Referencing_QM_data_to_anchor_species

July 30, 2024

Supplementary notebook to the manuscript "Unifying thermochemistry concepts in computational heterogeneous catalysis", by Dr. Bjarne Kreitz (Brown University), Dr. Gabriel S. Gusmão (Georgia Tech), Dingqi Nai (Georgia Tech), Sushree Jagriti Sahoo (Georgia Tech), Prof. Andrew A. Peterson (Brown University), Dr. David H. Bross (Argonne National Laboratory), Prof. C. Franklin Goldsmith (Brown University), Prof. Andrew J. Medford (Georgia Tech).

Correspondence to Bjarne Kreitz (bjarne_kreitz@brown.edu) and Andrew J. Medford (ajm@gatech.edu)

[21]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

eV_to_kJmol=96.485
```

1 Referencing QM data to a set of anchor species

Relative enthalpies of species (gas phase or adsorbed) can be calculated from the DFT energies by anchoring them to a set of reference species. A reference species is selected for every element to form an atomic basis set for the target species. These reference species are combined in a hypothetical reaction to form the target P, which is a generic $C_xH_yO_z$ gas-phase species or $C_xH_yO_z^*$ adsorbate. Common choices of reference species are $[CH_4, H_2, H_2O]$ and in the case of Pt(111), the empty slab has to be added as a reference species.

$$\nu_{CH_4} CH_4 + \nu_{H_2O} H_2O + \nu_{H_2} H_2 + \nu_* * \rightarrow P^* \tag{1}$$

The enthalpy of reaction of the target from the reference species is defined as

$$\Delta_{\mathrm{A}} H_{\mathrm{P}}(0K) = \Delta_{\mathrm{r}} H \tag{2}$$

The reaction enthalpy can be determined from the DFT energies via

$$\Delta_{\mathrm{A}} H_{\mathrm{P}}(0K) = E_{\mathrm{P}} + \sum_{i \neq P}^{N} \nu_i E_i \tag{3}$$

In matrix notation this reads as

$$\underline{\mathbf{H}}_{\mathrm{A}} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^{\mathrm{A}} \tag{4}$$

where $\underline{\underline{\mathbf{M}}}$ can be determined from the elemental composition matrix of the target species $\underline{\underline{\mathbf{N}}}$ and the reference species $\underline{\underline{\mathbf{N}}}^{\mathrm{A}}$

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{\mathrm{A}^{-1}} \tag{5}$$

This is the list of species with their DFT energies from BEEF-vdW for which we want to compute the relative enthalpies. We can create a vector $\underline{\mathbf{E}}$ that contains the energies from this dictionary.

[2]:
```
#Target species for which we need to compute the EOFs
species_energies = {
    "C2H6": -615.3019939,
    "C2H6X" : -378231.5655,
    "XH": -377632.6636,
    "XCH2CH3": -378214.9168,
    "XO": -378193.2832,
    "XCH2XCH2": -378198.4442,
    "XCO": -378453.0261,
    "CO2X": -379030.8164,
    "CO2": -1414.682864,
    "XOH": -378209.8045,
    "H2OX": -378227.2801,
    "CO": -835.5389907,
    "C2H4":-581.44565067567,
}

#vector of DFT energies of the target species from the dictionary
E=np.array(list(species_energies.values()))
```

We want to determine the enthalpies relative to $[CH_4, H_2, H_2O, Pt(111)]$. For which we determined the DFT energies. We construct the vector $\underline{\mathbf{E}}^{\mathrm{A}}$ that contains the DFT energies of the reference species

[3]:
```
ref_energies = {
    "CH4": -324.2935569,
    "H2": -32.69844421,
    "H2O": -611.0186083,
    "X": -377616.072,
}

#vector of DFT energies of the reference species from the dictionary
E_A=np.array(list(ref_energies.values()))
```

We start now with constructing the elemental composition matrix of our target species $\underline{\underline{\mathbf{N}}}$, which is an $m \times n$ matrix with m target species and n elements.

[4]:
```
# Define the species and their elemental compositions in a dictionary
species_compositions = {
    "C2H6": {"C": 2, "H": 6, "O": 0, "X": 0},
    "C2H6X": {"C": 2, "H": 6, "O": 0, "X": 1},
    "XH": {"C": 0, "H": 1, "O": 0, "X": 1},
    "XCH2CH3": {"C": 2, "H": 5, "O": 0, "X": 1},
    "XO": {"C": 0, "H": 0, "O": 1, "X": 1},
    "XCH2XCH2": {"C": 2, "H": 4, "O": 0, "X": 1},
    "XCO": {"C": 1, "H": 0, "O": 1, "X": 1},
    "CO2X": {"C": 1, "H": 0, "O": 2, "X": 1},
    "CO2": {"C": 1, "H": 0, "O": 2, "X": 0},
    "XOH": {"C": 0, "H": 1, "O": 1, "X": 1},
    "H2OX": {"C": 0, "H": 2, "O": 1, "X": 1},
    "CO": {"C": 1, "H": 0, "O": 1, "X": 0},
    "C2H4": {"C": 2, "H": 4, "O": 0, "X": 0},
}

species=list(species_compositions.keys())


# Create a matrix to hold the elemental compositions of the target species
num_species = len(species_compositions)
num_elements = 4  # C, H, O, X
N = np.zeros((num_species, num_elements))

# Fill in the elemental composition matrix of the target species
for s, composition in species_compositions.items():
    i = species.index(s)
    N[i, :] = [composition["C"], composition["H"], composition["O"],␣
 ↪composition["X"]]

N
```

[4]:
```
array([[2., 6., 0., 0.],
       [2., 6., 0., 1.],
       [0., 1., 0., 1.],
       [2., 5., 0., 1.],
       [0., 0., 1., 1.],
       [2., 4., 0., 1.],
       [1., 0., 1., 1.],
       [1., 0., 2., 1.],
       [1., 0., 2., 0.],
       [0., 1., 1., 1.],
       [0., 2., 1., 1.],
       [1., 0., 1., 0.],
       [2., 4., 0., 0.]])
```

We construct the elemental composition matrix of our reference species $\underline{\underline{\mathbf{N}}}^{\mathrm{A}}$, which is an $m \times n$

matrix with m reference species and n elements.

[5]:
```
# Define the species and their elemental compositions in a dictionary
references_compositions = {
    "CH4": {"C": 1, "H": 4, "O": 0, "X": 0},
    "H2": {"C": 0, "H": 2, "O": 0, "X": 0},
    "H2O": {"C": 0, "H": 2, "O": 1, "X": 0},
    "X": {"C": 0, "H": 0, "O": 0, "X": 1},
}

references=list(references_compositions.keys())

# Create a matrix to hold the elemental compositions of the reference species
num_references = len(references_compositions)
N_A = np.zeros((num_references, num_elements))

# Fill in the elemental composition matrix of the reference species
for s, composition in references_compositions.items():
    i = references.index(s)
    N_A[i, :] = [composition["C"], composition["H"], composition["O"],␣
 ↪composition["X"]]

N_A
```

[5]:
```
array([[1., 4., 0., 0.],
       [0., 2., 0., 0.],
       [0., 2., 1., 0.],
       [0., 0., 0., 1.]])
```

We can now determine $\underline{\underline{\mathbf{M}}}$ from the elemental composition matrix of the target species $\underline{\underline{\mathbf{N}}}$ and the reference species $\underline{\underline{\mathbf{N}}}^{\mathrm{A}}$ via

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{\mathrm{A}^{-1}} \tag{6}$$

[10]:
```
#Calculate the matrix of stoichiometric coefficients to form the target from␣
 ↪the reference species
M=-N.dot(np.linalg.inv(N_A))
M
```

[10]:
```
array([[-2. ,  1. , -0. , -0. ],
       [-2. ,  1. , -0. , -1. ],
       [-0. , -0.5, -0. , -1. ],
       [-2. ,  1.5, -0. , -1. ],
       [-0. ,  1. , -1. , -1. ],
       [-2. ,  2. , -0. , -1. ],
       [-1. ,  3. , -1. , -1. ],
       [-1. ,  4. , -2. , -1. ],
```

```
[-1. ,  4. , -2. , -0. ],
[-0. ,  0.5, -1. , -1. ],
[-0. , -0. , -1. , -1. ],
[-1. ,  3. , -1. , -0. ],
[-2. ,  2. , -0. , -0. ]])
```

The last step is to compute the relative enthalpies using the matrix of stoichiometric coefficients of the formation reactions and the DFT energies of the target species and references

$$\underline{\mathbf{H}}_{\mathrm{A}} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^{\mathrm{A}} \tag{7}$$

[14]:
```
#Determine the enthalpy of formation of the target
H_A=(E+M.dot(E_A))*eV_to_kJmol

#Create a dictionary with the results
enthalpies_relative_to_anchors = {species[i]: H_A[i] for i in␣
 ↪range(len(species))}
enthalpies_relative_to_anchors
```

[14]:
```
{'C2H6': 56.6054039496493,
 'C2H6X': 38.12793788724724,
 'XH': -23.38583120360563,
 'XCH2CH3': 67.02306258857047,
 'XO': 106.99840022391058,
 'XCH2XCH2': 78.92717878597584,
 'XCO': 25.349752016584098,
 'CO2X': 76.47368874304667,
 'CO2': 82.41098970007852,
 'XOH': 90.39546452235372,
 'H2OX': -18.283106671951536,
 'CO': 161.88657282694535,
 'C2H4': 168.32529034728336}
```

The relative enthalpies of our chosen reference species is zero because it is a null reaction to form the reference. Pt(111) is denoted as a generic X.

[18]:
```
#Append the enthalpies of formation of the reference species to the enthalpies␣
 ↪of formation dictionary
enthalpies_relative_to_anchors.update({'CH4': 0, 'H2': 0, 'H2O':0, 'X':0})
enthalpies_relative_to_anchors
```

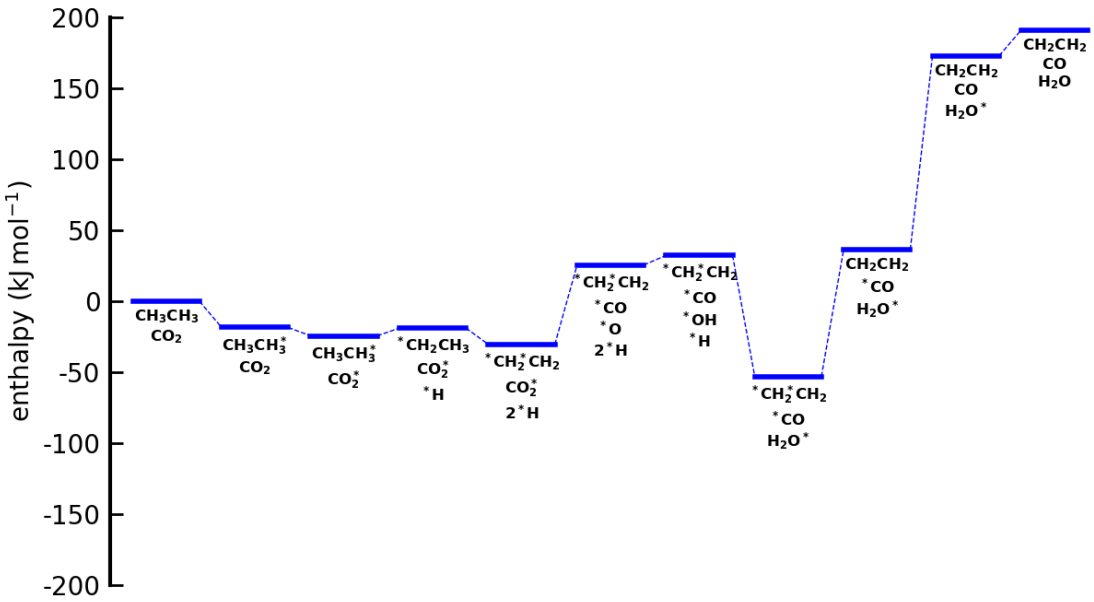
[18]:
```
{'C2H6': 56.6054039496493,
 'C2H6X': 38.12793788724724,
 'XH': -23.38583120360563,
 'XCH2CH3': 67.02306258857047,
 'XO': 106.99840022391058,
 'XCH2XCH2': 78.92717878597584,
```

```
 'XCO': 25.349752016584098,
 'CO2X': 76.47368874304667,
 'CO2': 82.41098970007852,
 'XOH': 90.39546452235372,
 'H2OX': -18.283106671951536,
 'CO': 161.88657282694535,
 'C2H4': 168.32529034728336,
 'CH4': 0,
 'H2': 0,
 'H2O': 0,
 'X': 0}
```

[20]:
```
plt.rcParams['figure.figsize'] = (14, 8)
plt.rcParams['axes.linewidth'] = 3
plt.rc('xtick', labelsize=20)
plt.rc('ytick', labelsize=20)
plt.rc('axes', labelsize=20)
plt.rc('legend', fontsize=20)
plt.rcParams['lines.markersize'] = 10
plt.rcParams['lines.linewidth'] = 4
plt.rcParams['xtick.direction'] = 'in'
plt.rcParams['ytick.direction'] = 'in'
plt.rcParams['xtick.major.size'] = 10
plt.rcParams['xtick.major.width'] = 2
plt.rcParams['ytick.major.size'] = 10
plt.rcParams['ytick.major.width'] = 2
plt.rcParams['legend.edgecolor'] = 'k'
plt.rcParams['axes.unicode_minus'] = False
plt.rcParams["legend.framealpha"] = 1
plt.rcParams['xtick.major.pad'] = 8
plt.rcParams['ytick.major.pad'] = 8
plt.rcParams['legend.handletextpad'] = 0.2
plt.rcParams['legend.columnspacing'] = 0.1
plt.rcParams['legend.labelspacing'] = 0.1
plt.rcParams['legend.title_fontsize'] = 14
plt.rcParams['axes.formatter.limits'] = (-3, 6)


gs = gridspec.GridSpec(nrows=1, ncols=1)
gs.update(wspace=0.5, hspace=0.5)
ax0 = plt.subplot(gs[0, 0])

ax0.set_ylim([-200,200])
ax0.set_xlim([-1,44])
ax0.get_xaxis().set_visible(False)
ax0.spines['right'].set_visible(False)
ax0.spines['top'].set_visible(False)
```

```
ax0.spines['bottom'].set_visible(False)
ax0.set_ylabel('$\mathrm{enthalpy\ (kJ\,mol^{-1})}$')


mechanism=({'C2H6':1,'CO2':1},
    {'C2H6X':1,'CO2':1},
    {'C2H6X':1,'CO2X':1},
    {'XCH2CH3':1,'CO2X':1,'XH':1},
    {'XCH2XCH2':1,'CO2X':1,'XH':2},
    {'XCH2XCH2':1,'XCO':1,'XO':1,'XH':2},
    {'XCH2XCH2':1,'XCO':1,'XOH':1,'XH':1},
    {'XCH2XCH2':1,'XCO':1,'H2OX':1},
    {'C2H4':1,'XCO':1,'H2OX':1},
    {'C2H4':1,'CO':1,'H2OX':1},
    {'C2H4':1,'CO':1,'H2O':1},
    )

tags=('$\mathbf{CH_3CH_3}$\n$\mathbf{CO_2}$',
      '$\mathbf{CH_3CH_3^*}$\n$\mathbf{CO_2}$',
      '$\mathbf{CH_3CH_3^*}$\n$\mathbf{CO_2^*}$',
      '$\mathbf{^*CH_2CH_3}$\n$\mathbf{CO_2^*}$\n$\mathbf{^*H}$',
      '$\mathbf{^*CH_2^*CH_2}$\n$\mathbf{CO_2^*}$\n$\mathbf{2 ^*H}$',
      '$\mathbf{^*CH_2^*CH_2}$\n$\mathbf{^*CO}$\n$\mathbf{^*O}$\n$\mathbf{2␣
 ↪^*H}$',
     ␣
 ↪'$\mathbf{^*CH_2^*CH_2}$\n$\mathbf{^*CO}$\n$\mathbf{^*OH}$\n$\mathbf{^*H}$',
      '$\mathbf{^*CH_2^*CH_2}$\n$\mathbf{^*CO}$\n$\mathbf{H_2O^*}$',
      '$\mathbf{CH_2CH_2}$\n$\mathbf{^*CO}$\n$\mathbf{H_2O^*}$',
      '$\mathbf{CH_2CH_2}$\n$\mathbf{CO}$\n$\mathbf{H_2O^*}$',
      '$\mathbf{CH_2CH_2}$\n$\mathbf{CO}$\n$\mathbf{H_2O}$',
)

# Function to compute the sum for each step in the mechanism
def compute_sum(vec,dictionary):
    return sum(vec[key] * value for key, value in dictionary.items())

def ediagram(ref_system):
    system = [compute_sum(ref_system,entry) for entry in mechanism]
    rel_enthalpies=np.zeros(len(system))
    for i, enthalpies in enumerate(system):
        start = i * 4
        end = start + 3
        ax0.plot((start, end), (enthalpies-system[0], enthalpies-system[0]),␣
 ↪linestyle='solid', color='b')
        if i>0:
```

```
            ax0.
 ↪plot((start-1,start),(system[i-1]-system[0],enthalpies-system[0]),linestyle='dashed',color=
 ↪linewidth=1)
        rel_enthalpies[i]=enthalpies-system[0]
    return rel_enthalpies

values=ediagram(enthalpies_relative_to_anchors)


for i in range(len(np.array(values).T)):
    start = i * 4
    ax0.text(start+1.5,np.array(values).
 ↪T[i]-5,tags[i],va='top',ha='center',size=12)
```



[ ]: