# HYDROGEN DIFFUSION IN INDIUM OXIDE

Hui Xu

## *ABSTRACT*

*Hydrogen is an n-type dopant in In2O3, a conducting oxide. Hydrogen is an important source of n-type conductivity in In2O3 and has been shown by studies to be a shallow donor.*[1]

*Using data from an IR absorption experiment*[2] *that studied a vibrational line at 3306cm$^{-1}$ using Fourier Transform InfraRed spectrometry, hydrogen's diffusivity is examined in this work. Hydrogen's diffusivity is studied through its in-diffusion and out-diffusion in In2O3 single crystals*[3]*. The diffusion process has been simulated in Python language by solving Fick's second law with the appropriate boundary conditions. Hydrogen in-diffusion data was fitted using Scipy*[4] *curve fit function. Fick's second law, a second*-order *partial differential equation, has been solved using Fipy*[5]*, a partial differential equation (PDE) solver written in Python. The simulation result was then fitted to published experimental data3 through regression and error analysis to determine the diffusivity of $H^{i+}$ in $In_2O_3$.*

## *KEYWORDS*

*Diffusion, hydrogen, temperature, oxide*

## 1. INTRODUCTION

Transparent conducting oxides (TCOs) are transparent electrodes that combine high conductivity with transparency in the visible spectrum. $In_2O_3$ is a prototypical TCO with a cubic bixbyite structure with a unit cell that contains 80 atoms[6]. Since TCOs are also wide band-gap semiconductors, they have impurities from defects or dopants. These impurities act as recombination centers due to their concentrated free electrons in the conduction band and can be used to emit light or act as artificial atoms. TCOs have long been used in circuitry, windows, and transparent electrical contacts.

Hydrogen is an impurity in $In_2O_3$ that introduces electrical levels in the bandgap of semiconductors as an important source of conductivity. As an interstitial shallow donor, hydrogen is an important source of n-type conductivity in $In_2O_3$. Hydrogen modifies the properties of electronic devices and semiconductors.

The purpose of this paper is to determine the diffusivity of hydrogen in $In_2O_3$ by finding diffusion constants in order to understand the physics of conductivity in TCOs. In Fick's second law, the diffusion constant, D, is a proportionality constant between the diffusion flux and the concentration gradient of the species, hydrogen. D is dependent on pressure and temperature, so a diffusion constant was found at multiple temperatures.

Such work can provide insight into the effect of hydrogen on conductivity. The hydrogen diffusion process is simulated by solving Fick's second law, written in Python with Scipy and Fipy libraries, and using published experimental results[1] to determine the diffusivity of $H^{i+}$ in $In_2O_3$. Scipy is a library for mathematical functions with modules for scientific programming. FiPy was used to fit in-diffusion data with an error function and solve a partial differential equation.

[7]SciPy (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

FiPy[3], an object-oriented, partial differential equation (PDE) solver, written in Python, based on a standard finite volume (FV) approach. The approach combines Python and FV to create an extensible tool with a framework that has terms for diffusion and standard sources.

## 2. HYDROGEN’S DIFFUSIVITY IN $In_2O_3$

### 2.1. In-Diffusion

The in-diffusion and out-diffusion behaviors of hydrogen can be used to find hydrogen’s diffusivity in $In_2O_3$.

The in-diffusion experiments[3] utilized hydrogenated $In_2O_3$ samples by annealing them in $N_2$ at 1273 K to eliminate any present H and then in $H_2$ ambient at nearly 700K. Layers were thinned and the absorbance spectra were measured after the removal of each layer.

The in-diffusion data[2] from the published experiment consists of data points of the integrated absorbance at each thickness removed. Fick’s second law models the in-diffusion data. The analytical solution of Fick’s second law with the in-diffusion boundary condition, constant surface concentration, is a complementary error function, “special.erfc” from SciPy (Eq. 1).

$$\text{special.erfc()} = \left( 2/(\text{pi})^{½} \right) * \int (e^{\wedge}(-t^2))\, dt,\ t=x..\infty \tag{1}$$

“special.erfc” was integrated vs. the thickness removed from the layers, as shown in Fig. 1.

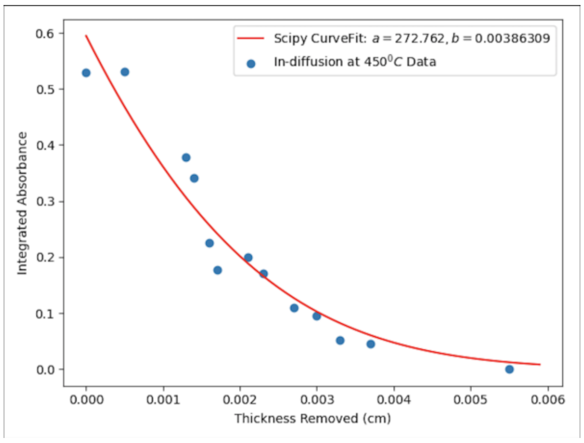

Fig. 1: A fit of the integral of an error function vs. the thickness removed by thinning

Fig. 1 was fitted in PyCharm, an integrated development environment for python8 with Scipy curve fit, “curve_fit”. Using “curve_fit”, the complementary error function was fit with the data, and the coefficients for the complementary error function were found, which are also the boundary conditions for the initial state of out-diffusion. As seen in Fig. 1, a = 272.762, and b = 0.00386, and the square root of the mean error between the data and the analytical solution is 0.4114. The complementary error function was used with a and b as the initial condition in solving Fick’s second law for hydrogen out-diffusion. The code is in Appendix I.

## 2.2. Out-Diffusion

The out-diffusion of hydrogen was investigated by modeling it in PyCharm, and the code is in Appendix II. The in-diffusion equation for the fit curve yielded the initial state of out-diffusion. Therefore, the initial condition of hydrogen concentration (phi) was defined as in Eq. 2, where phi is the function of concentration.

```
phi = CellVariable(name='solution variable', mesh=mesh,                                    (2)
            value=272.762 * (erfc(x / 0.00386309) + erfc((x2 - x) / 0.00386309)))
```

Fick's second Law, as in Eq. 3, was solved to simulate the hydrogen diffusion process with the boundary conditions for out-diffusion and find the diffusion constant over time.

$$\partial c / \partial t = D\,(\partial^2 c / \partial x^2) \tag{3}$$

c = concentration, t = time, x = diffusion depth, D = diffusion coefficient

Using the out-diffusion data[2] at different annealing temperatures, the integrated absorbance vs. annealing time (h) was plotted in Fig. 3. To obtain the fit line for Fig. 3, Fick's second Law was integrated, and diffusion constants were found for each data set as seen in Table 1.

To find the diffusion constant, D, for different temperatures, a for loop was used to check the mean square error between the data and the fit line for each D within a range. In PyCharm, the range was slowly decreased, and so was the possible error ("smallest error"), the mean square error of the out-diffusion data, and the simulated integrated absorbance. The "best diffusion constant" was found within the selected range and had an error less than the "smallest error".

## 3. Results

The "best diffusion constant" was selected by slowly decreasing the range to decrease the error calculated. Table 1 shows the constant at each temperature found, and the constants generally decreased with temperature.

Table 1. Diffusion Constants

| Temperature (°C) | Best Diffusion Constant ($10^{-6}$ cm$^2$/s) |
|---|---|
| 400 | 2.30 |
| 375 | 1.00 |
| 350 | 1.80 |
| 325 | 0.220 |

In Figs. 2-5, the absorbance vs. annealing time graphs are shown for each temperature. As the temperature decreases, the annealing time increases, and the slope decreases. Hydrogen diffuses the quickest at 400°C, as seen in Fig. 2, and it has the greatest diffusion constant. These results indicate that higher temperatures increase the diffusivity and mobility of hydrogen in $In_2O_3$.

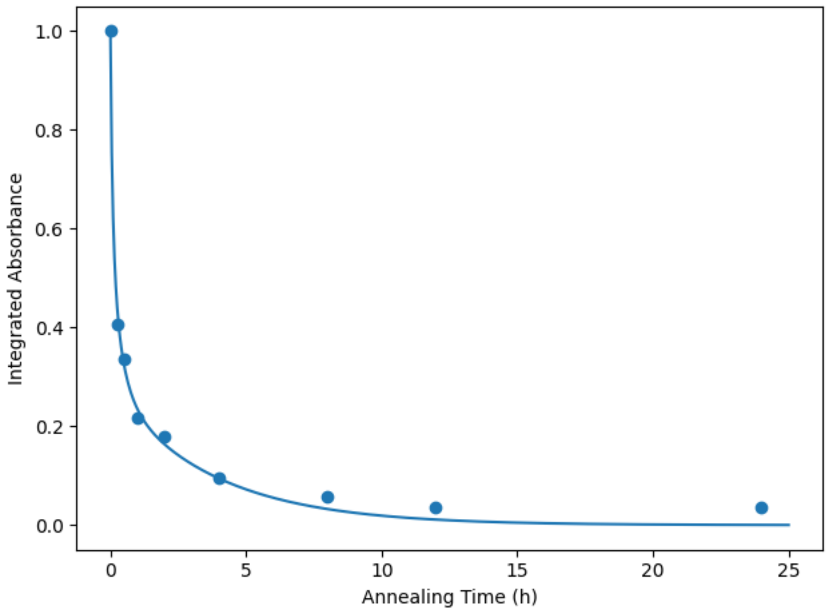

Fig. 2: The integrated absorbance vs. annealing time at $325^0$C

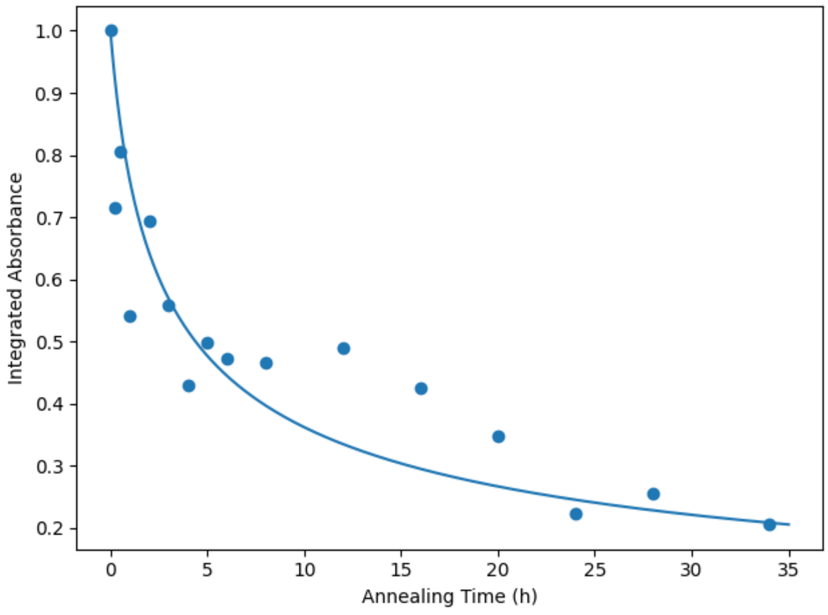

Fig. 3: The integrated absorbance vs. annealing time at $375^0$C

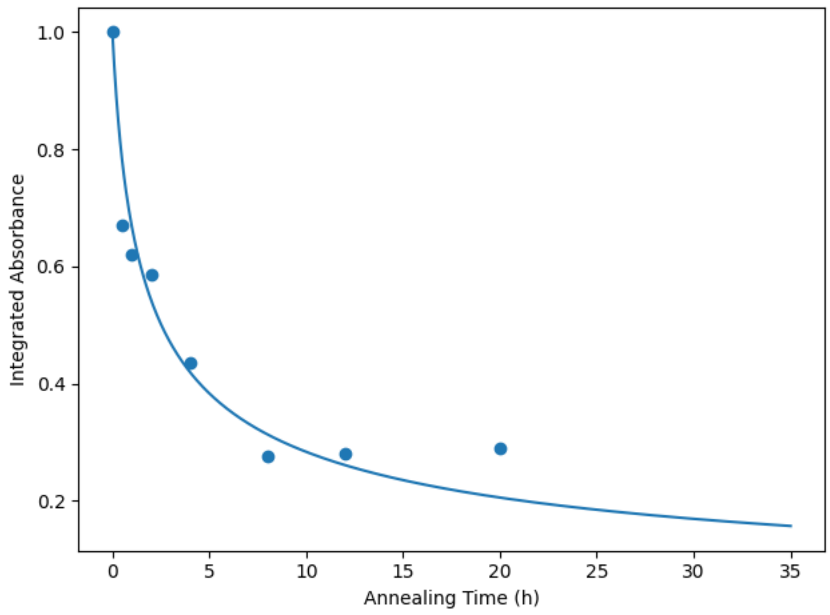

Fig. 4: The integrated absorbance vs. annealing time at $350^0$C

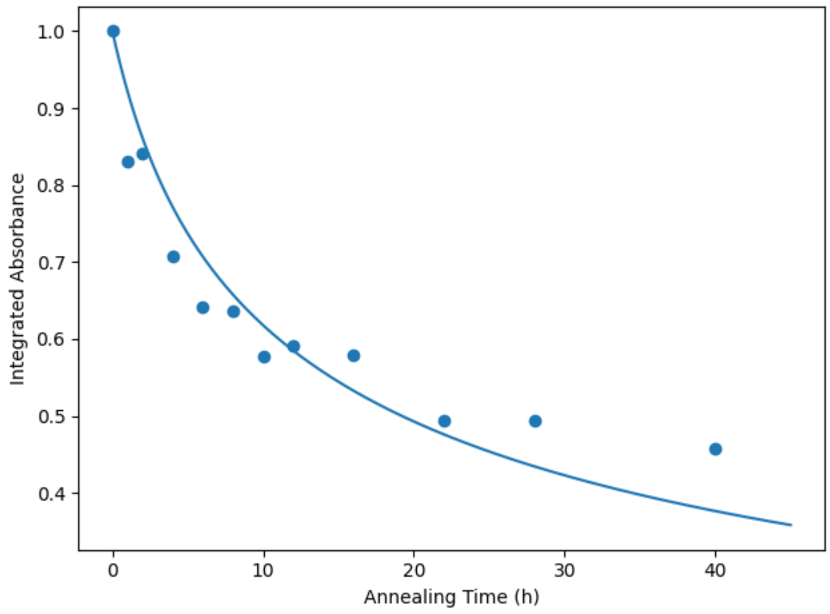

Fig. 5: The integrated absorbance vs. annealing time at $325^0$C

## 4. Conclusions

$In_2O_3$ is a conducting oxide that has many applications in electronic devices. Hydrogen gives unintentional conductivity to $In_2O_3$ and is an n-type dopant. Published experimental results were used to obtain the analytical(in-diffusion) and numerical(out-diffusion) solution of Fick's second law. The complementary error function was integrated vs. the thickness of the layers, and the boundary conditions for out-diffusion were found from the in-diffusion fit. The boundary conditions were used as the initial state of out-diffusion and the out-diffusion process was simulated by numerically solving Fick's second law using Fipy, a partial differential equation solver written in Python. The diffusion constants were obtained by fitting the plot of annealing time vs. the integration absorbance to the numerical solution of Fick's second law while achieving the smallest mean square error.

Annealing $In_2O_3$ in hydrogen at 400°C is the most effective temperature to maximize the effect of hydrogen on conductivity as a shallow donor. The temperature that hydrogen annealed the quickest in was 400°C, and it correspondingly had the greatest diffusion constant, 2.30 x $10^{-6}$ $cm^2/s$, at 400°C. According to the Beer-Lambert law, absorbance is proportional to concentration, so the integrated absorbance graphs depict the concentration of hydrogen. The fit lines obtained by integrating Fick's second law in Figs. 2-5 indicate that, at 400°C, the integrated absorbance decreases at the greatest rate, so hydrogen diffuses the quickest at this temperature. These results confirm the Arrhenius equation (4), in which the diffusion constant is positively correlated with temperature, since the greatest diffusion constant was found at the greatest temperature measured, 400°C.

$$D = Do * \exp(- EA / RT) \tag{4}$$

D = diffusion constant, = activation energy, T = absolute temperature, R = universal gas constant

## 5. DISCUSSION

This research focuses on the physics of conductivity in TCOs and investigates the effect of the hydrogen impurity and defect complexes in technologically important prototypical oxide materials[2]. Hydrogen impurities have recently been recognized as an important source of conductivity. There have been exciting theoretical predictions about conductivity in semiconducting oxides and hydrogen's role in it, but there is only a limited experimental understanding. The diffusion of defects and atoms are at the core of semiconductor processing. The diffusivity of hydrogen in $In_2O_3$ describes the movement of atoms in $In_2O_3$, and the mobility of a defect is defined by the diffusion constant, D[9]. D was generally found to obey an Arrhenius relation, in which kinetic energy increases with temperature, and, thus, hydrogen's mobility.

In this work, the assumption was made that hydrogen diffusion is one dimensional. Using the areal density of hydrogen, hydrogen was assumed to be homogenous on the cross sections of the sample. The experiment was limited by the temperature of the furnace, which was only accurate to the first decimal, so there could have been temperature fluctuations. The modeling of hydrogen diffusion was limited by the error. The program iterates through each possible diffusion constant to select the constant with the smallest least square error between the data and the numerical solution to Fick's second law for that constant. Further experimentation to collect data would optimize the results.

Gaining insight into the diffusion of hydrogen experimentally is crucial to the use and application of hydrogen impurities in TCOs. The use of hydrogen in TCOs can enhance the conductive and electric properties of oxides. Hydrogen has many applications in TCOs, particularly in $In_2O_3$. Due to its potential applications in lighting and display, $In_2O_3$ has many technological implications in electronic devices. $In_2O_3$ can be used in flat-panel displays, conductive films, light-emitting diodes, and solar cells. With the addition of a hydrogen impurity, $In_2O_3$ conductive properties can be enhanced. Hydrogen introduces electrical levels and is responsible for n-type conductivity in $In_2O_3$ and other TCOs.

Future extensions of this work are finding the diffusivity of other solid-state materials. These results can be compared and assessed to find which materials are best suited for applications in fields like technology, lighting, and energy.

## ACKNOWLEDGEMENTS

I would like to thank Dr. Ying Qin Taylor from Lehigh University for mentoring me and providing me with her patient guidance. Her feedback and support have been absolutely indispensable. I am also very grateful to my high school teachers for all of their invaluable lessons.

## REFERENCES

[1] Limpijumnong, S., Reunchan, P., Janotti, A., & Van de Walle, C. G. (2009). Hydrogen doping in indium oxide: An ab initio study. Physical Review B, 80(19), 193202, https://link.aps.org/doi/10.1103/PhysRevB.80.193202.

[2] Yin, W., Smithe, K., Weiser, P., Stavola, M., Fowler, W. B., Boatner, L., ... & Koch, S. G. (2015). Hydrogen centers and the conductivity of In2O3 single crystals. Physical Review B, 91(7), 075208, https://link.aps.org/doi/10.1103/PhysRevB.91.075208.

[3] Qin, Y., Weiser, P., Villalta, K., Stavola, M., Fowler, W. B., Biaggio, I., & Boatner, L. (2018). Diffusivity of the interstitial hydrogen shallow donor in In2O3. Journal of Applied Physics, 123(16), 161506, https://aip.scitation.org/doi/am-pdf/10.1063/1.4995593.

[4] Scipy.optimize.curve_fit https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

[5] FiPy: A Finite Volume PDE Solver Using Python. https://www.ctcms.nist.gov/fipy/

[6] I. Hamberg and C. G. Granqvist, J. Appl. Phys. 60, R123 (1986).

[7] https://scipy.org

[8] JetBrains, S. R. O. "PyCharm: The Python IDE for Professional Developers." https://www.jetbrains.com/pycharm/. Accessed 20 Jun. 2020.

[9] Shaw, D. (1973). General features of diffusion in semiconductors. In Atomic Diffusion in Semiconductors (pp. 1-63). Springer, Boston, MA.

## APPENDIX I: IN-DIFFUSION

```
indif450 = [(0.0, 0.52887013), (0.0005, 0.5316378550), (0.0013, 0.3774652950),
        (0.0014, 0.341124080),
        (0.0016, 0.225380820), (0.0017, 0.177666440), (0.0021, 0.199492530),
        (0.0023, 0.171156076),
        (0.0027, 0.11020440), (0.003, 0.094437730), (0.0033, 0.051531765),
        (0.0037, 0.044612611),
        (0.0055, 0)]
x2 = 0.0515
from scipy import special
from scipy import integrate
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
a = 272.767
b = 0.00386304

def ana_soln(x, i, j):

  def erfc(x):
    return special.erfc(x / j)
  n = []
  for x1 in x:
     n.append(i * integrate.quad(erfc, x1, x2)[0])
  return n
x_data = []
y_data = []

for point in indif450:
  x_data.append(point[0])
  y_data.append(point[1])

popt, pcov = curve_fit(ana_soln, x_data, y_data, bounds=(0, [1000., 1.0]))
print(popt, pcov)
# pcov is a 2d array: estimated covariance of popt, diagonals are variance of the parameter estimate.
perr = np.sqrt(np.diag(pcov))  # one standard deviation errors on the parameters
print(perr)
print("a =", popt[0], "+/-", pcov[0, 0]**0.5)
print("b =", popt[1], "+/-", pcov[1, 1]**0.5)
sum_sqr = 0.0

for point in indif450:
  q = 0
  sum_sqr += (point[1] - ana_soln(x_data, *popt)[q]) ** 2
  q += 1

sqrterr = np.sqrt(sum_sqr / len(x_data))
print('sqrt of the mean square error between data and analytical solution is: ', sqrterr)
sum_sqr1 = 0.0

for point in indif450:
  q = 0
  sum_sqr1 += (point[1] - ana_soln(x_data, a, b)[q]) ** 2
  q += 1

sqrterr1 = np.sqrt(sum_sqr1 / len(x_data))
print('sqrt of the mean square error between data and analytical solution from Mathematica is: ', sqrterr1)
plt.scatter(*zip(*indif450), label='In-diffusion at $450^0C$ Data')
plt.xlabel('Thickness Removed (cm)')
plt.ylabel('Integrated Absorbance')

x = np.arange(0, 0.006, 0.0001)
plt.plot(x, ana_soln(x, *popt), color='red',
      label=('Scipy CurveFit: $a = {{{}}}, b = {{{}}}$').format(round(popt[0], 3),
                                      round(popt[1], 9)))

plt.legend()
plt.savefig('indif450_fit2.png')
plt.show()
exit()
```

## APPENDIX II: OUT-DIFFUSION

```
from fipy import *
from fipy.tools import numerix
from scipy.special import erfc
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import simps
from numpy import trapz
from builtins import range
outdif4001 = ((0, 1), (0.25, 0.809697915), (0.5, 0.723727356), (1, 0.646647227), (2, 0.467862744), (4,
0.42251084), (8, 0.323907231), (12, 0.229748172), (24, 0.150810664), (36, 0.155564008))
        // “outdif4001” is the data set for each temperature, so it changes for each temperature
nx = 50
dx = 0.001
x2 = nx * dx
valueLeft = 0
valueRight = 0
D = 1.112 * 10 ** (-6)
timeStepDuration = 0.1 * dx ** 2 / (2 * 10 ** (-6))
print('timeStepDuration: ', timeStepDuration)
steps = 800
// “steps” varies for each data set based on how long it took hydrogen to diffuse at each
temperature
t = timeStepDuration * steps
print('total t: ', t)

def pde_solver(d):
  mesh = Grid1D(nx=nx, dx=dx)
  x = mesh.cellCenters[0]
  phi = CellVariable(name='solution variable', mesh=mesh,
              value=272.762 * (erfc(x / 0.00386309) + erfc((x2 -
                                                     x) / 0.00386309)))
  phi.constrain(valueRight, mesh.facesRight)
  phi.constrain(valueLeft, mesh.facesLeft)

  eqI = TransientTerm() == DiffusionTerm(coeff=d)
  results = []
  for step in range(steps):
    eqI.solve(var=phi, dt=timeStepDuration)
    results.append(phi.value.copy())
  return results

def integr_results(d):
  results = pde_solver(d)
  results_integx = []
  norm = trapz(results[0], dx=dx)
  for t1 in range(steps):
    results_integx.append(trapz(results[t1], dx=dx) / norm)
    t1 += 1
  return results_integx

def cal_err(d, data=outdif4001):
  fit = integr_results(d)
  square = 0.0
  for point in data:
    square += (point[1] - fit[round(point[0]/timeStepDuration)]) ** 2
  mean_square = square / len(data)
  return mean_square

smallest_error = 0.000722
print('error with D', cal_err(D))
best_d = 0.0
```

```
possible_D = [d for d in np.arange(2.3 * 10 ** (-6), 2.3001 * 10 ** (-6), 10 ** (-9))]
              // "possible_D" changes
for d in possible_D:
   errorD = cal_err(d)
   if errorD < smallest_error:
       best_d = d
       print('d:', d, 'errorD:', errorD)
       smallest_error = errorD
print('best d: ', best_d, '\nsmallest error: ', smallest_error)

plt.scatter(*zip(*outdif4001))
plt.xlabel('Annealing Time (h)')
plt.ylabel('Integrated Absorbance')
x = np.linspace(0, t, steps)
plt.plot(x, integr_results(best_d))
plt.savefig('outdiftest.png')
plt.show()
exit()
```

## AUTHOR

Hui Xu is a high school senior who enjoys studying engineering and technology. She is passionate about environmental and social issues, and she aspires to create solutions to these issues through her work.