Biophotonics Code

Jack Symonds

Measuring the Q value:

```
import numpy as np
import matplotlib.pyplot as plt

c = 3*10**8
ref_fluid = 1.332
visc_water = 0.001

def power(current):
    power = 0.8154*current - 41.954
    return 0.5*power

class VTrap:
    def __init__(self, filename, radius):
        self.radius = radius
        with open(filename) as meas:
            lines = meas.readlines()
            currents = []
            vels = []
            devs = []
            for line in lines[1:]:
                line = line.split()
                for i in range(len(line)): line[i] = float(line[i])
                currents.append(line[0])
                vals = np.array(line[1:])
                avg = np.mean(vals)
                dev = np.sqrt((1/(vals.size-1))*np.sum((vals-avg)**2))/np.sqrt(vals.size)
                vels.append(avg)
                devs.append(dev)
            self.currents = np.array(currents)
            self.vels = np.array(vels)
            self.power = power(self.currents)
            self.devs = np.array(devs)

    def plot(self, ax, color):
        ax.set_xlabel('Power Output of Laser (mW)')
        ax.xaxis.tick_top()
        ax.xaxis.set_label_position('top')
        ax.set_ylabel(r'Average Velocity of Trapped Particle ($\mu$m/s)')
        (m, b), res, _,_,_ = np.polyfit(self.power, self.vels, 1, full=True)
        print(m, b, res)
        x = np.linspace(np.min(self.power), np.max(self.power), 4)
        ax.plot(x, m*x+b, linestyle='--', color=color, linewidth=0.8)

        Q = 6*np.pi*visc_water*self.radius/(ref_fluid*m)
        radius = r'$r = %s$ $\mu$m'%self.radius
        trend = r'$P_L = %.3f\cdot v_c %+.3f$'%(m,b)
        q_value = r'$Q = %.3f$'%Q
        cellText.append([radius, trend, q_value])
        ax.scatter(self.power, self.vels, label=radius, color=color)
        ax.errorbar(self.power, self.vels, yerr=self.devs, fmt='none', color=color)

fig, ax = plt.subplots()
ax.grid(True)
cellText=[]

size1 = VTrap('1mm.tsv', 1)
size3 = VTrap('3mm.tsv', 3)

size1.plot(ax, 'grey')
size3.plot(ax, 'black')
```

```
ax.legend(loc='upper left')
ax.table(cellText=cellText, cellLoc='left')
#fig.savefig('../images/qval.png', quality=100)
#plt.show()
```

Varying Fluorescein concentration:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

molar_conc = 75.23 #micromolar/m^3

class CvES:
    def __init__(self, filename):
        df = pd.read_csv(filename, sep='\t')
        self.conc = df['conc']*molar_conc/(df['div']+df['conc'])
        self.em = df['em']
        self.strength = df['strength']/(0.001*df['itime'])

    def plot(self, ax):
        ax.set_yscale("log")
        ax.set_xscale("log")
        ax.scatter(self.conc, self.strength)


fig, ax = plt.subplots()
fig.subplots_adjust(left=0.15, bottom=0.15)
ax.grid(True)
ax.set_title('Fluorescein Concentration Signal Strength')
ax.set_xlabel(r'Molar Concentration of Sample $\left( \frac{\mu m}{m^3} \right)$')
ax.set_ylabel('Peak Strength (Counts per Second)')

conc_data = CvES('conc.tsv')

conc_data.plot(ax)

fig.savefig('../images/conc.png')
plt.show()
```

Varying concentration on wavelength:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

molar_conc = 75.23 #micromolar/m^3

class CvES:
    def __init__(self, filename):
        df = pd.read_csv(filename, sep='\t')
        self.conc = df['conc']*molar_conc/(df['div']+df['conc'])
        self.em = df['em']
        self.strength = df['strength']/(0.001*df['itime'])

    def plot(self, ax):
        ax.set_yscale("linear")
        ax.set_xscale("log")
        ax.scatter(self.conc, self.em)


fig, ax = plt.subplots()
fig.subplots_adjust(left=0.15, bottom=0.15)
ax.grid(True)
ax.set_title('Fluorescein Concentration Peak Wavelength')
ax.set_xlabel(r'Molar Concentration of Sample $\left( \frac{\mu m}{m^3} \right)$')
ax.set_ylabel('Peak Fluorescence Wavelength (nm)')

conc_data = CvES('conc.tsv')

conc_data.plot(ax)

fig.savefig('../images/conc_wav.png')
plt.show()
```

Stain traces

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

class Spectrum:
    def __init__(self, filename):
        with open(filename) as text:
            lines = text.readlines()
            int_time = float(lines[6].split()[-1])
            wv = []
            strength = []
            for line in lines[14:]:
                line = line.split()
                wv.append(float(line[0]))
                strength.append(float(line[1]))

        self.wavelengths = np.array(wv)
        self.strength = np.array(strength)/int_time

    def plot(self, ax, color, label):
        peaks_ind = sig.find_peaks(self.strength, height=400, distance=70, width=1)[0][:4]
        for i in peaks_ind: ticks.append(self.wavelengths[i])
        ax.set_xticks(ticks)
        ax.plot(self.wavelengths, self.strength, linewidth=0.5, color=color,label=label)




fig, ax = plt.subplots()
ax.axhline(0,linewidth=0.5,linestyle='--', color='black')
ax.set_xlim(340,620)
ax.set_xlabel('Wavelength (nm)')
ax.set_ylabel('Strength (counts/second)')
ax.set_title('Traces of LEDs on x1 SYBR1')
ticks=[]
ax.grid(True)

blue = Spectrum('rstain.txt')
uv = Spectrum('rstainuv.txt')

blue.plot(ax, 'blue', 'Blue LED')
uv.plot(ax, 'black', 'UV LED')

ax.legend(loc='upper right')
fig.savefig('../images/stain.png', quality=100)
plt.show()
```

Spectra for dsDNA concentrations

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

class Spectrum:
    def __init__(self, filename):
        with open(filename) as text:
            lines = text.readlines()
            int_time = float(lines[6].split()[-1])
            wv = []
            strength = []
            for line in lines[14:]:
                line = line.split()
                wv.append(float(line[0]))
                strength.append(float(line[1]))

        self.wavelengths = np.array(wv)
        self.strength = np.array(strength)/int_time

    def plot(self, ax, color, label):
        ax.plot(self.wavelengths, self.strength, linewidth=0.5, color=color,label=label)




fig, ax = plt.subplots()
ax.axhline(0,linewidth=0.5,linestyle='--', color='black')
ax.set_xlim(400,670)
ax.set_xlabel('Wavelength (nm)')
ax.set_ylabel('Strength (counts/second)')
ax.grid(True)

full_conc = Spectrum('full_conc.txt')
conc_80_1 = Spectrum('80_1.txt')
conc_80_2 = Spectrum('80_2.txt')
conc_50_1 = Spectrum('50_1.txt')
conc_50_2 = Spectrum('50_2.txt')

full_conc.plot(ax,'black', '1')
conc_80_1.plot(ax,'black','1')
conc_80_2.plot(ax,'black','1')
conc_50_1.plot(ax,'black','1')
conc_50_2.plot(ax,'black','1')

fig.savefig('../images/conc2.png', quality=100)
plt.show()
```

Peak strength for dsDNA concentrations

```
import numpy as np
import matplotlib.pyplot as plt

def get_peak(filename):
    with open(filename) as text:
        lines = text.readlines()
        int_time = float(lines[6].split()[-1])
        wv = []
        strength = []
        for line in lines[14:]:
            line = line.split()
            wv.append(float(line[0]))
            strength.append(float(line[1]))

    strength = np.array(strength)/int_time
    peak = np.max(strength)
    strengths.append(peak)
    peak_wav.append(wv[np.argmax(strength)])

strengths = []
peak_wav = []

get_peak('full_conc.txt')
get_peak('80_1.txt')
get_peak('80_2.txt')
get_peak('50_1.txt')
get_peak('50_2.txt')
get_peak('50_3.txt')

conc = []; c = 1
for i in [100,80,80,50,50,50]:
    c *= (i/100)
    conc.append(c)
conc = np.array(conc)

fig, ax = plt.subplots()
plt.subplots_adjust(left=0.15)
ax.grid(True)
ax.scatter(conc, strengths)
ax.set_xlabel(r'Concentration (relative to 60 $\mu$l in 2 ml of SYBR1/PBS)')
ax.set_ylabel('Strength of Peak Wavelength in Spectrum')

fig.savefig('../images/dsDNA_conc.png', quality=100)
plt.show()
```

UV vs Blue LED

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

class Spectrum:
    def __init__(self, filename):
        with open(filename) as text:
            lines = text.readlines()
            int_time = float(lines[6].split()[-1])
            wv = []
            strength = []
            for line in lines[14:]:
                line = line.split()
                wv.append(float(line[0]))
                strength.append(float(line[1]))

        self.wavelengths = np.array(wv)
        self.strength = np.array(strength)/int_time

    def plot(self, ax, color, label):
        peaks_ind = sig.find_peaks(self.strength, height=400, distance=70, width=1)[0][:4]
        for i in peaks_ind: ticks.append(self.wavelengths[i])
        ax.set_xticks(ticks)
        ax.plot(self.wavelengths, self.strength, linewidth=0.5, color=color,label=label)




fig, ax = plt.subplots()
ax.axhline(0,linewidth=0.5,linestyle='--', color='black')
ax.set_xlim(340,620)
ax.set_xlabel('Wavelength (nm)')
ax.set_ylabel('Strength (counts/second)')
ax.set_title('Traces of LEDs on x1 SYBR1')
ticks=[]
ax.grid(True)

blue = Spectrum('50_3.txt')
uv = Spectrum('50_3_UV.txt')

blue.plot(ax, 'blue', 'Blue LED')
uv.plot(ax, 'black', 'UV LED')

ax.legend(loc='upper right')
fig.savefig('../images/stain2.png', quality=100)
plt.show()
```