

Supplementary Data: Plasma-Catalytic Ammonia Synthesis

Beyond the Equilibrium Limit

Prateek Mehta,¹ Patrick Barboun,¹ Yannick Engelmann,² David B. Go,^{1,3}

Annemie Bogaerts,^{2,*} William F. Schneider,^{1,†} and Jason C. Hicks^{1,‡}

¹*Department of Chemical and Biomolecular Engineering,
University of Notre Dame, Notre Dame, Indiana 46556, United States*

²*Department of Chemistry, Antwerp University,
Campus Drie Eiken, Universiteitsplein 1, 2610 Wilrijk*

³*Department of Aerospace and Mechanical Engineering,
University of Notre Dame, Notre Dame, Indiana 46556, United States*

(Dated: December 2, 2019)

* annemie.bogaerts@uantwerpen.be

† wschneider@nd.edu

‡ jhicks3@nd.edu

SUPPLEMENTARY DATA

Source code and raw data is provided in an external Zenodo repository at [link](#). The python class `simpleMkm` in `simplemkm.py` contains the core functions necessary to perform the microkinetic calculations, while the class `mkmRunner` in `mkmutils.py` contains utility functions to run the calculations in an automated fashion on our computing cluster. The input (`*.mkminp`) and output (`*.mkmout`) files for the microkinetic model are included in the Zenodo directory. Raw experimental data is also provided as an Excel spreadsheet (`high-T-expts.xlsx`). Example python scripts to perform the calculations in this work, and to create the figures in the main text are provided below. These scripts were executed within an Emacs org-mode document (`supporting-data.org`), which was then exported to create this supplementary data pdf file.

Using the kinetic model: Plasma-off calculations

```
1 from utils import cd
2 from mkmutils import mkmRunner as runner
3
4 EAs = [-1.2, -0.6, 0.0]
5
6 plasma_on = False
7 rxn2_barrier = True
8
9 for EA in EAs:
10     mod = runner(EA,
11                  plasma_on=plasma_on,
12                  rxn2_barrier=rxn2_barrier,
13                  npts=100,
14                  concentration_based=True)
15
16     with cd('mkm-calcs/{0}'.format(mod.prefix)):
17         mod.write_input()
18         mod.run_job()
```

Using the kinetic model: Plasma-on calculations

```
1 from utils import cd
2 from mkmutils import mkmRunner as runner
```

```

3
4  EAs = [-1.2, -0.6, 0.0]
5
6  kei = range(-15, -12, 1)
7  ks_eimpact=[float('1.0e{0}'.format(i)) for i in kei] # cm3 / s
8
9  # Excitation energy, eV
10 Evib_A2 = 1.0
11
12 for EA in EAs:
13     for k_eimpact in ks_eimpact:
14         mod = runner(EA,
15                     plasma_on=True,
16                     k_eimpact_A2=k_eimpact,
17                     npts=200,
18                     rxn2_barrier=True,
19                     excite_type='vib',
20                     Evib_A2=Evib_A2)
21
22     with cd('mkm-calcs/{0}'.format(mod.prefix)):
23         mod.write_input(ncores=2)
24         mod.run_job()

```

Figure 2: Modeled plasma-off NH₃ yields

```

1  from simplemkm import simpleMkm as mkm
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from mkmutils import *
5
6  plt.style.use('seaborn-paper')
7  plt.rcParams["font.family"] = "Helvetica"
8
9  fig = plt.figure(figsize = (3.5, 3.25), dpi=200)
10
11  rates = []
12
13  EAmetal = [-1.2, -0.6, 0.0]
14
15
16  for EA in EAmetal:
17      mod = mkmRunner(EA, plasma_on=False, rxn2_barrier=True)
18      prefix = mod.prefix
19      d = load_variables('mkm-calcs/{0}/{0}.mkmout'.format(prefix))

```

```

20
21     allpressures = d['pressures']
22     pABs = []
23
24     for p in allpressures:
25         # N2, H2, and NH3 pressures
26         pA2, pB, pAB = p
27         pABs.append(np.float(pAB))
28
29     plt.plot(d['T'], pABs, '-',
30             label='$E_{\mathrm{N}} = {}$ eV'.format(EA))
31
32     Xeq = [float(x) for x in d['Xeq']]
33     pABeq = [float(x[-1]) for x in d['eq_pressures']]
34
35     plt.plot(d['T'], pABeq, c='k', ls='--', label='Eqb. limit')
36
37     plt.ylim(-0.001, 0.1)
38     plt.xlim(350, 1000)
39     plt.legend(frameon=False, fontsize=8)
40     plt.xlabel('Temperature (K)')
41     plt.ylabel('NH3 pressure (atm)')
42
43     # Inset
44     ax2 = fig.add_axes([0.65, 0.4, 0.25, 0.25])
45
46     EAs = np.linspace(1., -1.5, 150)
47     T = 473.
48
49     theta = 0.0
50
51     X = 0.05
52
53     rates = []
54     for i, EA in enumerate(EAs):
55         mod = mkm(T, EA, rxn2_barrier=True)
56
57         kf, kr = mod.get_rate_constants()
58         K2 = kf[1] / kr[1]
59         pA2, pB, pAB = mod.get_pressures(X)
60
61         theta = mod.integrate_odes(theta0=theta, X=X)[0]
62
63         try:
64             theta = mod.find_steady_state_roots(theta0=[theta], X=X)
65         except:
66             theta = mod.integrate_odes(theta0=theta, X=X)[0]

```

```

67         try:
68             theta = mod.find_steady_state_roots(theta0=[theta], X=X)
69         except:
70             pass
71
72     r = mod.get_rates(theta, mod.get_pressures(X))
73
74     if r[0] > 0:
75         ls = '-'
76     else:
77         ls = '--'
78
79     rates.append(abs(r[0]))
80 ax2.plot(EAs,
81          np.log10(rates),
82          ls,
83          label='$p_{\mathrm{AB}} = {0:1.3f}$ atm'.format(pAB), c='C7')
84
85 EAmetal = [-1.2, -0.6, 0.0]
86
87 for EA in EAmetal:
88     mod = mkm(T, EA, rxn2_barrier=True)
89
90     kf, kr = mod.get_rate_constants()
91     K2 = kf[1] / kr[1]
92     pA2, pB, pAB = mod.get_pressures(X)
93
94     theta = mod.integrate_odes(theta0=theta, X=X)[0]
95
96     try:
97         theta = mod.find_steady_state_roots(theta0=[theta], X=X)
98     except:
99         theta = mod.integrate_odes(theta0=theta, X=X)[0]
100     try:
101         theta = mod.find_steady_state_roots(theta0=[theta], X=X)
102     except:
103         pass
104
105     r = mod.get_rates(theta, mod.get_pressures(X))
106
107     if r[0] > 0:
108         ls = '-'
109     else:
110         ls = '--'
111
112     ax2.plot(EA, np.log10(abs(r[0])), 'o')
113

```

```

114 plt.ylim(-16, -4)
115 plt.xlim(-1.5, 0.5)
116
117 plt.yticks(np.arange(-15, 0, 5))
118
119 plt.xlabel('$E_{\mathrm{N}}$ (eV)')
120 plt.ylabel('log$_{10}$ (TOF [s$^{-1}$])')
121
122 plt.tight_layout()
123
124 for ext in ['eps', 'pdf', 'png']:
125     plt.savefig('figures/thermal-pNH3.{0}'.format(ext), dpi=200)
126 plt.show()

```

Figure 3: Modeled plasma-on NH₃ yields

```

1 from mkmutils import *
2 import matplotlib.pyplot as plt
3 from mkmutils import mkRunner as runner
4
5 plt.style.use('seaborn-paper')
6 plt.rcParams["font.family"] = "Helvetica"
7
8 # Barrier reduced by,
9 Evib_A2 = 1.0
10
11 plt.figure(figsize=(3, 4), dpi=200)
12
13 grid = plt.GridSpec(3, 1, hspace=0)
14 ax1 = plt.subplot(grid[0, 0])
15 ax2 = plt.subplot(grid[1, 0], sharex=ax1)
16 ax3 = plt.subplot(grid[2, 0], sharex=ax1)
17
18 axes = [ax1, ax2, ax3]
19
20 EAs = [-1.2, -0.6, 0.0]
21 kei = range(-15, -12, 1)
22 ks_eimpact=[float('1.0e{0}'.format(i)) for i in kei] # cm3 / s
23
24 plasma_on = True
25 rxn2_barrier = True
26
27 for ax, EA in zip(axes, EAs):
28

```

```

29     for c, k_eimpact in zip(['tan', 'palevioletred', 'seagreen'], ks_eimpact):
30         mod = runner(EA,
31                       plasma_on=plasma_on,
32                       k_eimpact_A2=k_eimpact,
33                       rxn2_barrier=rxn2_barrier,
34                       excite_type='vib',
35                       Evib_A2=Evib_A2)
36
37     prefix = mod.prefix
38     try:
39         d = load_variables('mkm-calcs/{0}/{0}.mkmout'.format(prefix))
40
41         if plasma_on:
42             inp = mod.input_params
43             kene = inp['k_eimpact_A2'] * inp['n_e']
44             allpressures = d['pressures']
45             pABs = []
46
47             for p in allpressures:
48                 pA2, pA2prime, pB, pAB, pABprime = p
49                 pABs.append(np.float(pAB))
50
51             label = '$k_{e} n_{e} = 10^{{{0:1.0f}}}$ s$^{{-1}}$'.format(np.log10(kene)))
52             line, = ax.plot(d['T'],
53                             pABs,
54                             '-',
55                             c=c,
56                             label=label)
57
58     except Exception as E:
59         print prefix, E
60
61     modoff = runner(EA,
62                     plasma_on=False,
63                     rxn2_barrier=rxn2_barrier)
64
65     prefix = modoff.prefix
66     doff = load_variables('mkm-calcs/{0}/{0}.mkmout'.format(prefix))
67
68     allpressures = doff['pressures']
69     pABsoff = []
70
71     for p in allpressures:
72         pA2, pB, pAB = p
73         pABsoff.append(np.float(pAB))
74
75     pABsoff2plot = [pAB for i, pAB in enumerate(pABsoff) if i % 4. == 0]

```

```

76     T2plot = [T for i, T in enumerate(doff['T']) if i % 4. == 0]
77
78     ax.plot(T2plot, pABsoff2plot, 'o', ms=4, mew=1, mfc='None', mec='C5', label='plasma off')
79
80     pABeq = [float(x[-1]) for x in d['eq_pressures']]
81
82     ax.plot(d['T'], pABeq, c='k', ls='--', label='Eqb. limit')
83
84     ax.set_ylim(-0.01, 0.3)
85     ax.set_yticks([0.0, 0.1, 0.2])
86     ax.set_yticklabels([0.0, 0.1, 0.2], fontsize=7.5)
87     ax.set_ylabel('NH3 pressure (atm)', fontsize=7.5)
88
89     ax1, ax2, ax3 = axes
90
91     ax1.legend(fontsize=6, frameon=False, ncol=2, columnspacing=1, handlelength=1.5)
92
93     plt.setp(ax1.get_xticklabels(), visible=False)
94     plt.setp(ax2.get_xticklabels(), visible=False)
95
96     ax3.set_xlim(350, 950)
97     ax3.set_xlabel('Temperature (K)', fontsize=7.5)
98
99     plt.figtext(0.22, 0.93, 'a', fontsize=8, fontweight='bold')
100    plt.figtext(0.22, 0.65, 'b', fontsize=8, fontweight='bold')
101    plt.figtext(0.22, 0.37, 'c', fontsize=8, fontweight='bold')
102
103    plt.figtext(0.7, 0.75, '$E_{N} = -1.2$ eV', fontsize=7.5)
104    plt.figtext(0.7, 0.5, '$E_{N} = -0.6$ eV', fontsize=7.5)
105    plt.figtext(0.74, 0.32, '$E_{N} = 0.0$ eV', fontsize=7.5)
106
107    plt.tight_layout()
108    for ext in ['eps', 'png', 'pdf']:
109        plt.savefig('figures/plasma-on-NH3-syn.{0}'.format(ext), dpi=200)
110
111    plt.show()

```

Figure 4: Experimental plasma-only NH₃ yields

```

1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  plt.style.use('seaborn-paper')
5  plt.rcParams["font.family"] = "Helvetica"

```



```

6
7 plt.figure(figsize = (3.25, 3), dpi=200)
8
9 data = pd.read_excel('high-T-expts.xlsx',
10                     'eqb-data',
11                     header=0)
12
13 plt.plot(data['T(K)'], data['X'] * 100, 'k--', label='Eqb. limit')
14
15 data = pd.read_excel('high-T-expts.xlsx', 'plasma-sweep', header=1)
16
17 plt.errorbar(data['Temperature.1'] + 273.15,
18             data['Conversion.1'],
19             data['Conversion Error.1'],
20             capthick=1.5,
21             fmt=None,
22             label=None)
23
24 plt.plot(data['Temperature.1'] + 273.15,
25         data['Conversion.1'],
26         'o',
27         ms=6,
28         mew=1.5,
29         mfc='w',
30         c='C0',
31         label='10 W')
32
33 plt.errorbar(data['Temperature'] + 273.15,
34             data['Conversion'],
35             data['Conversion Error'],
36             c='tan',
37             fmt=None,
38             capthick=1.5,
39             label=None)
40
41 plt.plot(data['Temperature'] + 273.15,
42         data['Conversion'],
43         'o',
44         ms=6,
45         mew=1.5,
46         mfc='w',
47         c='tan',
48         label='5 W')
49
50 plt.errorbar(data['Temperature.2'] + 273.15,
51             data['Conversion.2'],
52             data['Conversion Error.2'],

```

```

53         c='C3', capthick=1.5,
54         fmt=None,
55         label=None)
56 plt.plot(data['Temperature.2'] + 273.15,
57          data['Conversion.2'], 'o',
58          ms=6, mew=1.5,
59          mfc='w',
60          c='C3',
61          label='15 W')
62
63 plt.ylim(0, 4)
64 plt.xlim(400, 1173)
65 plt.xlabel('Temperature (K)')
66 plt.ylabel('NH3 Yield (%)')
67 plt.legend(frameon=False, fontsize=8, ncol=2)
68 plt.tight_layout()
69
70
71 for ext in ['eps', 'pdf', 'png']:
72     plt.savefig('figures/NH3-power-expts.{0}'.format(ext), dpi=200)
73 plt.show()

```

Figure 5: Experimental plasma-catalytic NH₃ yields

```

1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  plt.style.use('seaborn-paper')
5  plt.rcParams["font.family"] = "Helvetica"
6
7  plt.figure(figsize = (4, 5), dpi=200)
8
9
10 ax2 = plt.subplot(212)
11
12 data = pd.read_excel('high-T-expts.xlsx', 'NH3-decomposition', header=1)
13
14 plt.plot(data['Temperature.2'] + 273.15,
15          data['Conversion.2'],
16          'o',
17          ms=6,
18          mew=1.5,
19          mfc='w',
20          c='C0',

```

```

21         label='Al$_{2}$O$_{3}$')
22     plt.errorbar(data['Temperature.1'] + 273.15,
23                 data['Conversion.1'],
24                 data['Error.1'],
25                 capthick=1.5, c='C1',
26                 fmt=None,
27                 label=None)
28
29     plt.plot(data['Temperature.1'] + 273.15,
30             data['Conversion.1'],
31             '^', ms=6, mew=1.5,
32             mfc='w',
33             c='C1',
34             label='Ni/Al$_{2}$O$_{3}$')
35
36     plt.errorbar(data['Temperature'] + 273.15,
37                 data['Conversion'],
38                 data['Error'],
39                 fmt=None,
40                 c='C2',
41                 capthick=1.5,
42                 label=None)
43
44     plt.plot(data['Temperature'] + 273.15,
45             data['Conversion'],
46             's',
47             ms=6,
48             mew=1.5,
49             mfc='w',
50             c='C2',
51             label='Pt/Al$_{2}$O$_{3}$')
52
53     plt.xlim(400, 1173)
54     plt.ylim(-5, 105)
55     plt.legend(frameon=False, fontsize=8)
56     plt.xlabel('Temperature (K)', fontsize=10)
57     plt.xticks(fontsize=10)
58     plt.yticks(fontsize=10)
59     plt.ylabel('NH$_{3}$ Conversion (%)', fontsize=10)
60
61     ax1 = plt.subplot(211, sharex=ax2)
62
63     data = pd.read_excel('high-T-expts.xlsx', 'eqb-data', header=0)
64     plt.plot(data['T(K)'], data['X'] * 100, 'k--', label='Eqb. limit')
65
66     data = pd.read_excel('high-T-expts.xlsx', 'metal-sweep', header=1)
67

```

```

68 plt.errorbar(data['Temperature.2'] + 273.15,
69              data['Conversion.2'],
70              data['Conversion Error.2'],
71              capthick=1.5,
72              fmt=None,
73              label=None)
74 plt.plot(data['Temperature.2'] + 273.15,
75          data['Conversion.2'],
76          'o',
77          ms=6,
78          mew=1.5,
79          mfc='w',
80          c='C0',
81          label='Al$_{2}$O$_{3}$')
82
83 plt.errorbar(data['Temperature.1'] + 273.15,
84              data['Conversion.1'],
85              data['Conversion Error.1'],
86              capthick=1.5,
87              fmt=None,
88              label=None)
89 plt.plot(data['Temperature.1'] + 273.15,
90          data['Conversion.1'],
91          '^',
92          ms=6,
93          mew=1.5,
94          mfc='w',
95          c='C1',
96          label='Ni/Al$_{2}$O$_{3}$')
97
98 plt.errorbar(data['Temperature'] + 273.15,
99              data['Conversion'],
100             data['Conversion Error'],
101             fmt=None,
102             capthick=1.5,
103             label=None)
104 plt.plot(data['Temperature'] + 273.15,
105          data['Conversion'],
106          's',
107          ms=6,
108          mew=1.5,
109          mfc='w',
110          c='C2',
111          label='Pt/Al$_{2}$O$_{3}$')
112
113 plt.ylim(0, 2.5)
114 plt.xlim(400, 1173)

```

```
115 plt.yticks(fontsize=10)
116 plt.setp(ax1.get_xticklabels(), visible=False)
117
118 plt.ylabel('NH3 Yield (%)', fontsize=10)
119 plt.legend(frameon=False, fontsize=8)
120 plt.tight_layout()
121
122 plt.figtext(0.03, 0.96, 'a', fontsize=12, fontweight='bold')
123 plt.figtext(0.03, 0.49, 'b', fontsize=12, fontweight='bold')
124
125
126 for ext in ['eps', 'pdf', 'png']:
127     plt.savefig('figures/NH3-plasma-metal.{0}'.format(ext), dpi=200)
128
129 plt.show()
```
