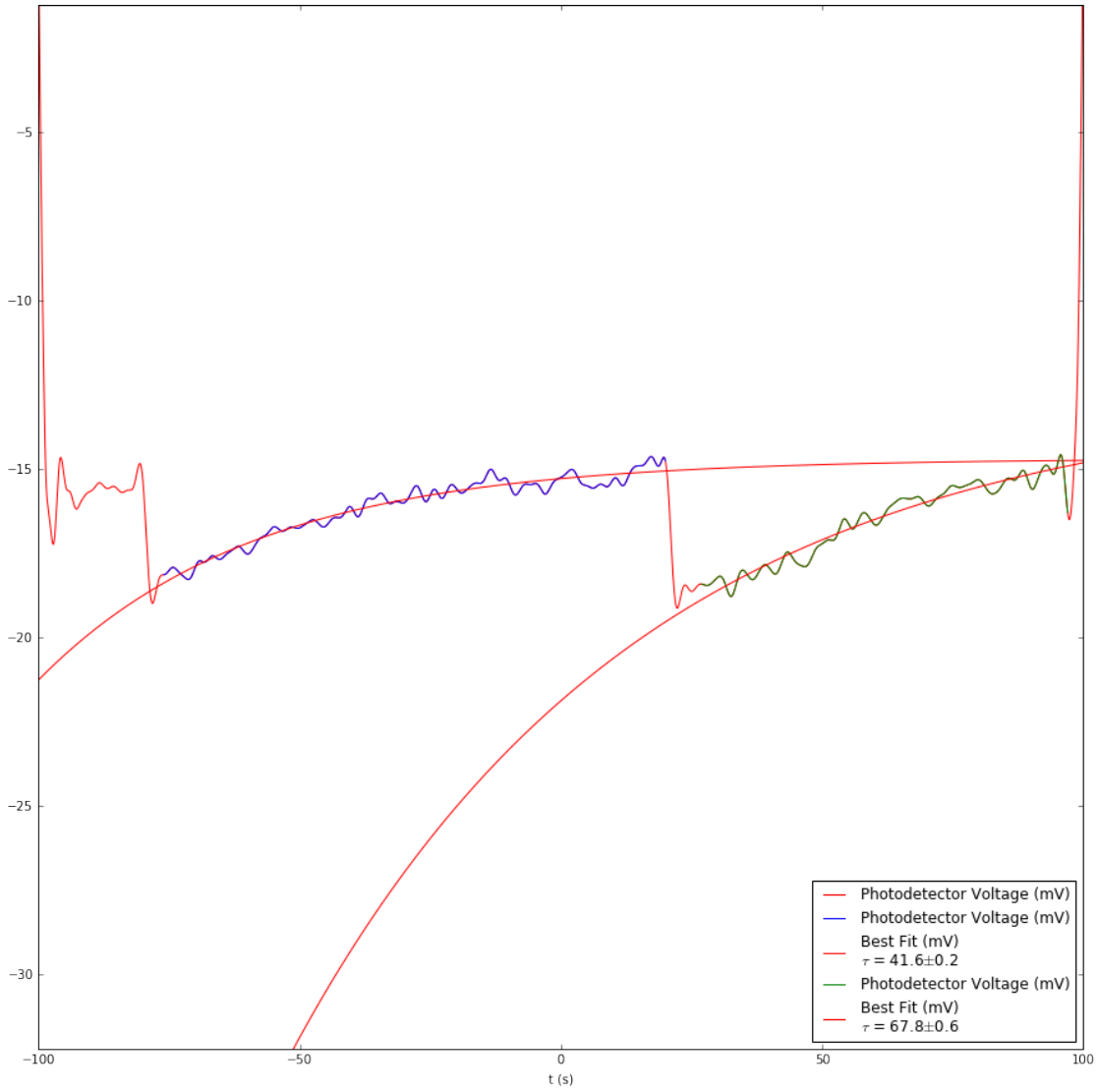

```
In [28]: test8 = MOT_data('./Data/22FEB2018/tek0008CH3.csv', 't (s)', 'Photodetector Voltage (mV
         # test8.plot_data([test8.data])
         test8.plot_data([test8.zoom(test8.data, -76,20), test8.zoom(test8.data,27,98)])
```




```
In [29]: test8.best_w_data([test8.data, test8.zoom(test8.data, -76,20), test8.zoom(test8.data,27
                            [[],[5, 50., -78.,-20], [5, 50., 21.,-20]])
```




```
In [30]: test8.best_w_data([test8.filter_data_gust(5,0.001,120,.001),test8.zoom(test8.filter_dat
                            test8.zoom(test8.filter_data_gust(5,0.001,120,.001),27,97)] ,\
                           [[],[5, 50., -78.,-20], [5, 50., 21.,-20]])
```


−5
−10
−15
−20
−25
−30
−100
−50
0
50
100
t (s)
Photodetector Voltage (mV)
Photodetector Voltage (mV)
Best Fit (mV)
$\tau = 41.6 \pm 0.2$
Photodetector Voltage (mV)
Best Fit (mV)
$\tau = 67.8 \pm 0.6$