# Preamble

[ ] ↳ 2 cells hidden

# Data

[ ] ↳ 8 cells hidden

# Michaelis-Menten Kinetics

pH 9.0

[ ] ↳ 16 cells hidden

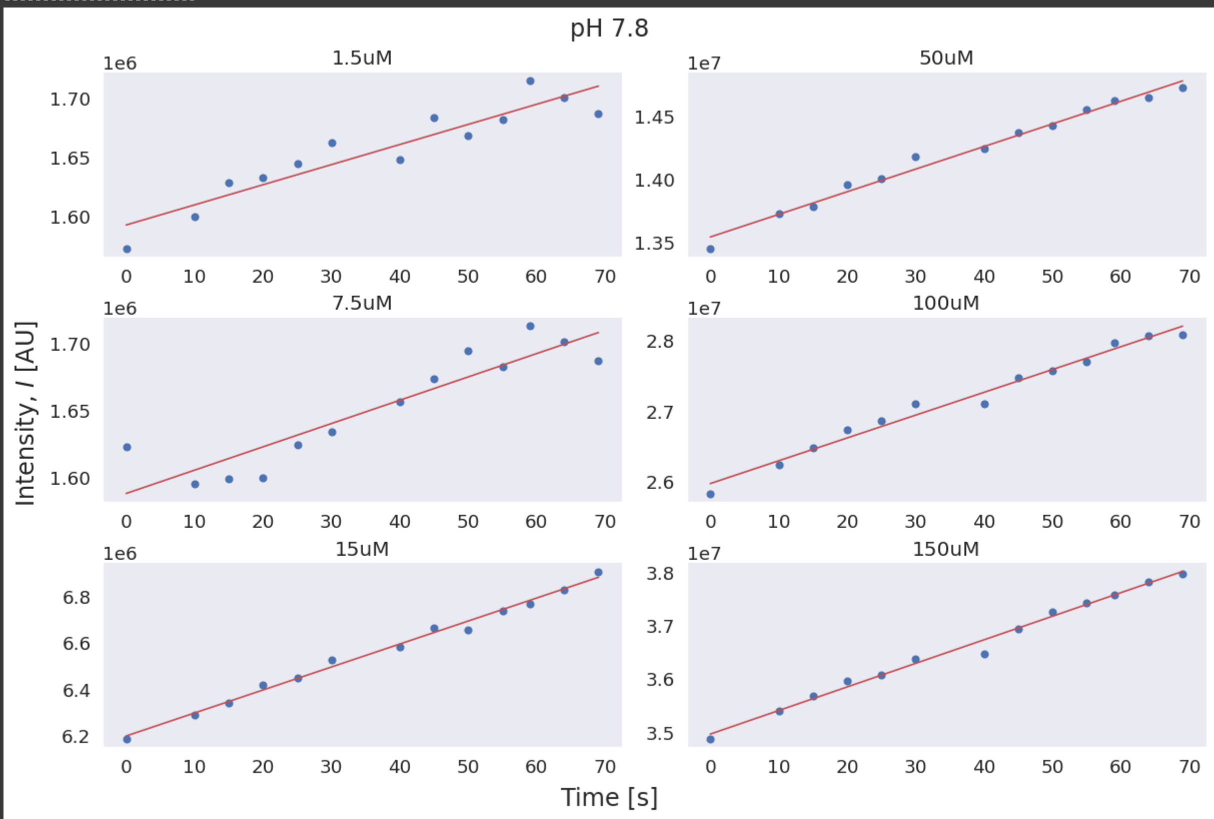
# Michaelis-Menten Kinetics

pH 7.8

```
## Data Import and Initial Processing
# This section cleans up the .txt files supplied by Nikos.

# path where files are stored
dir_path = "/content/Practicals/Spectrometer/Cteam_cleaned"
# defining empty df
#MM_data = pd.DataFrame(columns=['Data Series', 'Concentration', 'Time', 'Intensity'])

# Array of all .txt files located at 'dir_path' - including subdirectories
paths_list = Path(dir_path).glob('**/*.txt')


## Make loop for importing each CSV file.
## Reuse from below
## One DF or four? Or two?
## pd.read_csv('data.csv')

MM_data_ph78 = pd.read_csv('/content/Practicals/Spectrometer/Cteam_cleaned/ph78_cleaned.csv')
MM_data_ph78c = pd.read_csv('/content/Practicals/Spectrometer/Cteam_cleaned/calibration_ph78_cleaned.csv')

MM_data_ph90 = MM_data_ph78
MM_data_ph90c = MM_data_ph78c

## subplot init


fig, axs = plt.subplots(3, 2, figsize = (12, 8), sharex=False, sharey=False, layout='constrained')
## MM_data_ph90
t = MM_data_ph90.iloc[:, 0:1]
# ['1.5uM', '7.5uM', '15uM', '50uM', '100uM', '150uM']


print("############################")
print('pH 7.8:')
print("------------------------")
res = stats.linregress(t['Time'], MM_data_ph90['1.5uM'])
axs[0,0].plot(t, MM_data_ph90['1.5uM'], 'o', label='original data')
axs[0,0].plot(t, res.intercept + res.slope*t, 'r', label='fitted line')
print('Conc1:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")

res = stats.linregress(t['Time'], MM_data_ph90['7.5uM'])
axs[1,0].plot(t, MM_data_ph90['7.5uM'], 'o', label='original data')
axs[1,0].plot(t, res.intercept + res.slope*t, 'r', label='fitted line')
print('Conc2:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")

res = stats.linregress(t['Time'], MM_data_ph90['15uM'])
axs[2,0].plot(t, MM_data_ph90['15uM'], 'o', label='original data')
axs[2,0].plot(t, res.intercept + res.slope*t, 'r', label='fitted line')
print('Conc3:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")

res = stats.linregress(t['Time'], MM_data_ph90['50uM'])
axs[0,1].plot(t, MM_data_ph90['50uM'], 'o', label='original data')
axs[0,1].plot(t, res.intercept + res.slope*t, 'r', label='fitted line')
print('Conc4:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")

res = stats.linregress(t['Time'], MM_data_ph90['100uM'])
axs[1,1].plot(t, MM_data_ph90['100uM'], 'o', label='original data')
axs[1,1].plot(t, res.intercept + res.slope*t, 'r', label='fitted line')
print('Conc5:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")

res = stats.linregress(t['Time'], MM_data_ph90['150uM'])
axs[2,1].plot(t, MM_data_ph90['150uM'], 'o', label='original data')
axs[2,1].plot(t, res.intercept + res.slope*t, 'r', label='fitted line')
print('Conc6:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")

print("------------------------")

## MM_data_ph90c
#cols_order = ['Time', 'Temperature', 'conc5', 'conc4', 'conc3', 'conc2', 'conc1']
#MM_data_ph90c = MM_data_ph90c[cols_order]
```

```python
#conc_values = np.array([1.0e-9, 10.0e-9, 50.0e-9, 200.0e-9, 1000.0e-9], dtype="float64")
#conc_values = np.array([0.001, 0.01, 0.05, 0.2, 1.0], dtype="float64")
#conc_values = np.array([1.0, 10.0, 50.0, 200.0, 1000.0], dtype="float64")
conc_values = np.array([1.5e-6, 7.5e-6, 15.0e-6, 50.0e-6, 100.0e-6, 150.0e-6], dtype="float64")
conc_avg = MM_data_ph90c.iloc[:, 2:8].mean(axis=0)
conc_SEM = MM_data_ph90c.iloc[:, 2:8].sem(axis=0)

#res = stats.linregress(conc_values, conc_avg)
#axs[2,1].plot(conc_values, conc_avg, 'o', label='Mean Intensity')
#axs[2,1].plot(conc_values, res.intercept + res.slope*conc_values, 'r', label='fitted line')

axs[0,0].set_title("1.5uM")
axs[1,0].set_title("7.5uM")
axs[2,0].set_title("15uM")
axs[0,1].set_title("50uM")
axs[1,1].set_title("100uM")
axs[2,1].set_title("150uM")
fig.supxlabel(r'Time [s]');
fig.supylabel(r'Intensity, $I$ [AU]');
fig.suptitle(r'pH 7.8')
plt.ticklabel_format(style='scientific', axis='y', scilimits=(0,0), useMathText=False)
plt.show()
```

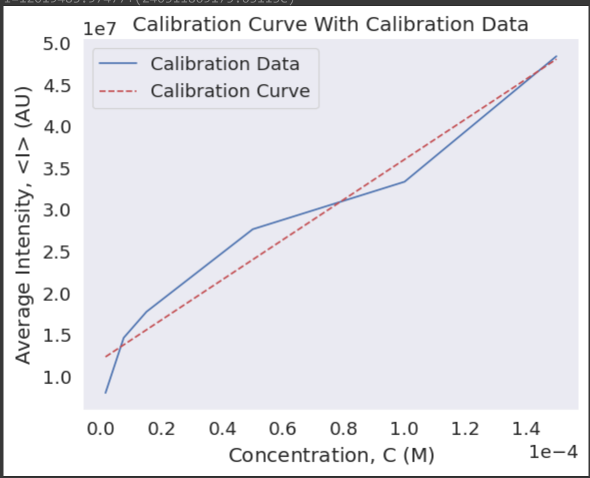
```
############################
pH 7.8:
------------------------
Conc1:
R-squared: 0.871378
I=1591974.45656+(1712.99806t)
Conc2:
R-squared: 0.833577
I=1588090.59814+(1740.18926t)
Conc3:
R-squared: 0.991561
I=6196612.48958+(9926.04904t)
Conc4:
R-squared: 0.985025
I=13542517.57506+(17853.49279t)
Conc5:
R-squared: 0.980451
I=25967347.64615+(32285.99295t)
Conc6:
R-squared: 0.990869
I=34958669.51756+(44209.69351t)
------------------------
```




```python
print("############################")

print('Calibration:')
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "t)")
print("Standard Error of the Mean:")
  print(conc_SEM)

print("------------------------")
print('Calibration Model:')

fig,ax = plt.subplots()
xdata = conc_values #conc_flipped  #conc_values #conc_flipped
ydata = conc_avg #np.flipud(conc_avg)   #conc_avg
plt.plot(xdata, ydata, 'b-', label='Calibration Data')

res = stats.linregress(xdata, ydata)
print(res)
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 5)) + "+(" + str(round(res.slope, 5)) + "C)")

plt.plot(xdata, res.intercept + res.slope*xdata, 'r--', label="Calibration Curve")  #label="I=" + str(round(res.intercept, 5)) + "\n +(" + str(round(res.slop
```

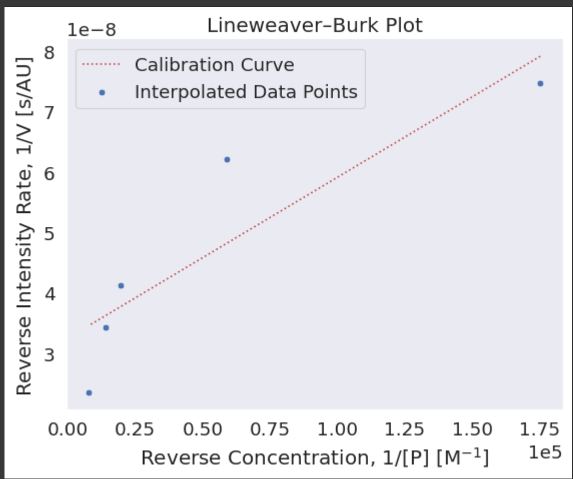
```python
from matplotlib import ticker
#ax.xaxis.set_major_formatter(ticker.StrMethodFormatter("{x:.2f}"))
plt.ticklabel_format(style='scientific', axis='y', scilimits=(0,0), useMathText=False)
plt.ticklabel_format(style='scientific', axis='x', scilimits=(0,0), useMathText=False)
#plt.xlabel('Conc.')
#plt.ylabel('I')
ax.set(xlabel='Concentration, C $(\mathrm{M}{})$', ylabel='Average Intensity, <I> $(\mathrm{AU}{})$', title="Calibration Curve With Calibration Data")
plt.legend()
plt.show()
```

```
############################
Calibration:
R-squared: 0.990869
I=34958669.51756+(44209.69351t)
Standard Error of the Mean:
1.5uM     55476.815524
7.5uM    215986.905443
15uM     286664.151274
50uM     573189.971453
100uM    708344.270272
150uM    973012.174114
dtype: float64
-------------------------
Calibration Model:
LinregressResult(slope=240311869175.63113, intercept=12019483.974772332, rvalue=0.9789201735787293, pvalue=0.0006618551167682339, stderr=25069495851.794266, intercept_stderr=1922
R-squared: 0.958285
I=12019483.97477+(240311869175.63113C)
```



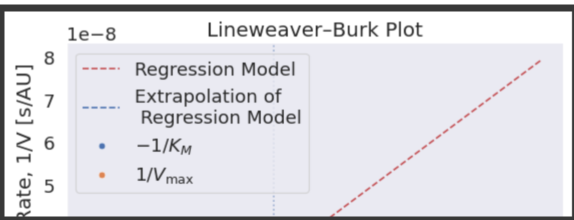

```python
## plot from MM data
# "The reaction changes from approximately first-order in substrate concentration at low concentrations
# to approximately zeroth order at high concentrations."
# (https://en.wikipedia.org/wiki/Michaelis%E2%80%93Menten_kinetics)
# Vi har kun data til at dække den lineære del af kurven


#t = np.array([0.0, 63.0], dtype='float32')
t = 1

# slopes (I/t = v)
#intensity_rates = np.array([(-10119.75805),(-39631.2509),(-52719.25862),(-105582.07944),(-130468.45906),(-179262.46539)])
intensity_rates = np.array([(7715616.25247+(10119.75805*t)), (13350739.91261+(39631.2509*t)), (16040484.4259+(52719.25862*t)), (24179380.90472+(105582.07944*t)), (29046879.39586+(13
# average I
#mean_intensity_rates = np.array([(7715616.25247+(10119.75805*t)).mean(), (13350739.91261+(39631.2509*t)).mean(), (16040484.4259+(52719.25862*t)).mean(), (24179380.90472+(105582.079

# from slopes
concentration = ( intensity_rates - 12019483.97477 ) / (240311869175.63113)
# from average I
#concentration = ( mean_intensity_rates - 12019483.97477 ) / (240311869175.63113)

y = intensity_rates
x = concentration

plt.ticklabel_format(style='scientific', axis='x', scilimits=(0,0), useMathText=False)
plt.ticklabel_format(style='scientific', axis='y', scilimits=(0,0), useMathText=False)

#ax = sns.scatterplot(x=concentration_2, y=intensity_rates, label='INVERSE CF')
plt.plot((xdata), res.intercept + res.slope*(xdata), 'r--', label="Calibration Curve")
ax = sns.scatterplot(x=x, y=y, label='Interpolated Data Points')
ax.set(xlabel='Concentration, C $(\mathrm{M}{})$', ylabel='Intensity Rate, V $(\mathrm{AU/s}{})$', title="Calibration Curve With Data Points")
plt.legend()
plt.show()
```


```python
## inverse MM plot

y = intensity_rates[1::] #intensity_rates
x = concentration[1::] #concentration
y = 1/y
x = 1/x

res = stats.linregress(x, y, alternative='greater')
plt.plot(x, res.intercept + res.slope*x, 'r:', label="Calibration Curve")  #label="I=" + str(round(res.intercept, 5)) + "\n +(" + str(round(res.slope, 5)) + "C)"

plt.ticklabel_format(style='scientific', axis='x', scilimits=(0,0), useMathText=False)
ax = sns.scatterplot(x=x, y=y, label='Interpolated Data Points')


ax.set(xlabel='Reverse Concentration, 1/[P] $[\mathrm{M}{^{-1}}]$', ylabel='Reverse Intensity Rate, 1/V $[\mathrm{s/AU}{}]$', title="Lineweaver-Burk Plot")
plt.legend()
plt.show()

print("-----------------------------------")
print("Lineweaver-Burk Regression Model:")
print(res)
print(f"R-squared: {res.rvalue**2:.6f}")
print("I=" + str(round(res.intercept, 20)) + "+(" + str(round(res.slope, 20)) + "C)")
print("-----------------------------------")
```




```
-----------------------------------
Lineweaver-Burk Regression Model:
LinregressResult(slope=2.661715728976727e-13, intercept=3.246412361961343e-08, rvalue=0.8927988850715994, pvalue=0.020724932871894974, stderr=7.753526779698125e-14, intercept_std
R-squared: 0.797090
I=3.246412361961e-08+(2.6617157e-13C)
-----------------------------------
```

```python
# Hvis ovenstående er et Lineweaver-Burke plot, så gælder følgende:
# (fra linjens ligning)

# 1/V_max = 1 / (3.246412361961e-08)

# K_M =
#3.246412361961e-08 + (2.6617157e-13 * 0)

C_list = np.arange(-121966.909, 7843.55789755)
I_list = 3.246412361961e-08+(2.6617157e-13*C_list)
#data_points = np.array([ [, 0.1]])



plt.plot(x, res.intercept + res.slope*x, 'r--', label="Regression Model")
plt.plot(C_list, I_list, 'b--', label="Extrapolation of\n Regression Model");

x_zero = [-121966.9088611154]
y_zero = [0]
ax = sns.scatterplot(x=x_zero, y=y_zero, label='$-1/K_{M}$');

x_zero = [0]
y_zero = [3.246412361961e-08]
ax = sns.scatterplot(x=x_zero, y=y_zero, label='$1/V_{\mathrm{max}{}}$');

plt.axhline(y=0, alpha=0.5, linestyle=':')
plt.axvline(x=0, alpha=0.5, linestyle=':')

ax.set(xlabel='Inverse Concentration, 1/[P] $[\mathrm{M}{^{-1}}]$', ylabel='Inverse Intensity Rate, 1/V $[\mathrm{s/AU}{}]$', title="Lineweaver-Burk Plot")
#plt.xticks(rotation=30)
plt.ticklabel_format(style='scientific', scilimits=[-2, 2], axis='both', useMathText=False)
plt.legend()
plt.show()
```




## Diffusion Coefficient

[ ] ↳ 6 cells hidden

## Code Snippets From Workshop

[ ] ↳ 6 cells hidden

## Out of Scope

[ ] ↳ 21 cells hidden