PHYS 403

Lab Report (100 pts) Grading Method

Names_____ Eric Yu, Matthew Fehr ______

CRITERIA	Eugene	Alexey
Science overview (20)		
Procedures (30)		
Results / Analysis (30)		
Technical quality of the report: graphs, figure captions, tables, references, check spelling etc. (20)		
Final Totals (100)	100	100

100

OTHER COMMENTS:

Excellent results and great presentation (ec)

Measuring the Fluorescence Lifetimes of Rubies

Eric Yu, Matthew Fehr

Department of Physics, University of Illinois at Urbana-Champaign

2/15/2023

Abstract

In this experiment, fluorescence spectroscopy was used to determine the fluorescence lifetime and amount of lifetimes of ruby samples. The temperature dependence of the fluorescence lifetime and amount of lifetimes were also studied. The wavelength and frequency were measured and used to find the time- and frequency-domain in different ruby samples, which varied in chromium content by unknown amounts. The results of the experiment showed that the lifetime was temperature dependent with the lifetime increasing as the temperature decreased and that only one of the rubies was shown to have two lifetimes.

Introduction

When a luminescent species absorbs a photon, it is brought to an excited state. The emission of a photon following de-excitation is called luminescence. As shown in Fig. 1, fluorescence is a type of luminescence in which the electron returns to the singlet ground state from the lowest energy singlet excited state (as opposed to phosphorescence, where intersystem crossing takes place and the electron returns from a triplet excited state). In a fluorescent species, the average time an electron spends in an excited state is the fluorescence lifetime. Because there is no intersystem crossing in fluorescence, its lifetimes (nanosecond range) are generally much shorter than phosphorescence lifetimes (millisecond range) [4].

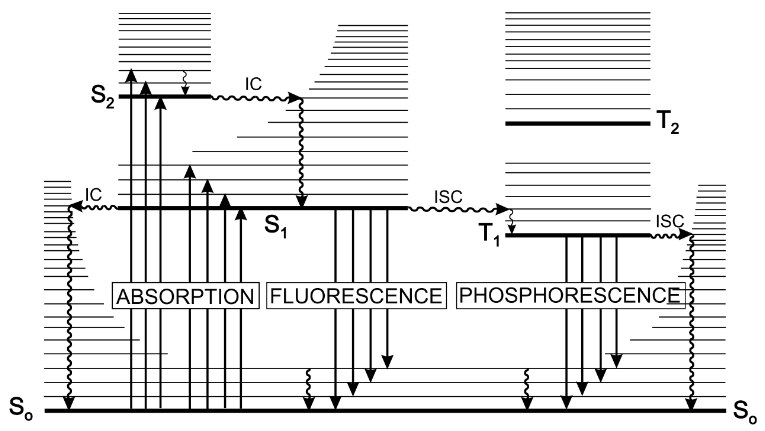


Fig. 1 Perrin-Jablonski diagram. Singlet electronic states are denoted S_0 (ground state), S_1, S_2, and the triplet states T_1, T_2 [4]

Ruby is used in this experiment because it has a fluorescence lifetime in the millisecond range. This relatively long lifetime allows ruby's fluorescence signal to be measured at a good resolution with existing lab equipment [1]. Pure corundum, also known as colorless sapphire, is crystalline Al_2O_3. In ruby, some aluminum ions are replaced by chromium ions, which donate three electrons to nearby O^{2-} ions and become Cr^{3+}. The Cr^{3+} electrons can be excited by ultraviolet, violet, and yellow-to-green light. Upon returning to the ground state, Cr^{3+} fluoresces red light (694 nm), which gives ruby its distinct color. Energy is lost when the electrons settle to the first singlet excited state, which is why the incident photons have a shorter wavelength than the emitted photons (Fig. 2) [3] [5].

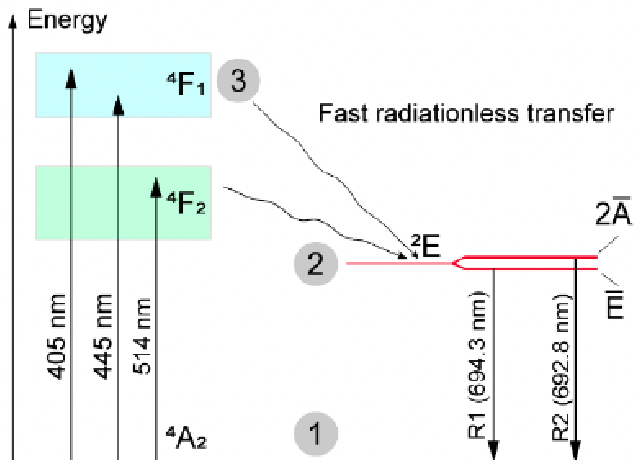


Fig. 2 Perrin-Jablonski diagram for the fluorescent process in ruby [5]

To study the fluorescence lifetime τ of ruby, measurements can be in the time domain and the frequency domain. In the time domain, the response of a single fluorescent species to a delta-function excitation is an exponential decay given by equation 1 (the vertical shift B is added to account for shifts made on the oscilloscope during data acquisition to correctly fit data, this shift is zero in reality) [1].

$$F_\delta(t) = A\exp(-t/\tau) + B \tag{1}$$

In the frequency domain, for a sine wave excitation, the response F and the phase shift relative to the excitation are given by equations 2 and 3. For a square wave excitation, these equations can still be used to carry out analysis if a lock-in amplifier is used in a mode that is “sensitive to the higher harmonics of the repetition frequency” [1].

$$F(\omega) = \frac{A}{\sqrt{1 + (\omega\tau)^2}} \tag{2}$$

$$\phi(\omega) = \arctan(\omega\tau) \tag{3}$$

If there exist multiple fluorescent lifetimes in the sample, then it is easier to resolve secondary lifetimes in the frequency domain. In the case of two lifetimes τ_1 and τ_2, the response and phase shift are given by equations 4 and 5, where α is the observed percentage of the dominant lifetime τ_1 [1].

$$F(\omega) = A\left[\left(\frac{\alpha}{1+(\omega\tau_1)^2} + \frac{1-\alpha}{1+(\omega\tau_2)^2}\right)^2 + \left(\frac{\alpha\omega\tau_1}{1+(\omega\tau_1)^2} + \frac{(1-\alpha)\omega\tau_2}{1+(\omega\tau_2)^2}\right)^2\right]^{1/2} \tag{4}$$

$$\phi(\omega) = \arctan\left[\left(\frac{\alpha\omega\tau_1}{1+(\omega\tau_1)^2} + \frac{(1-\alpha)\omega\tau_2}{1+(\omega\tau_2)^2}\right) / \left(\frac{\alpha}{1+(\omega\tau_1)^2} + \frac{1-\alpha}{1+(\omega\tau_2)^2}\right)\right] \tag{5}$$

In addition, the fluorescence lifetime τ of ruby decreases with an increase in temperature. The increase in thermal interactions at higher temperatures provides more pathways for de-excitation. An increase in temperature also increases intramolecular vibrational interactions, which causes electrons to relax to the lowest energy excited state (S_1 in Fig. 1) more quicker [6]. In the physiological range 288 K to 318 K, this decrease can be precisely modeled linearly, which has led to rubies being used in biological temperature sensors [2].

Procedure

Two different setups were used in this experiment, the time- and frequency-domain setup and the temperature-dependent time-domain setup. Five different rubies (Fig. 3) were tested using these setups. In the time- and frequency-domain setup, each of the five rubies was run three times, with the average of those three runs used in the results.

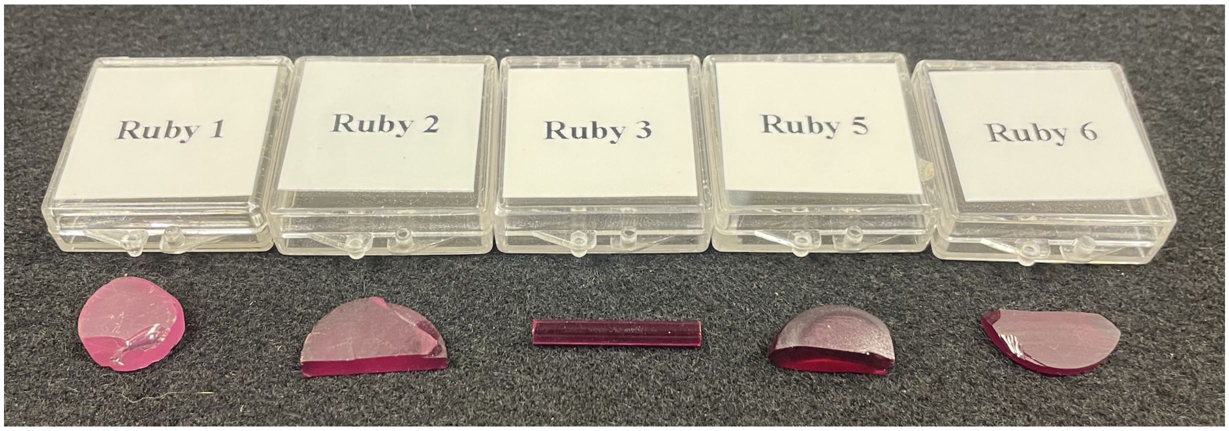
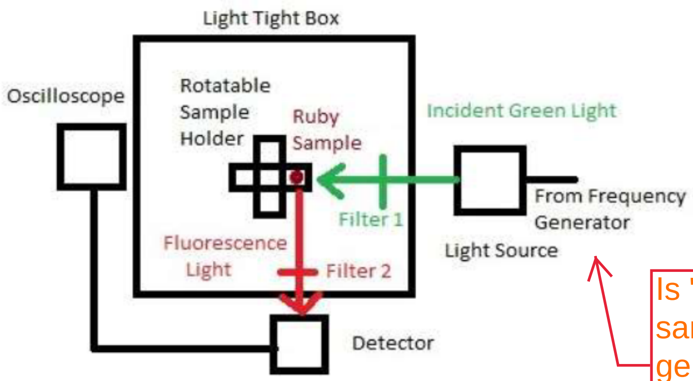


Fig. 3 The different ruby samples used in this experiment are different shades of red, which reflect differing amounts of chromium

Room Temperature Time- and Frequency-domain



Is "frequency generator" the same thing as "waveform generator"?

Fig. 4 A diagram of the time- and frequency-domain setup

In the time-domain setup, a waveform generator was used to generate a pulsating square wave. These waves were used to modulate a 530 nm LED which was then focused on the ruby sample. The fluorescence signal was then captured by the detector, which feeds the data into the oscilloscope as seen in Fig. 4. The final data taken from the oscilloscope will be in the form of a pulsating square wave, which will make it convenient to measure the lifetime as it will appear in the form of the exponential decay section of the square wave.

The frequency-domain setup is similar to the time-domain setup, but for the frequency domain the LED must be modulated by a signal that has a changing frequency. The LED is modulated from 1-150 Hz. This setup makes use of a lock-in amplifier, which helps isolate frequency and reduce noise, as well as for determining the phase shift.

Temperature-Dependent Time-Domain

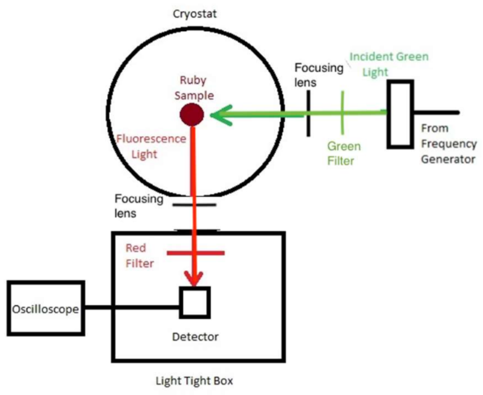


Fig. 5 A diagram of the temperature dependent setup

To measure the temperature dependence, an altered version of the time-domain setup is used to be able to adjust the temperature using a heater and liquid nitrogen (Fig. 5). The same principles from the time-domain setup are used, A square wave-modulated LED is incident on the ruby sample and the fluorescent response is captured by the detector and oscilloscope. The ruby sample is held in a cryostat where a heater can increase the temperature. The cryostat is also connected to a container of liquid helium which is used to cool the ruby. Collecting data was accomplished by heating the cryostat to 400K and then cooling it down to 100K while collecting data for the time domain every 2K.

Results & Analysis

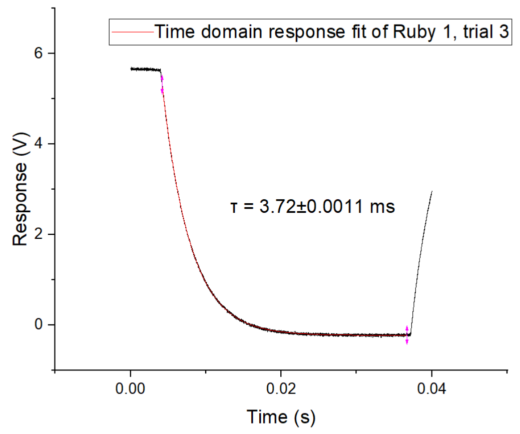
Room Temperature Time- and Frequency-domain

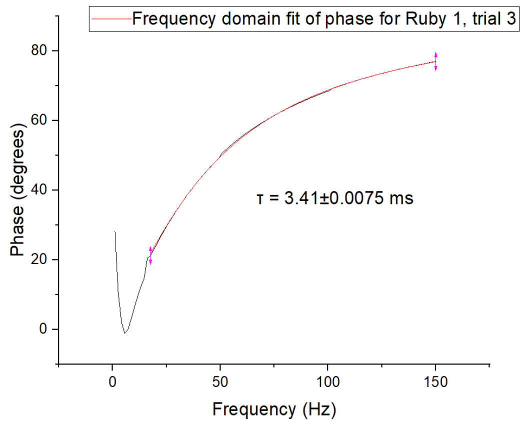
The data collected at room temperature using the time-domain setup was fitted for one lifetime using equation 1. The fitting domain for the exponential decay was always selected to be between the sharp cutoffs, which correspond to when the LED turns off and on again (top left of Fig. 6). Due to the nature of time-domain data combined with factors such as noise and resolution, it is difficult to resolve multiple lifetimes in the time-domain.

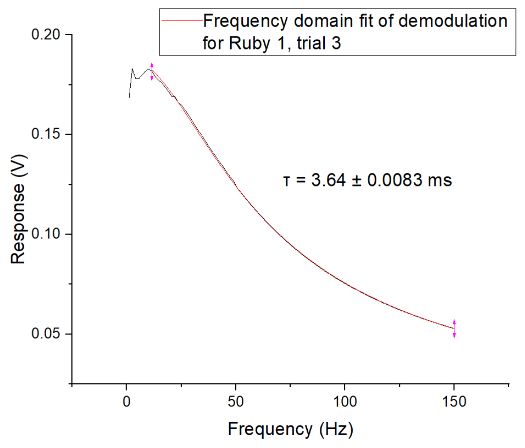
Data collected at room temperature using the frequency domain setup was fitted for one and two lifetimes using equations 2-5. We observed irregular behavior at low frequencies and omitted these from our fitting range. These nonlinear equations can converge to local minima while fitting. To get around this, parameters were initialized to expected values and the fitting was done step by step so we could monitor convergence. For the two-lifetime fittings (equations 4, 5), some rubies were found to converge to nonsensical values (e.g. values greater than one for the fraction parameter α, or negative values for the lifetime) or values absurdly large error bars (10^5 times greater than the value) even after extensive testing. These fittings were omitted. Example fittings in OriginPro are shown in Fig. 6. The calculated fluorescence lifetimes from these fittings are condensed in table 1.

Ruby	τ_{time} (ms)	$\tau_{\text{freq-demod}}$ (ms)	$\tau_{\text{freq-phase}}$ (ms)	$\tau_{\text{slow-demod}}$ (ms)	$\tau_{\text{slow-phase}}$ (ms)
1	3.727±0.0014	3.647±0.0080	3.413±0.0047	- - -	- - -
2	4.353±0.0019	4.153±0.0097	4.313±0.0091	- - -	4.27±0.0013
3	4.553±0.0029	4.130±0.0234	4.173±0.0093	4.31±0.0231	4.29±0.0598
5	3.990±0.0019	3.713±0.0211	4.123±0.0136	4.18±0.0025	4.00 ±0.0143
6	4.080±0.0031	3.827±0.0172	4.177±0.0120	- - -	4.13 ±0.0147

Table 1: Room temperature time- and frequency-domain fluorescence lifetimes Column 1 is the ruby sample number. Columns 2-4 are the τ from equations 1-3 respectively. Columns 5 and 6 are the longer lifetimes from equations 4 and 5. Center dashes '- - -' mean that a good fit was not achieved for that specific τ.







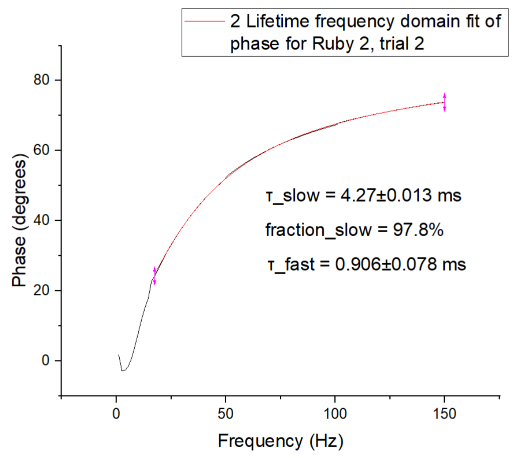


Fig. 6 Example of fitted curves (red) to collected data (black) in OriginPro

In table 2, more details about the fittings for rubies 3 and 5. For these rubies, we could fit the frequency domain data to both one and two lifetimes using both the demodulation and phase.

Ruby	τ (ms)	$R^2_{\text{1-lifetime}}$	τ_{slow} (ms)	α_{slow} (%)	τ_{fast} (ms)	$R^2_{\text{2-lifetimes}}$
3-demod	4.153±0.0097	0.99980	4.31±0.0231	95.8±1.1	0.406±0.274	0.99993
3-phase	4.173±0.0093	0.99981	4.29±0.0598	94.3±3.3	2.38 ±0.353	0.99988
5-demod	3.713±0.021	0.99831	4.123±0.0136	91.9±.0.16	0.705±0.0074	0.99995
5-phase	4.123±0.0136	0.99951	4.00 ±0.0143	93.7±0.28	0.988±0.0339	0.99986

Table 2: Comparison of one- versus two-lifetime fitting of frequency-domain data
The first column refers to the ruby sample and the fitting method (demodulation is eq. 2 and 4, phase is equations 3 and 5). The second and third columns are single lifetime fittings (eq. 2 and 3) and the fourth through seventh columns are the two-lifetime fittings (eq. 4 and 5).

As evident from the greater R^2 values when fitting frequency-domain data to two lifetimes, a clear second lifetime is present in rubies 3 and 5. When fitting the frequency-domain of rubies 1, 2, and 6 to two lifetimes, the failure of convergence to precise and sensical values can be interpreted as a nonexistent second lifetime for those samples. The lack of agreement between lifetime measurements in the time and frequency domain may be attributed to an untuned lock-in amplifier. As mentioned earlier, the frequency domain fitting equations rely on a properly tuned lock-in amplifier to select the higher harmonics of the signal if a square wave source is used. This is something we did not catch while collecting data.

R2 value can not be used for making statement about the number of the lifetimes - more fitting parameters - better fitting. In your case the results of the analysis for ruby3 and 5 look realistic.

Temperature-Dependent Time-Domain

Since there were hundreds of data files for the temperature dependent experiment, a Python library was used to fit each data file with equation 1 so an automatic script could be used instead of fitting each file by hand in OriginPro (specifically Scipy.optimize). The script used is in the appendix. Plots of the fluorescence lifetime versus temperature for rubies 1 and 2 are shown in Fig. 7.

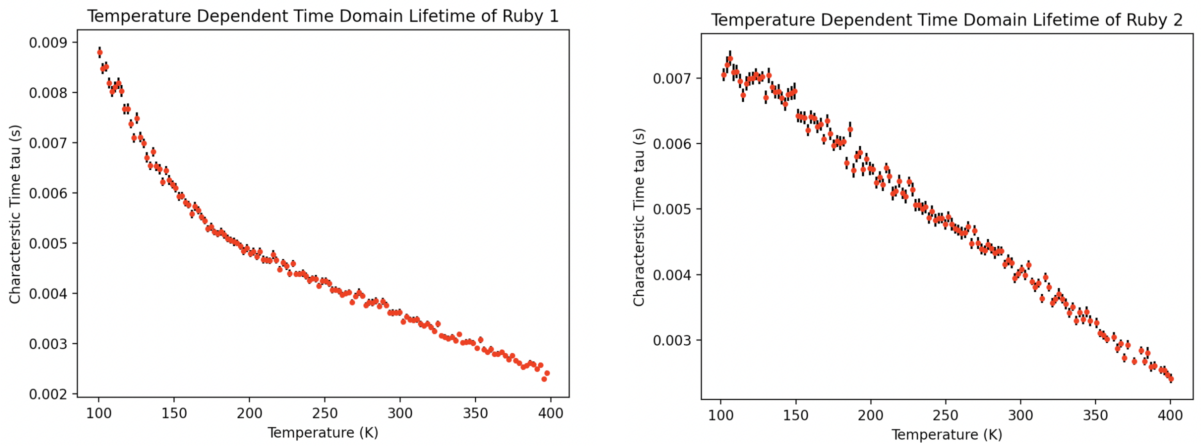


Fig. 7 Fluorescence lifetime versus temperature in the time-domain (black bars above and below each point are the standard deviation)

As shown by Fig. 7, the relationship between the fluorescence lifetime and temperature is linear in and well beyond the physiological range of 288 K to 318 K. It is interesting to note that ruby 1 deviates from linearity in the lower part of the temperature range, but has very small error bars and appears to be better described by a line in the temperature that looks linear. Ruby 2, on the other hand, seems to be linear for the entire temperature range, but has larger error bars. Though ruby 2 is linear from 100 K to 400 K, rubies like ruby 1 may be more useful for physiological applications because the fluorescence lifetimes in the physiological temperature range are more precise. Future studies might study the temperature dependence of fluorescence lifetime for rubies with specifically known chromium concentrations.

Conclusions

The experiment was successful in showing the linear temperature dependence of fluorescence lifetime at the physiological temperature range. The analysis shows that the fluorescence lifetime increases as the temperature decreases. The experiment also presented results that suggest that only two of the rubies are modeled best using two lifetimes.

References

1. D. Chandler, Z. Majumdar, et al. Ruby crystal for demonstrating time- and frequency-domain methods of fluorescence lifetime measurements. *Journal of Fluorescence* (2006). doi:10.1007/s10895-006-0123-7
2. J. Alcala, S-C. Liao, and J. Zheng. Real time frequency domain fibreoptic temperature sensor using ruby crystals. *Medical Engineering and Physics* (1996). doi:10.1016/1350-4533(95)00014-3
3. B. V. Thosar. On the Fluorescent Ion of Chromium in Ruby. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1938). doi:10.1080/14786443808562133
4. B. Valeur and M. Berberan-Santos. Molecule Fluorescence: Principles and Applications. *John Wiley and Sons* (2012)
5. Ruby Crystal Fluorescence. *PhysicsOpenLab* (2020), link:https://physicsopenlab.org/2020/06/15/ruby-crystal-fluorescence/
6. A. Periasamy and R. Clegg. FLIM Microscopy in Biology and Medicine. *Taylor and Francis Group* (2009)

Appendix

The python code used to fit the temperature dependent time-domain data is pasted below. A portion of the code is adapted from a window selecting function written by Lucas Slattery from a past semester. The same code with detailed comments on how to run the code can be found at the link below.
https://gist.github.com/aeric-underscore/43b732367793eac014725c4bcdcd04bc

```
import numpy as np
import torch
import matplotlib.pyplot as plt
import scipy as sp
import scipy.optimize
import os

# written by eric yu
def main():
   # # # # parameters # # # # # # # # # # # # # # # # # # # # # # # # # # #
   curr = "/sample/path/"
   file_dir_1 = "/sample/path/"
   file_dir_2 = "/sample/path/"
   file_dir = file_dir_1
   test_fit_file = "file.dat"
   test_fit_flag = False
```

```
    gen_data_flag = False
    out_file = “file.txt"
    in_file = "file.txt"
    # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
    if test_fit_flag:
        fit(file_dir + test_fit_file,[],[],[],float(test_fit_file[5:10]))
    else:
        if gen_data_flag:
            Ts, taus, errs = gen_data(file_dir)
            np.savetxt(out_file, np.array([Ts, taus, errs]).T)
        else:
            data = np.loadtxt(in_file, dtype="float")
            Ts = data[:,0]; taus = data[:,1]; errs = data[:,2]
        # # # plotting code # # #
        plt.errorbar(Ts, taus, fmt='.k', yerr=errs, c='r', ecolor='k')
        plt.title("Temperature Dependent Time Domain Lifetime of Ruby 2")
        plt.xlabel("Temperature (K)"); plt.ylabel("Characterstic Time tau (s)")
        plt.show()

def fit(file_name, Ts, taus, errs, T):
    t, r = get_time_domain_dat(file_name)
    N = len(t)
    left = 3*N//7; right = N - 1
    t0 = t[left:right]; r0 = r[left:right];
    # # # test plotting code # # #
    # plt.clf(); plt.scatter(t,r);
    # plt.axvline(x=t[left],c='g'); plt.axvline(x=t[right],c='g'); plt.show()

    new_first_t = t0[0]; t0 = t0 - new_first_t
    r_max = max(r0); r_min = min(r0)
    start, stop = find_fit_range(r0, 0.75*r_max, r_min+0.10*r_max)
    t1 = t0[start]; t2 = t0[stop]
    if stop - start > 9:
        try:
            popt, pcov = sp.optimize.curve_fit(exp_dec_func, t0[start:stop],
r0[start:stop], p0=(r_max-r_min, 3e-3, r_min))
            A = popt[0]; tau = popt[1]; B = popt[2]
            perr = np.sqrt(np.diag(pcov))
            tau_err = perr[1]
            Ts.append(T); taus.append(tau); errs.append(tau_err)
            # # # plotting code for troubleshooting fit # # #
            # plt.clf(); plt.scatter(t0,r0);
            # plt.plot(t0,exp_dec_func(t0,A,tau,B),c='k')
            # plt.axvline(x=t1,c='r'); plt.axvline(x=t2,c='r')
            # plt.ylim(0, 1.25*r_max); plt.show()
        except Exception as e: print(e)

def gen_data(file_dir):
    file_list = os.listdir(file_dir)
    Ts = []; taus = []; errs = []
    for f_name in file_list:
        T = float(f_name[5:10]); print(T)
        fit(file_dir + f_name, Ts, taus, errs, T)
    return Ts, taus, errs

def get_time_domain_dat(filename):
    data = np.loadtxt(filename, dtype='float')
    t = data[:,0]; r = data[:,1]
    return t, r

#find_fit_range adapted from a script written by Lucas Slattery
#found in the PHYS 403 folder
def find_fit_range(r, r1, r2):
    buffer = 10
    x = np.array([int(i) for i in r > r1])
    y = np.array([int(i) for i in r > r2])
    b = np.array(buffer*[1])
    above_r1 = [(i, i+len(b)) for i in range(len(x)-len(b))
        if np.all(x[i:i+len(b)] == b)][0][0]
    b = np.array(buffer*[0])
    below_r1 = [(i, i+len(b)) for i in range(len(x[above_r1:])-len(b))
        if np.all(x[above_r1+i:above_r1+i+len(b)] == b)][0][0]
    b = np.array(buffer*[0])
    below_r2 = [(i, i+len(b)) for i in range(len(y[above_r1+below_r1:])-len(b))
        if np.all(y[above_r1+below_r1+i:above_r1+below_r1+i+len(b)] == b)][0][0]
    start = above_r1+below_r1
    stop = above_r1+below_r1+below_r2
    return start, stop

def exp_dec_func(t, A, tau, B):
    return A * np.exp(-t/tau) + B

if __name__ == '__main__':
    main()
```