2.2 Generation Consumption Analysis

Summary

This Jupyter notebook demonstrate the use of the symbolic algebra package `Sympy` for the generation/consumption analysis for the production of ammonia using basic principles of reaction stoichiometry.

Example: Ammonia Production

BASF, headquartered in Ludwigshafen, Germany, is the largest chemical company in the world. In 1913, under its original name Badische Anilin- und Soda-Fabrik, BASF commercialized the Haber-Bosch process for the production of ammonia from natural gas, water, and air.

Prior to this invention, American and European agriculture was dependent on guano mined from the 'Guano Islands' in the Caribbean Sea and Pacific Ocean, and saltpeter mined from the deserts of Peru, Chile, and Bolivia. The competition for these limited resouces led to the notorious U.S. Guano Islands Act of 1856, and multiple wars (the Guano War, the War of the Pacific, later resulting in acute fertilizer shortages that was called 'the Wheat Problem' in England by Sir William Crookes in 1898.

The following video produced by BASF provides a technical overview of the Haber-Bosch process.

In [1]:

```
from IPython.display import YouTubeVideo
YouTubeVideo("uMkzxV_y7tY",560,315,rel=0)
```

Out[1]:

Problem Statement

Consider three reactions for the production of ammonia

1. Steam-reforming of methane

   $CH_4 + H_2O \rightarrow CO + 3\ H_2$

2. Water-gas shift

   $CO + H_2O \rightarrow CO_2 + H_2$

3. Haber-Bosch reaction

   $N_2 + 3\ H_2 \rightarrow 2\ NH_3$

Determine if it is possible to construct a process for the production of ammonia with no wasted hydrogen and no release of carbon monoxide.

Solution

We begin by setting up the stoichiometric matrix for generation/consumption analysis

Species	R_1	R_2	R_3	Net
	χ_1	χ_2	χ_3	$\sum_k \nu_k \chi_k$
CH_4	-1	0	0	≤ 0
H_2O	-1	-1	0	≤ 0
CO	1	-1	0	0
H_2	3	1	-3	0
CO_2	0	1	0	≥ 0
N_2	0	0	1	≤ 0
NH_3	0	0	2	1

which includes three equality constraints which need to be solved for χ_1, χ_2, and χ_3.

The first step is to import `sympy`.

In [2]:

```
import sympy
```

When imported in this way, the functions from `sympy` must be accessed with the prefix `sympy.`. This avoids overwriting functions with the same name as those in `sympy`, such as `plot`.

Next we use the `sympy.var` function to create three symbolic variables corresponding to χ_1, χ_2, and χ_3.

In [3]:

```
sympy.var('x1 x2 x3')
```

Out[3]:

```
(x1, x2, x3)
```

The net stoichiometric coefficients can be written in terms of the symbolic variables.

In [4]:

```
v = dict()
v['CH4'] = -x1
v['H2O'] = -x1 - x2
v['CO']  = x1 - x2
v['H2']  = 3*x1 + x2 - 3*x3
v['CO2'] = x2
v['N2'] = -x3
v['NH3'] = 2*x3
```

The three process constraints are encoded as equations using the sympy function `Eq()`

In [5]:

```
eqns = [
    sympy.Eq(v['NH3'],1),
    sympy.Eq(v['CO'],0),
    sympy.Eq(v['H2'],0)
]
```

These equations are solved for χ_1, χ_2, and χ_3.

In [6]:

```
soln = sympy.solve(eqns)
print(soln)
```

```
{x2: 3/8, x3: 1/2, x1: 3/8}
```

To finish the problem, the solutions are substituted back into the expressions for the stoichiometric coefficients, and the non-zero coefficients are displayed.

In [7]:

```
for k in v.keys():
    a = v[k].subs(soln)
    if a != 0:
        print("{0:<3s} {1:>6s}".format(k,str(a)))
```

```
N2    -1/2
H2O   -3/4
NH3     1
CH4   -3/8
CO2    3/8
```

In [ ]:

```

```