cheg325 homework7 SIS 11.5-8

AUTHOR
kyle wodehouse

PUBLISHED
April 9, 2025

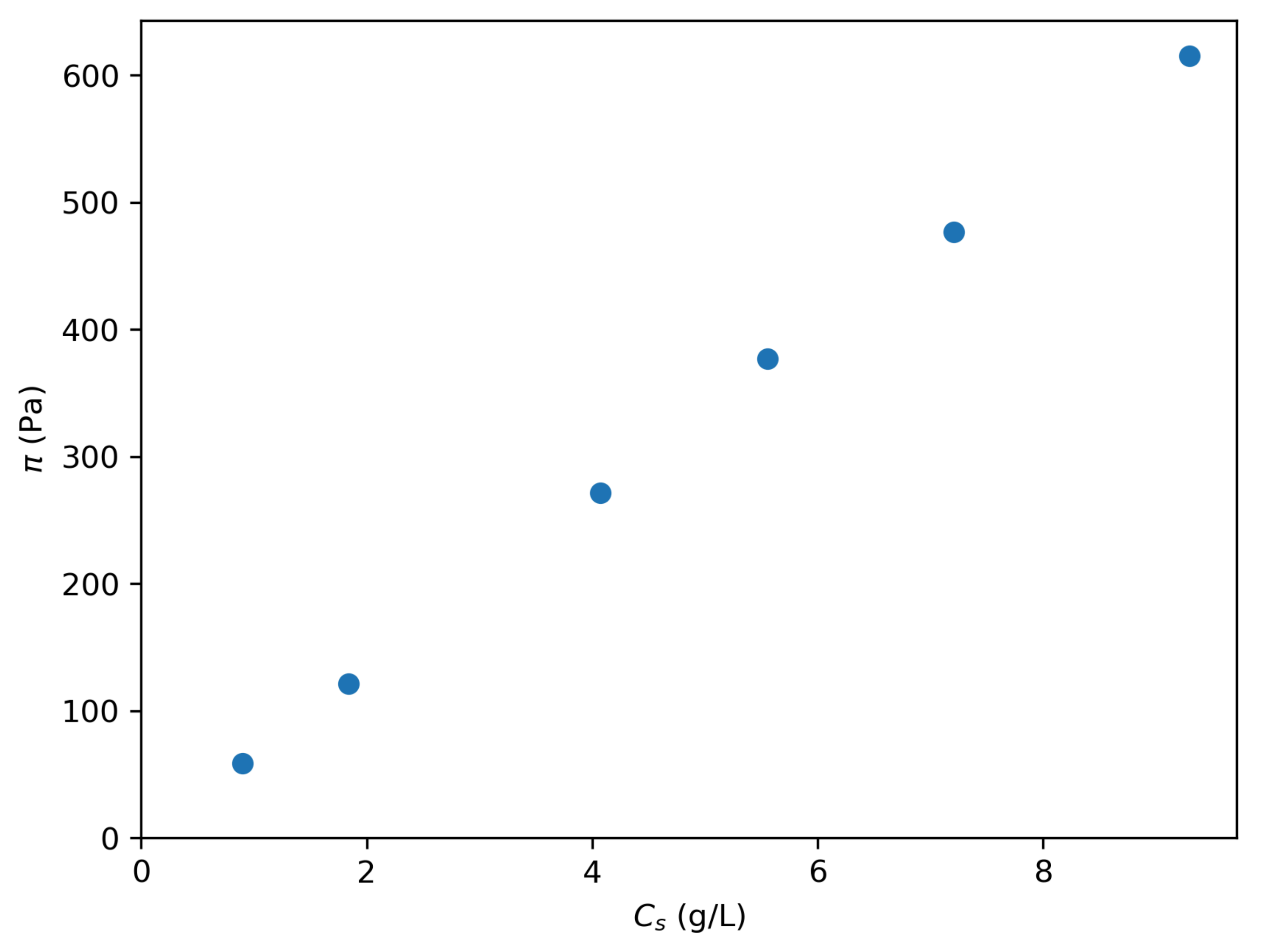
first let's get our data into the notebook and see what it looks like

```
import pandas as pd
from scipy.constants import R
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import numpy as np

T = 273.15 + 25.0
df = pd.read_csv('data.txt', sep=' ')
df.head(3)
```

	Cs	pi
0	0.901	58.7
1	1.841	121.2
2	4.070	271.3

lets quickly visualize it to get an ideal of how big we should expect B_2 to be. if it looks pretty linear we'll expect B_2 to be pretty small

```
x = np.array(df['Cs'])
y = np.array(df['pi'])

fig, ax = plt.subplots(dpi=300)
ax.scatter(x, y)
ax.set(xlim=0, ylim=0, xlabel='$C_s$ (g/L)', ylabel='$\pi$ (Pa)');
```



the equation we want to fit our experimental data to is this

$$\frac{\pi}{RT} = \frac{C_s}{M}\left[1 + B_2\left(\frac{C_s}{M}\right)\right]$$

and we can actually just do this with straight up curve fitting if we rearrange slightly

$$\pi = RT\frac{C_s}{M}\left[1 + B_2\left(\frac{C_s}{M}\right)\right]$$

let's convert all our units to SI units so they're easy to discern at the end

$$\frac{\mathrm{g}}{\mathrm{L}} \times \frac{1\mathrm{kg}}{1000\mathrm{g}} \times \frac{1000\,\mathrm{L}}{1\,\mathrm{m}^3} = \frac{1\,\mathrm{kg}}{1\,\mathrm{m}^3}$$

now we can just simply curve fit and know our molecular mass will be in kg/mol. and in order to figure out the units on the fitted B_2 we can multiply out the π equation

$$\pi = RT\frac{C_s}{M} + RTB_2\left(\frac{C_s}{M}\right)^2$$

we see

$$\mathrm{Pa} = \left(\frac{\mathrm{J}}{\mathrm{mol}\cdot\mathrm{K}}\right)(\mathrm{K})(B_2)\left(\frac{\mathrm{kg/m^3}}{\mathrm{kg/mol}}\right)^2 = \left(\frac{\mathrm{J}}{\mathrm{mol}}\right)(B_2)\left(\frac{\mathrm{mol}}{\mathrm{m^3}}\right)^2$$

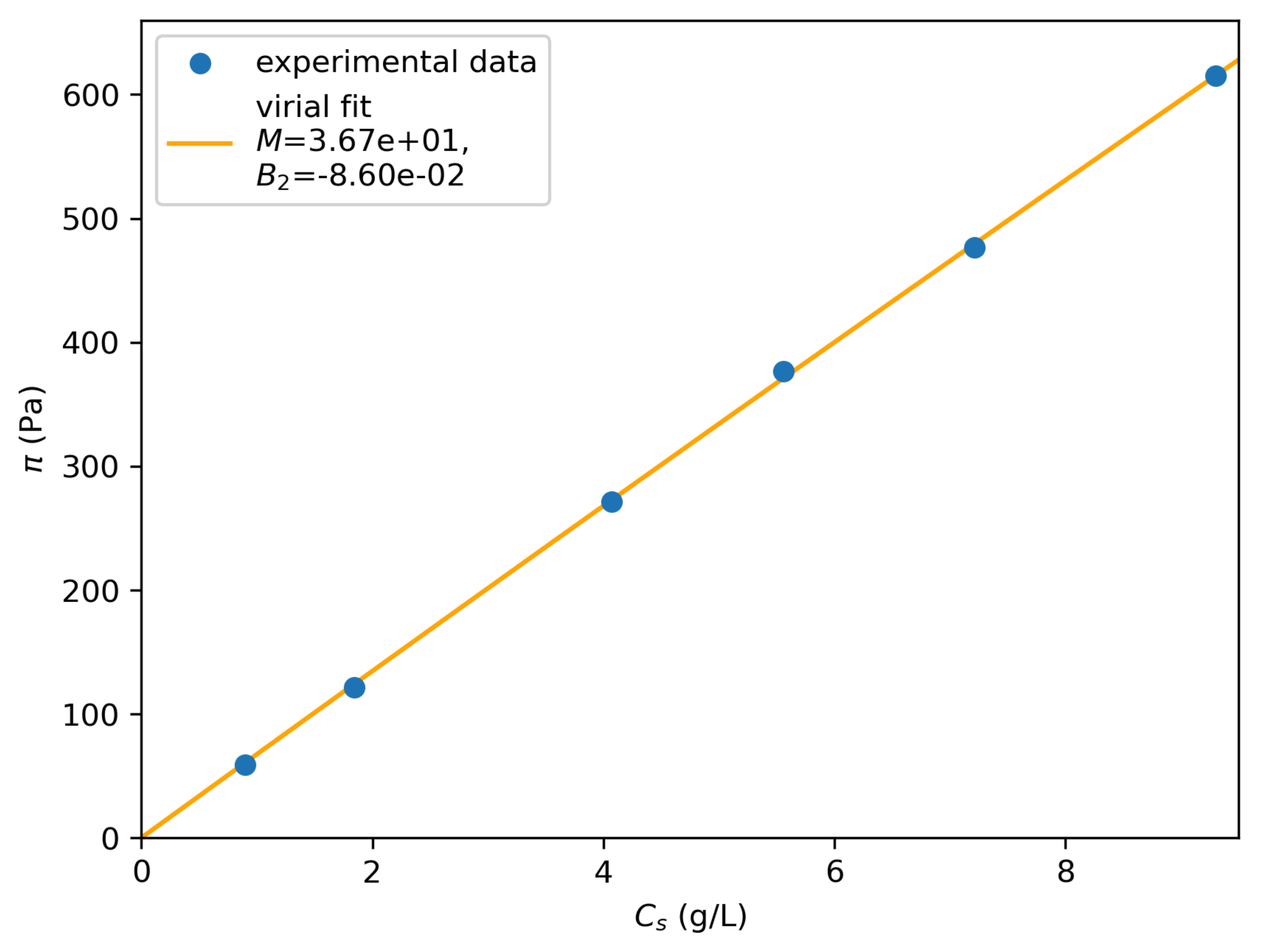
$$\frac{\mathrm{J}}{\mathrm{m}^3} = \mathrm{J}\cdot\mathrm{mol}\cdot B_2\cdot\mathrm{m}^{-6} \qquad \longrightarrow \qquad B_2 = \frac{\mathrm{m}^3}{\mathrm{mol}}$$

```
def pi(Cs, M, B2):
    return R * T * (Cs/M) * (1 + B2 * (Cs/M))

(m,b2),_ = curve_fit(pi, df['Cs'], df['pi'])
print(f'molecular weight: {1000*m:.1f} g/mol')
print(f'B2: {b2:.4f} m^3/mol')
```

```
molecular weight: 36655.0 g/mol
B2: -0.0860 m^3/mol
```

and visualize our fit

```
fig, ax = plt.subplots(dpi=300)
ax.scatter(x, y)
ax.plot(np.linspace(0,9.5,1000), pi(np.linspace(0,9.5,1000), M=m, B2=b2), c='orange', zor
ax.legend(['experimental data', f'virial fit\n$M$={m:.2e},\n$B_2$={b2:.2e}'])
ax.set(xlim=(0,9.5),ylim=0, xlabel='$C_s$ (g/L)', ylabel='$\pi$ (Pa)');
```



looks good to me. we can also try just a simple polynomial fit approach by rearranging our equation a little bit

$$\frac{\pi}{RT} = \left(\frac{1}{M}\right)C_s + \left(\frac{B_2}{M^2}\right)C_s^2$$

and then we can use fitted parameters to find M and B_2 from

$$\pi = X_0 + X_1C_s + X_2C_s^2$$

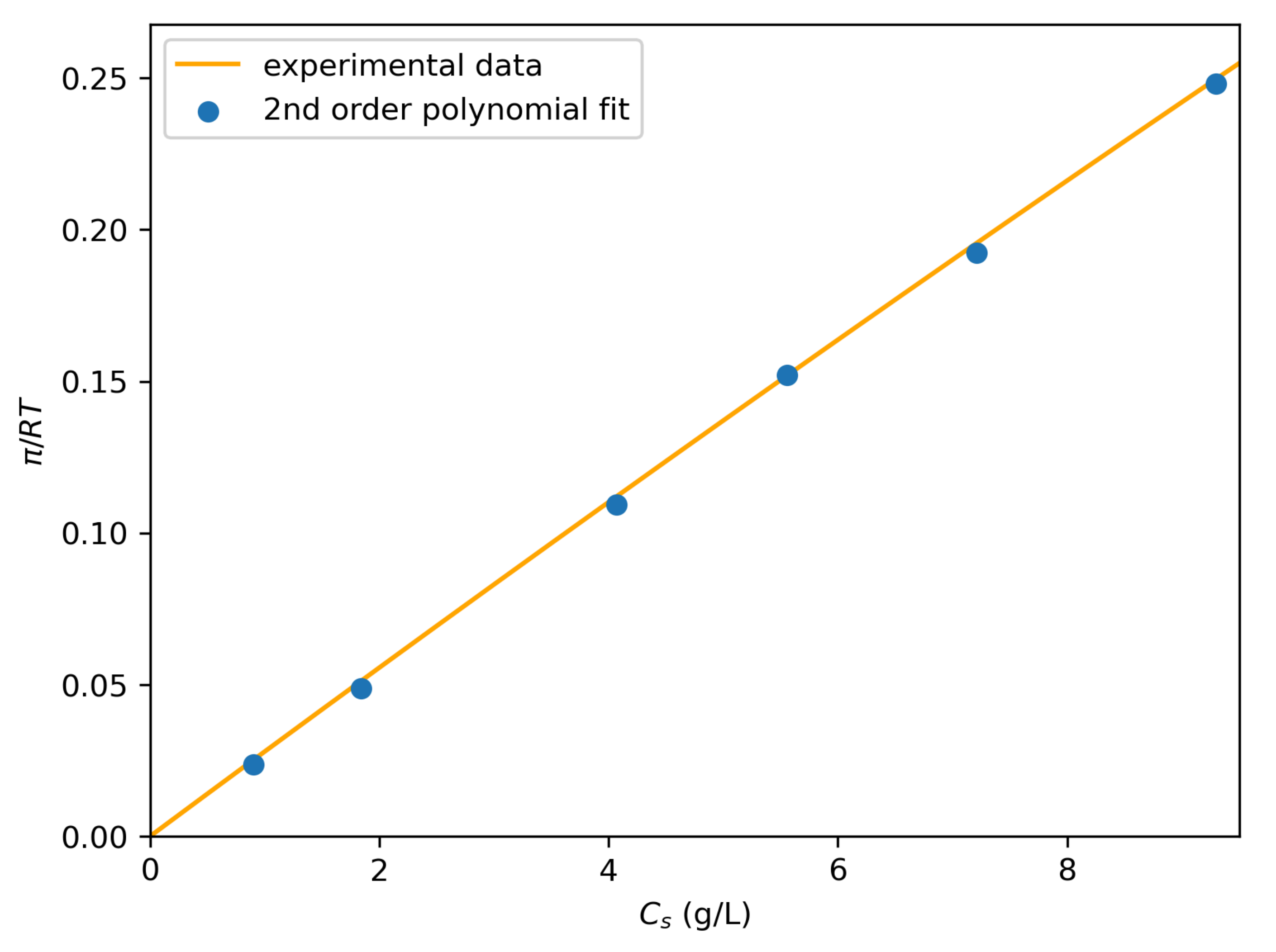
$$M = \frac{1}{X_1} \qquad B_2 = M^2X_2$$

```
(x2, x1, x0) = np.polyfit(x, y/R/T, 2)
print(f'fit parameters: {np.round((x0, x1, x2),4)}')

M = 1/x1
print(f'M: {1e3*M:.1f}')

B2 = M**2 * x2
print(f'B2: {B2:.3f}')
```

```
fit parameters: [-0.0019  0.0281 -0.0001]
M: 35641.2
B2: -0.165
```

now visualizing this fit

```
fig, ax = plt.subplots(dpi=300)
plt.plot(np.linspace(0,9.5,1000), x1*np.linspace(0,9.5,1000) + x2*np.linspace(0,9.5,1000)
plt.scatter(x, y/R/T)
ax.legend(['experimental data', '2nd order polynomial fit'])
ax.set(xlim=(0,9.5), ylim=0, xlabel='$C_s$ (g/L)', ylabel='$\pi/RT$');
```



another thing we could try (even though whoever grades this will probably never even read this line) is dividing the equation by C_s and doing just a linear fit!

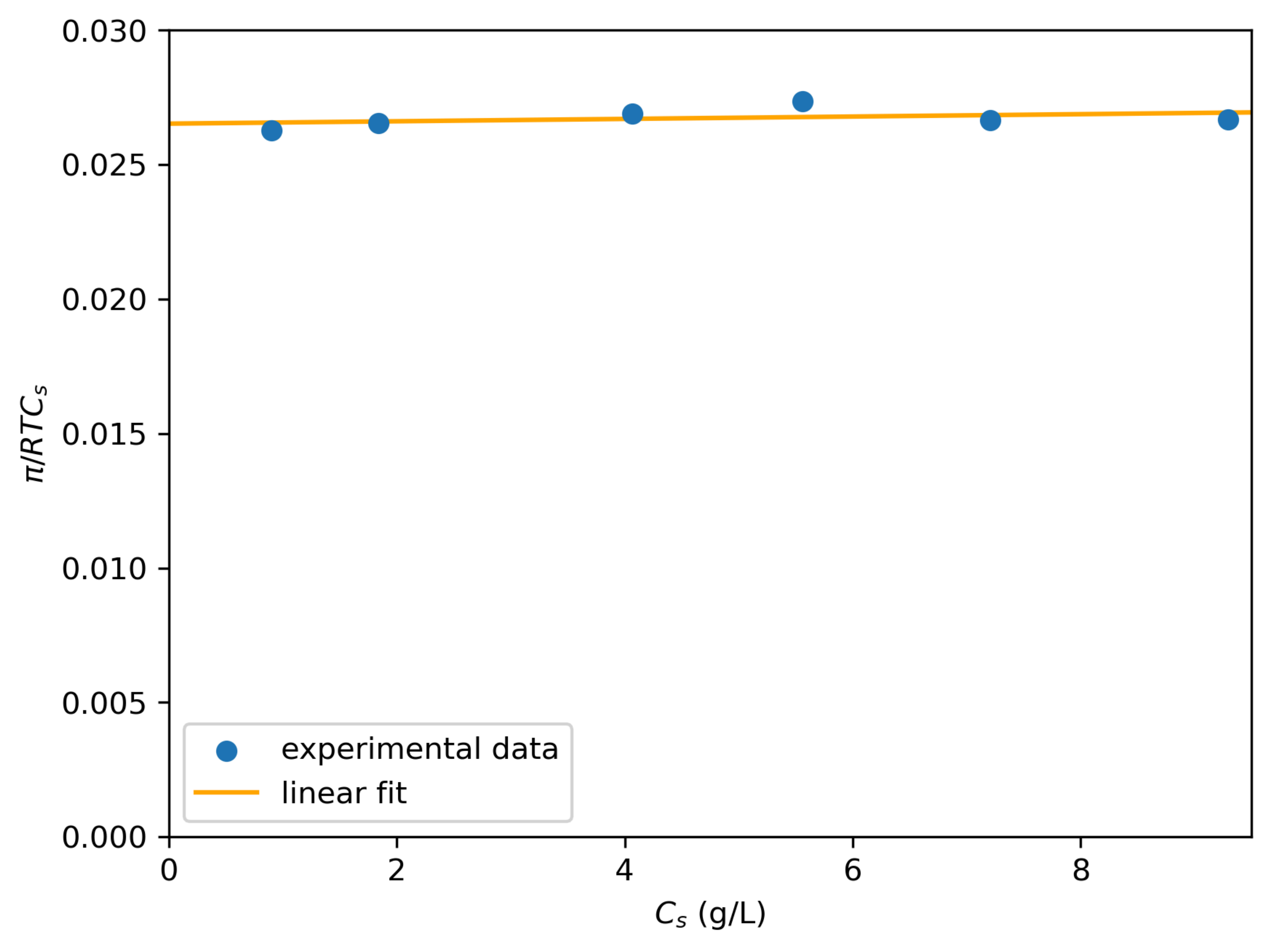
$$\frac{\pi}{RTC_s} = \left(\frac{1}{M}\right) + \left(\frac{B_2}{M^2}\right)C_s$$

```
m, b = np.polyfit(x, y/R/T/x, 1)

M = 1/b
print(f'M: {1e3*M:.1f} g/mol')

B2 = M**2 * m
print(f'B2: {B2:.3f}')
```

```
M: 37697.8 g/mol
B2: 0.063
```

it's interesting that this method produces a positive B_2. let's look at what we're actually trying to fit a line to

```
fig, ax = plt.subplots(dpi=300)
plt.scatter(x, y/R/T/x)
ax.plot(np.linspace(0,9.5,1000), m*np.linspace(0,9.5,1000) + b, c='orange', zorder=-10)
ax.legend(['experimental data', 'linear fit'])
ax.set(xlim=(0,9.5), ylim=(0,0.03), xlabel='$C_s$ (g/L)', ylabel='$\pi/RTC_s$');
```



the different approaches giving different results is an artifact of the data being real experimental data and not data that perfectly fits this model.

```
# filler
```