Effects of O_2 and SO_2 on the capture capacity of a primary-amine based polymeric CO_2 sorbent

Alexander P. Hallenbeck[†] and John R. Kitchin[*,†,‡]

National Energy Technology Laboratory-Regional University Alliance (NETL-RUA), Pittsburgh, Pennsylvania 15236, and Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes, Ave, Pittsburgh, PA 15213

E-mail: jkitchin@cmu.edu

About this document

This supporting information document was prepared in org-mode and exported to pdf. The document in its native form is in plain text format, marked up using org-mode syntax.[1] Org-mode is a lightweight text markup language that enables intermingling of narrative text, data and analysis code in an active document[2] when viewed in the editor Emacs (http://www.gnu.org/software/emacs/). This approach is known as literate programming and reproducible research.[3] The advantage of this approach is that it enables a complete record of what work was done, data files can be directly embedded in the final pdf, and the analysis used to generate the figures can be easily included. All of the data used in the manuscript is contained in this file.

The native org-file is available here (double-click on pushpin to open the file):

[*]To whom correspondence should be addressed
[†]National Energy Technology Laboratory-Regional University Alliance (NETL-RUA)
[‡]Carnegie Mellon University

Total volumetric flowrate during a typical experiment

This spreadsheet contains representative data for the total flowrate during the adsorption and desorption phase of the experiment. Graphs showing the data are embedded in the spreadsheet.

Datafile (double-click on pushpin to open the file):

BET data for the experiments reported in the manuscript.

1. BET data for the resin after drying.

Datafile (double-click on pushpin to open the file):

1. BET data for the resin after poisoning by SO_2.

Datafile (double-click on pushpin to open the file):

1. BET data for the resin after exposure to air at 120 °C) for seven days.

Datafile (double-click on pushpin to open the file):

Showing no thermal desorption of SO_2

This data shows that no SO_2 desorbs from the poisoned resin at temperatures up to about 200 °C).

Datafile:

```
import xlrd
import string
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.ticker import MaxNLocator
from matplotlib import rcParams
rcParams['font.size'] = 12
rcParams['figure.figsize'] = (3,4);
rcParams['figure.subplot.bottom'] = 0.12;
```

```
rcParams['figure.subplot.top'] = 0.91;
rcParams['figure.subplot.right'] = 0.76;
rcParams['figure.subplot.left'] = 0.30;
rcParams['axes.labelsize'] = 12;

wb = xlrd.open_workbook('data/so2-thermal-des.xls')
sh = wb.sheet_by_name('Sheet1')

# time is in column A
# CO$_2$ is in column AB
# Temperature is in column H
# this block makes constants for the column names
for i in range(2*26):
    if i / 26 == 0:
        exec('{0} = {1}'.format(string.ascii_uppercase[i % 26], i))
    else:
        exec('A{0} = {1}'.format(string.ascii_uppercase[i % 26], i))

time = np.array(sh.col_values(A, start_rowx=1))
SO2 = np.array(sh.col_values(R, start_rowx=1))
T = np.array(sh.col_values(H, start_rowx=1)) + 273.15

# I want to plot from t=7920 to 10300
ind = (time > 7920)

time = time[ind] - 7920
SO2 = SO2[ind]
T = T[ind]

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.semilogy(time, SO2, 'b', label='SO$_2$')
#ax1.set_xlim([7920, 10300])
ax1.set_ylim([1e-13, 1])
ax1.set_xlabel('Time (sec)')
ax1.set_ylabel('SO$_2$ M.S. intensity (arb. units)', color='b')
for tl in ax1.get_yticklabels():
    tl.set_color('b')

ax2 = ax1.twinx()
ax2.plot(time, T, 'r', label='Temperature (K)')
```

```
ax2.set_ylabel('Temperature (K)', color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')

ax2.xaxis.set_major_locator(MaxNLocator(3))

for ext in ('.png','.eps','.pdf'):
    plt.savefig('figures/so2-thermal-des' + ext)

plt.show()
```

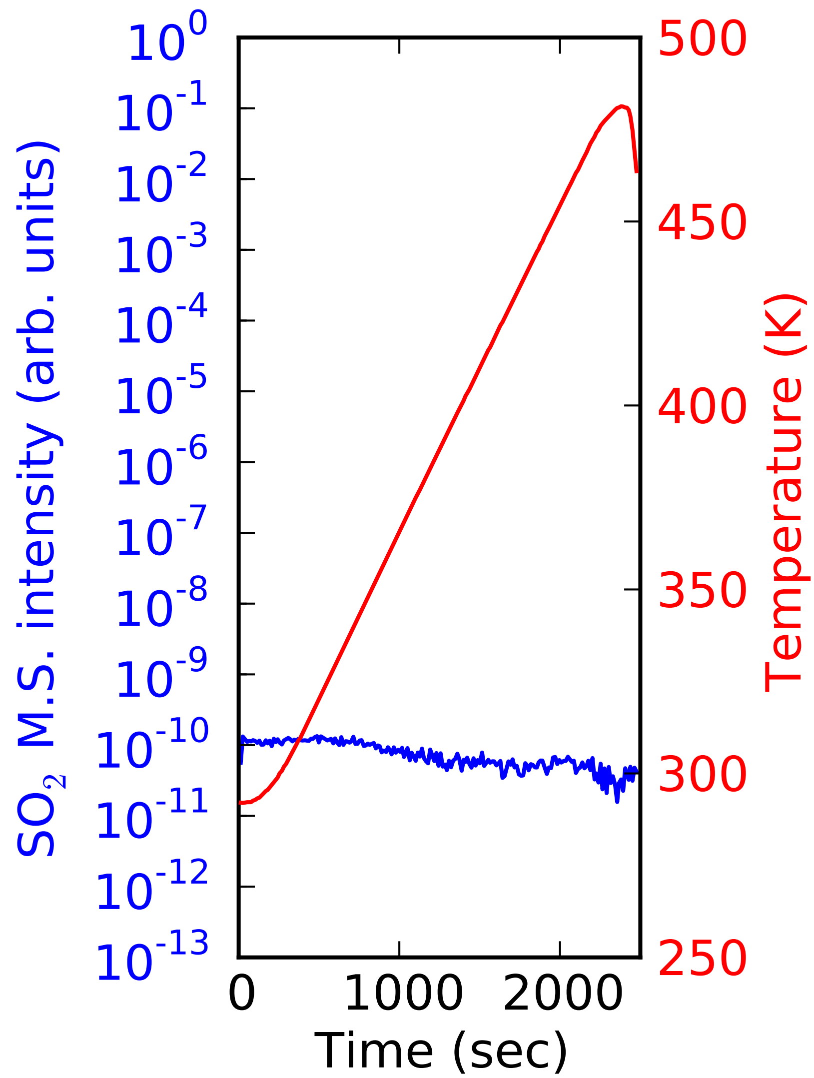

Figure 1: Temperature swing regeneration of SO_2-poisoned resin under inert N_2 gas stream.

Figure 1

Datafile:

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

from matplotlib import rcParams
rcParams['font.size'] = 12
rcParams['figure.figsize'] = (6,4);
rcParams['figure.subplot.bottom'] = 0.12;
rcParams['figure.subplot.top'] = 0.91;
rcParams['figure.subplot.right'] = 0.9;
rcParams['figure.subplot.left'] = 0.10;
rcParams['axes.labelsize'] = 12;

wb = xlrd.open_workbook('data/typ-ads-des.xls')
sh = wb.sheet_by_name('Sheet1')

# time is in column A
# CO$_2$ is in column AB
# Temperature is in column H
# this block makes constants for the column names
for i in range(2*26):
    if i / 26 == 0:
        exec('{0} = {1}'.format(string.ascii_uppercase[i % 26], i))
    else:
        exec('A{0} = {1}'.format(string.ascii_uppercase[i % 26], i))

T = np.array(sh.col_values(H, start_rowx=1)) + 273.15
CO2 = sh.col_values(AB, start_rowx=1)
time = sh.col_values(A, start_rowx=1)

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(time, CO2)
ax1.set_xlabel('Time (sec)')
ax1.set_ylabel('vol% CO$_2$', color='b')
for tl in ax1.get_yticklabels():
```

```
    tl.set_color('b')
ax1.set_xlim([0, 6000])
ax1.set_ylim([0,12])
ax1.text(75, 11, 'Adsorption', va='top')
ax1.text(3050, 5, 'Pressure\nswing\ndesorption')
ax1.text(4300, 4, 'Thermal\nDesorption', va='top')

ax2 = ax1.twinx()
ax2.plot(time, T, 'r')
ax2.set_ylabel('Temperature (K)',color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')
ax2.set_ylim([273, 140 + 273])

# the arrow units are the axes
ax2.arrow(4937, 360, 500, 0, color='r', head_width=5, head_length=150)
ax2.set_xlim([0, 6000])

for ext in ('.png','.eps','.pdf'):
    plt.savefig('figures/fig1' + ext)
plt.show()
```

Figure 2

Cycle #	Adsorption capacity (mol/kg)	Desorption capacity (mol/kg)
1	1.56	1.46
2	1.18	1.35
3	1.39	1.37
4	1.11	1.34
5	1.21	1.3
6	1.22	1.26
10	1.47	1.28
11	1.27	1.25
12	1.1	1.27
13	1.16	1.28
14	1.26	1.2
15	1.28	1.29
16	1	1.26
17	1.1	1.43

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
rcParams['font.size'] = 12
rcParams['figure.figsize'] = (3,4);
rcParams['figure.subplot.bottom'] = 0.12;
rcParams['figure.subplot.top'] = 0.91;
rcParams['figure.subplot.right'] = 0.9;
rcParams['figure.subplot.left'] = 0.25;
rcParams['axes.labelsize'] = 11;

data= np.array(data)

cycles = data[:,0]
ads_capacity = data[:,1]
des_capacity = data[:,2]
```

```

plt.plot(cycles, ads_capacity, 'bd')
plt.plot(cycles, des_capacity, 'ro')
plt.ylim([0, 1.8])

plt.xlabel('Cycle #')
plt.ylabel('CO$_2$ Capture Capacity (mol/kg)')

plt.locator_params(axis = 'x', nbins = 4)

plt.legend(['Adsorption', 'Desorption'], loc='best')
for ext in ('.png','.eps','.pdf'):
    plt.savefig('figures/fig2' + ext)
plt.show()
```

Figure 3

Cycle #	Adsorption Capacity (mol/kg)	Desorption Capacity (mol/kg)
1	1.48	1.43
2	1.31	1.41
3	1.04	1.04
4	0.82	0.73
5	0.53	0.47
6	0.34	0.33
7	0.19	0.2
8	0.17	0.14
9	0.17	0.15

Cycle #	Adsorption Capacity (mol/kg)	Desorption Capacity (mol/kg)
1	1.51	1.51
2	1.26	1.35
3	0.97	0.91
4	0.79	0.68
5	0.47	0.45
6	0.32	0.29
7	0.11	0.12
8	0.16	0.15
9	0.16	0.17

Cycle #	Adsorption Capacity (mol/kg)	Desorption Capacity (mol/kg)
1	1.2	1.54
2	1.13	1.13
3	0.96	0.94
4	0.91	0.94
5	0.72	0.72
6	0.56	0.69
7	0.42	0.45
8	0.27	0.26
9	0.18	0.32
10	0.2	0.3
11	0.15	0.17

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import rcParams
rcParams['font.size'] = 12
rcParams['figure.figsize'] = (3,4);
rcParams['figure.subplot.bottom'] = 0.12;
```

```
rcParams['figure.subplot.top'] = 0.91;
rcParams['figure.subplot.right'] = 0.9;
rcParams['figure.subplot.left'] = 0.25;
rcParams['axes.labelsize'] = 12;
rcParams['legend.fontsize'] = 10
data1= np.array(data1)
data2 = np.array(data2)
data3 = np.array(data3)


plt.plot(data1[:,0], data1[:,1], 'bd', label='Adsorption 1')
plt.plot(data1[:,0], data1[:,2], 'ro', label='Desorption 1')

plt.plot(data2[:,0], data2[:,1], 'gd', label='Adsorption 2')
plt.plot(data2[:,0], data2[:,2], 'ko', label='Desorption 2')

plt.plot(data3[:,0], data3[:,1], 'md', label='Adsorption 3')
plt.plot(data3[:,0], data3[:,2], 'yo', label='Desorption 3')
plt.xlim([0, 15])
plt.xlabel('Cycle #')
plt.ylabel('CO$_2$ Capture Capacity (mol/kg)')
plt.ylim([0, 1.8])

plt.legend(loc='best', borderpad=0.5, handletextpad=0, fontsize='small', numpoints=1)
for ext in ('.png','.eps','.pdf'):
    plt.savefig('figures/fig3' + ext)
plt.show()
```

Figure 4

Datafile:

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

from matplotlib import rcParams
rcParams['font.size'] = 12
```

```python
rcParams['figure.figsize'] = (6,4);
rcParams['figure.subplot.bottom'] = 0.12;
rcParams['figure.subplot.top'] = 0.91;
rcParams['figure.subplot.right'] = 0.87;
rcParams['figure.subplot.left'] = 0.10;
rcParams['axes.labelsize'] = 12;

wb = xlrd.open_workbook('data/co2-so2-ms.xls')
sh = wb.sheet_by_name('Sheet1')

# time is in column A
# CO$_2$ is in column AB
# Temperature is in column H
# this block makes constants for the column names
for i in range(2*26):
    if i / 26 == 0:
        exec('{0} = {1}'.format(string.ascii_uppercase[i % 26], i))
    else:
        exec('A{0} = {1}'.format(string.ascii_uppercase[i % 26], i))

time = np.array(sh.col_values(A, start_rowx=1))/3600.
CO2 = sh.col_values(Y, start_rowx=1)
SO2 = sh.col_values(R, start_rowx=1)

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(time, CO2, label='CO$_2$')
ax1.set_xlabel('Time (h)')
ax1.set_ylabel('vol% CO$_2$', color='b')
for tl in ax1.get_yticklabels():
    tl.set_color('b')

ax2 = ax1.twinx()
ax2.plot(time, SO2, 'r', label='SO$_2$')
ax2.set_ylabel('SO$_2$ M.S. intensity (arb. units)', color='r')
for tl in ax2.get_yticklabels():
    tl.set_color('r')
ax2.set_xlim([0, 20.7])
for ext in ('.png','.eps','.pdf'):
    plt.savefig('figures/fig4' + ext)

```

```
plt.show()
```

Figure 5

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

from matplotlib import rcParams
rcParams['font.size'] = 12
rcParams['figure.figsize'] = (6,4);
rcParams['figure.subplot.bottom'] = 0.12;
rcParams['figure.subplot.top'] = 0.91;
rcParams['figure.subplot.right'] = 0.86;
rcParams['figure.subplot.left'] = 0.10;
rcParams['axes.labelsize'] = 12;

wb = xlrd.open_workbook('data/p-t-desorption.xls')
sh = wb.sheet_by_name('regular')

# this block makes constants for the column names
for i in range(2*26):
    if i / 26 == 0:
        exec('{0} = {1}'.format(string.ascii_uppercase[i % 26], i))
    else:
        exec('A{0} = {1}'.format(string.ascii_uppercase[i % 26], i))

time = np.array(sh.col_values(A, start_rowx=1))
CO2 = np.array(sh.col_values(B, start_rowx=1))

ind1 = time > 3000

time = time[ind1] - 3000
CO2 = CO2[ind1]

fig = plt.figure()
```

```python
ax1 = fig.add_subplot(111)
h1, = ax1.plot(time, CO2, 'r', label='as received')
ax1.set_xlabel('Time (sec)')
ax1.set_ylabel('vol% CO$_2$')

ax1.set_ylim([0, 12])

sh2 = wb.sheet_by_name('after 7 days')
time2 = np.array(sh2.col_values(A, start_rowx=1))
CO22 = np.array(sh2.col_values(AA, start_rowx=1))
T2 = np.array(sh2.col_values(H, start_rowx=1)) + 273.15

ind2 = time2 > 2267
time2 = time2[ind2] - 2267
CO22 = CO22[ind2]
T2 = T2[ind2]

h2, = ax1.plot(time2, CO22, 'b', label='Heated in air for 7 days')

ax2 = ax1.twinx()
h3, = ax2.plot(time2, T2, 'g')
ax2.set_ylabel('Temperature (K)',color='g')
for tl in ax2.get_yticklabels():
    tl.set_color('g')
ax2.set_xlim([0, 3000])

ax2.legend((h1, h2, h3),('As received','After 7 days heated in air','Temperature'),loc='best')


# # the arrow units are the axes
#ax2.arrow(2500, 110, 360, 500, 0, color='g', head_width=5, head_length=150)

for ext in ('.png','.eps','.pdf'):
    plt.savefig('figures/fig5' + ext)
plt.show()
```

References

(1) Dominik, C. *The Org-Mode 7 Reference Manual: Organize Your Life with GNU Emacs*; Network Theory: UK, 2010.

(2) Schulte, E.; Davison, D. Active Documents with Org-Mode. *Computing in Science Engineering* **2011**, *13*, 66 –73.

(3) Schulte, E.; Davison, D.; Dye, T.; Dominik, C. A Multi-Language Computing Environment for Literate Programming and Reproducible Research. *Journal of Statistical Software* **2012**, *46*, 1–24.