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 9, 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 [1]. 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

```python
from utils import cd
from mkmutils import mkmRunner as runner

EAs = [-1.2, -0.6, 0.0]

plasma_on = False
rxn2_barrier = True

for EA in EAs:
    mod = runner(EA,
                 plasma_on=plasma_on,
                 rxn2_barrier=rxn2_barrier,
                 npts=100,
                 concentration_based=True)

    with cd('mkm-calcs/{0}'.format(mod.prefix)):
        mod.write_input()
        mod.run_job()
```

Using the kinetic model: Plasma-on calculations

```python
from utils import cd
from mkmutils import mkmRunner as runner
```

```

EAs = [-1.2, -0.6, 0.0]

kei = range(-15, -12, 1)
ks_eimpact=[float('1.0e{0}'.format(i)) for i in kei]  # cm3 / s

# Excitation energy, eV
Evib_A2 = 1.0

for EA in EAs:
    for k_eimpact in ks_eimpact:
        mod = runner(EA,
                     plasma_on=True,
                     k_eimpact_A2=k_eimpact,
                     npts=200,
                     rxn2_barrier=True,
                     excite_type='vib',
                     Evib_A2=Evib_A2)

        with cd('mkm-calcs/{0}'.format(mod.prefix)):
                mod.write_input(ncores=2)
                mod.run_job()
```

Figure 2: Modeled plasma-off NH_3 yields

```
from simplemkm import simpleMkm as mkm
import numpy as np
import matplotlib.pyplot as plt
from mkmutils import *

plt.style.use('seaborn-paper')
plt.rcParams["font.family"] = "Helvetica"

fig = plt.figure(figsize = (3.5, 3.25), dpi=200)

rates = []

EAmetal = [-1.2, -0.6, 0.0]


for EA in EAmetal:
    mod = mkmRunner(EA, plasma_on=False, rxn2_barrier=True)
    prefix = mod.prefix
    d = load_variables('mkm-calcs/{0}/{0}.mkmout'.format(prefix))
```

```python

    allpressures = d['pressures']
    pABs = []

    for p in allpressures:
        # N2, H2, and NH3 pressures
        pA2, pB, pAB = p
        pABs.append(np.float(pAB))

    plt.plot(d['T'], pABs, '-',
             label='$E_{{\mathrm{{N}}}} = {0}$ eV'.format(EA))

Xeq = [float(x) for x in d['Xeq']]
pABeq = [float(x[-1]) for x in d['eq_pressures']]

plt.plot(d['T'], pABeq, c='k', ls='--', label='Eqb. limit')

plt.ylim(-0.001, 0.1)
plt.xlim(350, 1000)
plt.legend(frameon=False, fontsize=8)
plt.xlabel('Temperature (K)')
plt.ylabel('NH$_{3}$ pressure (atm)')

# Inset
ax2 = fig.add_axes([0.65, 0.4, 0.25, 0.25])

EAs = np.linspace(1., -1.5, 150)
T = 473.

theta = 0.0

X = 0.05

rates = []
for i, EA in enumerate(EAs):
    mod = mkm(T, EA, rxn2_barrier=True)

    kf, kr = mod.get_rate_constants()
    K2 = kf[1] / kr[1]
    pA2, pB, pAB = mod.get_pressures(X)

    theta = mod.integrate_odes(theta0=theta, X=X)[0]

    try:
        theta = mod.find_steady_state_roots(theta0=[theta], X=X)
    except:
        theta = mod.integrate_odes(theta0=theta, X=X)[0]
```

```
        try:
            theta = mod.find_steady_state_roots(theta0=[theta], X=X)
        except:
            pass

    r = mod.get_rates(theta, mod.get_pressures(X))

    if r[0] > 0:
        ls = '-'
    else:
        ls = '--'

    rates.append(abs(r[0]))
ax2.plot(EAs,
         np.log10(rates),
         ls,
         label='$p_{{\mathrm{{AB}}}} = {0:1.3f}$ atm'.format(pAB), c='C7')

EAmetal = [-1.2, -0.6, 0.0]

for EA in EAmetal:
    mod = mkm(T, EA, rxn2_barrier=True)

    kf, kr = mod.get_rate_constants()
    K2 = kf[1] / kr[1]
    pA2, pB, pAB = mod.get_pressures(X)

    theta = mod.integrate_odes(theta0=theta, X=X)[0]

    try:
        theta = mod.find_steady_state_roots(theta0=[theta], X=X)
    except:
        theta = mod.integrate_odes(theta0=theta, X=X)[0]
        try:
            theta = mod.find_steady_state_roots(theta0=[theta], X=X)
        except:
            pass

    r = mod.get_rates(theta, mod.get_pressures(X))

    if r[0] > 0:
        ls = '-'
    else:
        ls = '--'

    ax2.plot(EA, np.log10(abs(r[0])), 'o')

```

```python
plt.ylim(-16, -4)
plt.xlim(-1.5, 0.5)

plt.yticks(np.arange(-15, 0, 5))

plt.xlabel('$E_{\mathrm{N}}$ (eV)')
plt.ylabel('log$_{10}$(TOF [s$^{-1}$])')

plt.tight_layout()

for ext in ['eps', 'pdf', 'png']:
    plt.savefig('figures/thermal-pNH3.{0}'.format(ext), dpi=200)
plt.show()
```

Figure 3: Modeled plasma-on NH_3 yields

```python
from mkmutils import *
import matplotlib.pyplot as plt
from mkmutils import mkmRunner as runner

plt.style.use('seaborn-paper')
plt.rcParams["font.family"] = "Helvetica"

# Barrier reduced by,
Evib_A2 = 1.0

plt.figure(figsize=(3, 4), dpi=200)

grid = plt.GridSpec(3, 1, hspace=0)
ax1 = plt.subplot(grid[0, 0])
ax2 = plt.subplot(grid[1, 0], sharex=ax1)
ax3 = plt.subplot(grid[2, 0], sharex=ax1)

axes = [ax1, ax2, ax3]

EAs = [-1.2, -0.6, 0.0]
kei = range(-15, -12, 1)
ks_eimpact=[float('1.0e{0}'.format(i)) for i in kei]  # cm3 / s

plasma_on = True
rxn2_barrier = True

for ax, EA in zip(axes, EAs):

```

```
    for c, k_eimpact in zip(['tan', 'palevioletred', 'seagreen'], ks_eimpact):
        mod = runner(EA,
                     plasma_on=plasma_on,
                     k_eimpact_A2=k_eimpact,
                     rxn2_barrier=rxn2_barrier,
                     excite_type='vib',
                     Evib_A2=Evib_A2)

        prefix = mod.prefix
        try:
            d = load_variables('mkm-calcs/{0}/{0}.mkmout'.format(prefix))

            if plasma_on:
                inp = mod.input_params
                kene = inp['k_eimpact_A2'] * inp['n_e']
                allpressures = d['pressures']
                pABs = []

                for p in allpressures:
                    pA2, pA2prime, pB, pAB, pABprime = p
                    pABs.append(np.float(pAB))

                label = '$k_{{e}} n_{{e}} = 10^{{{0:1.0f}}}$ s$^{{-1}}$'.format(np.log10(kene))
                line, = ax.plot(d['T'],
                                pABs,
                                '-',
                                c=c,
                                label=label)

        except Exception as E:
            print prefix, E

    modoff = runner(EA,
                    plasma_on=False,
                    rxn2_barrier=rxn2_barrier)

    prefix = modoff.prefix
    doff = load_variables('mkm-calcs/{0}/{0}.mkmout'.format(prefix))

    allpressures = doff['pressures']
    pABsoff = []

    for p in allpressures:
        pA2, pB, pAB = p
        pABsoff.append(np.float(pAB))

    pABsoff2plot = [pAB for i, pAB in enumerate(pABsoff) if i % 4. == 0]
```

```
    T2plot = [T for i, T in enumerate(doff['T']) if i % 4. == 0]

    ax.plot(T2plot, pABsoff2plot, 'o', ms=4, mew=1, mfc='None', mec='C5', label='plasma off')

    pABeq = [float(x[-1]) for x in d['eq_pressures']]

    ax.plot(d['T'], pABeq, c='k', ls='--', label='Eqb. limit')

    ax.set_ylim(-0.01, 0.3)
    ax.set_yticks([0.0, 0.1, 0.2])
    ax.set_yticklabels([0.0, 0.1, 0.2], fontsize=7.5)
    ax.set_ylabel('NH$_{3}$ pressure (atm)', fontsize=7.5)

ax1, ax2, ax3 = axes

ax1.legend(fontsize=6, frameon=False, ncol=2, columnspacing=1, handlelength=1.5)

plt.setp(ax1.get_xticklabels(), visible=False)
plt.setp(ax2.get_xticklabels(), visible=False)

ax3.set_xlim(350, 950)
ax3.set_xlabel('Temperature (K)', fontsize=7.5)

plt.figtext(0.22, 0.93, 'a', fontsize=8, fontweight='bold')
plt.figtext(0.22, 0.65, 'b', fontsize=8, fontweight='bold')
plt.figtext(0.22, 0.37, 'c', fontsize=8, fontweight='bold')

plt.figtext(0.7, 0.75, '$E_{N} = -1.2$ eV', fontsize=7.5)
plt.figtext(0.7, 0.5, '$E_{N} = -0.6$ eV', fontsize=7.5)
plt.figtext(0.74, 0.32, '$E_{N} = 0.0$ eV', fontsize=7.5)

plt.tight_layout()
for ext in ['eps', 'png', 'pdf']:
    plt.savefig('figures/plasma-on-NH3-syn.{0}'.format(ext), dpi=200)

plt.show()
```

Figure 4: Experimental plasma-only NH_3 yields

```
import pandas as pd
import matplotlib.pyplot as plt

plt.style.use('seaborn-paper')
plt.rcParams["font.family"] = "Helvetica"
```

```
plt.figure(figsize = (3.25, 3), dpi=200)

data = pd.read_excel('high-T-expts.xlsx',
                     'eqb-data',
                     header=0)

plt.plot(data['T(K)'], data['X'] * 100, 'k--', label='Eqb. limit')

data = pd.read_excel('high-T-expts.xlsx', 'plasma-sweep', header=1)

plt.errorbar(data['Temperature.1'] + 273.15,
             data['Conversion.1'],
             data['Conversion Error.1'],
             capthick=1.5,
             fmt=None,
             label=None)

plt.plot(data['Temperature.1'] + 273.15,
         data['Conversion.1'],
         'o',
         ms=6,
         mew=1.5,
         mfc='w',
         c='C0',
         label='10 W')

plt.errorbar(data['Temperature'] + 273.15,
             data['Conversion'],
             data['Conversion Error'],
             c='tan',
             fmt=None,
             capthick=1.5,
             label=None)

plt.plot(data['Temperature'] + 273.15,
         data['Conversion'],
         'o',
         ms=6,
         mew=1.5,
         mfc='w',
         c='tan',
         label='5 W')

plt.errorbar(data['Temperature.2'] + 273.15,
             data['Conversion.2'],
             data['Conversion Error.2'],
```

```python
             c='C3', capthick=1.5,
             fmt=None,
             label=None)
plt.plot(data['Temperature.2'] + 273.15,
         data['Conversion.2'], 'o',
         ms=6, mew=1.5,
         mfc='w',
         c='C3',
         label='15 W')

plt.ylim(0, 4)
plt.xlim(400, 1173)
plt.xlabel('Temperature (K)')
plt.ylabel('NH$_{3}$ Yield (%)')
plt.legend(frameon=False, fontsize=8, ncol=2)
plt.tight_layout()


for ext in ['eps', 'pdf', 'png']:
    plt.savefig('figures/NH3-power-expts.{0}'.format(ext), dpi=200)
plt.show()
```

Figure 5: Experimental plasma-catalytic NH_3 yields

```python
import pandas as pd
import matplotlib.pyplot as plt

plt.style.use('seaborn-paper')
plt.rcParams["font.family"] = "Helvetica"

plt.figure(figsize = (4, 5), dpi=200)



ax2 = plt.subplot(212)

data = pd.read_excel('high-T-expts.xlsx', 'NH3-decomposition', header=1)

plt.plot(data['Temperature.2'] + 273.15,
         data['Conversion.2'],
         'o',
         ms=6,
         mew=1.5,
         mfc='w',
         c='C0',
```

```
         label='Al$_{2}$O$_{3}$')
plt.errorbar(data['Temperature.1'] + 273.15,
             data['Conversion.1'],
             data['Error.1'],
             capthick=1.5, c='C1',
             fmt=None,
             label=None)

plt.plot(data['Temperature.1'] + 273.15,
         data['Conversion.1'],
         '^', ms=6, mew=1.5,
         mfc='w',
         c='C1',
         label='Ni/Al$_{2}$O$_{3}$')

plt.errorbar(data['Temperature'] + 273.15,
             data['Conversion'],
             data['Error'],
             fmt=None,
             c='C2',
             capthick=1.5,
             label=None)

plt.plot(data['Temperature'] + 273.15,
         data['Conversion'],
         's',
         ms=6,
         mew=1.5,
         mfc='w',
         c='C2',
         label='Pt/Al$_{2}$O$_{3}$')

plt.xlim(400, 1173)
plt.ylim(-5, 105)
plt.legend(frameon=False, fontsize=8)
plt.xlabel('Temperature (K)', fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.ylabel('NH$_{3}$ Conversion (%)', fontsize=10)

ax1 = plt.subplot(211, sharex=ax2)

data = pd.read_excel('high-T-expts.xlsx', 'eqb-data', header=0)
plt.plot(data['T(K)'], data['X'] * 100, 'k--', label='Eqb. limit')

data = pd.read_excel('high-T-expts.xlsx', 'metal-sweep', header=1)

```

```
plt.errorbar(data['Temperature.2'] + 273.15,
             data['Conversion.2'],
             data['Conversion Error.2'],
             capthick=1.5,
             fmt=None,
             label=None)
plt.plot(data['Temperature.2'] + 273.15,
         data['Conversion.2'],
         'o',
         ms=6,
         mew=1.5,
         mfc='w',
         c='C0',
         label='Al$_{2}$O$_{3}$')

plt.errorbar(data['Temperature.1'] + 273.15,
             data['Conversion.1'],
             data['Conversion Error.1'],
             capthick=1.5,
             fmt=None,
             label=None)
plt.plot(data['Temperature.1'] + 273.15,
         data['Conversion.1'],
         '^',
         ms=6,
         mew=1.5,
         mfc='w',
         c='C1',
         label='Ni/Al$_{2}$O$_{3}$')

plt.errorbar(data['Temperature'] + 273.15,
             data['Conversion'],
             data['Conversion Error'],
             fmt=None,
             capthick=1.5,
             label=None)
plt.plot(data['Temperature'] + 273.15,
         data['Conversion'],
         's',
         ms=6,
         mew=1.5,
         mfc='w',
         c='C2',
         label='Pt/Al$_{2}$O$_{3}$')

plt.ylim(0, 2.5)
plt.xlim(400, 1173)
```

```
plt.yticks(fontsize=10)
plt.setp(ax1.get_xticklabels(), visible=False)

plt.ylabel('NH$_{3}$ Yield (%)', fontsize=10)
plt.legend(frameon=False, fontsize=8)
plt.tight_layout()

plt.figtext(0.03, 0.96, 'a', fontsize=12, fontweight='bold')
plt.figtext(0.03, 0.49, 'b', fontsize=12, fontweight='bold')


for ext in ['eps', 'pdf', 'png']:
    plt.savefig('figures/NH3-plasma-metal.{0}'.format(ext), dpi=200)

plt.show()
```

REFERENCES

[1] Mehta, P. *et al.* wfschneidergroup/SI-mehta-beyond-eq (2019). URL https://doi.org/10.5281/zenodo.3566833.