Supporting Information: Unifying thermochemistry concepts in computational heterogeneous catalysis

Bjarne Kreitz,[*,†] Gabriel Gusmão,[‡] Dingqi Nai,[‡] Sushree Jagriti Sahoo,[‡] Andrew A. Peterson,[†] David H. Bross,[¶] C. Franklin Goldsmith,[*,†] and Andrew J. Medford[*,‡]

†*School of Engineering, Brown University, Providence, RI 02912, USA*

‡*School of Chemical and Biomolecular Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States*

¶*Chemical Sciences and Engineering Division, Argonne National Laboratory, Lemont, Illinois 60439, United States*

E-mail: bjarne_kreitz@brown.edu; franklin_goldsmith@brown.edu; ajm@gatech.edu

1 Jupyter notebooks

The main supplementary material for this manuscript is a set of Jupyter notebooks that apply the approaches/methods that are discussed in the main manuscript. These notebooks provide a step-by-step explanation of how every method works and how it can be implemented in Python. The Jupyter notebooks can be found on Zenodo in ref. S1 in or forked from `https://github.com/bjkreitz/adsorbate-thermochemistry`. For convenience, they are also added in PDF format to this document and can be found at the end.

2 Case study

Table S1 summarizes the DFT data obtained with the BEEF-vdW and RPBE exchange-correlation functional. The reader is referred to Ref. S2 for details on the electronic structure calculations.

Table S1: Zero-point corrected DFT energies of the adsorbates and gas-phase species for the case study.

Species	BEEF-vdW energy (eV)	RPBE energy (eV)
Pt(111)	-377616.072	-366518.205
C_2H_6	-615.302	-600.361
H_2O	-611.019	-600.739
CO_2	-1414.683	-1389.568
C_2H_4	-581.446	-567.332
CO	-835.539	-819.812
$C_2H_6^*$	-378231.566	-367118.506
$^*CH_2CH_3$	-378214.917	-367102.198
$^*O$	-378193.283	-367085.354
$^*CH_2^*CH_2$	-378198.444	-367086.181
$^*CO$	-378453.026	-367339.346
CO_2^*	-379030.816	-367907.753
$^*OH$	-378209.805	-367101.878
H_2O^*	-378227.280	-367118.949
$^*H$	-377632.664	-366534.422
H_2	-32.698	-31.794
CH_4	-324.294	-316.411

3 Deriving enthalpies of formation of adsorbates using gas-phase reference species

An alternative derivation of the EOF of the adsorbates referenced to the global thermochemical network was proposed by Blondal et al.[S3] We start with an adsorption reaction of our target species, P.

$$\mathrm{P} + {}^* \rightleftharpoons \mathrm{P}^*$$

where P is the gas-phase precursor $C_aO_bH_c$ and P^* is the adsorbate $C_aO_bH_c^*$. The EOF of the adsorbate can be calculated if the enthalpy of the adsorption reaction and the EOF of the gas-phase precursor are known. The enthalpy of reaction can be easily computed using the DFT energies via

$$\Delta_r H^{\mathrm{QM}} = E_{\mathrm{P}^*} - E_{\mathrm{P}} - E_{\mathrm{Pt(111)}} \tag{1}$$

The enthalpy of reaction can also be computed from the EOFs of the individual species.

$$\Delta_\mathrm{r}H = \Delta_\mathrm{f}H_{\mathrm{P}^*} - \Delta_\mathrm{f}H_\mathrm{P} - \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} \tag{2}$$

Assuming that $\Delta_\mathrm{r}H = \Delta_\mathrm{r}H^\mathrm{QM}$, we can rearrange the equations to solve for the EOF of our target P^*, resulting in

$$\Delta_\mathrm{f}H_{\mathrm{P}^*} = \Delta_\mathrm{f}H_\mathrm{P} + \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} + \Delta_\mathrm{r}H^\mathrm{QM} \tag{3}$$

The EOF of the vacant Pt(111) surface is $\Delta_\mathrm{f}H_{\mathrm{Pt(111)}} = 0\,\mathrm{kJ\,mol^{-1}}$ by assertion as discussed in Section 4 of the manuscript. Thus, the only unknown to solve for the EOF of the adsorbate is the EOF of the gas-phase precursor. The EOF of the gas-phase precursor also provides the reference to an existing thermochemical network. Unfortunately, many gas-phase precursors are unstable and, therefore, not tabulated in the ATcT. Thus, it is necessary to rely on DFT energies and use these energies to estimate EOFs relative to gas-phase reference with EOFs tabulated in ATcT. A common practice in the gas-phase community is to identify reference species using an *isogyric* reaction[S4,S5] where every $\mathrm{C_aO_bH_c}$ molecule can be decomposed into an atomic reference basis set of $\mathrm{CH_4}$, $\mathrm{H_2O}$, and $\mathrm{H_2}$.

$$\nu_{\mathrm{CH_4}}\,\mathrm{CH_4} + \nu_{\mathrm{H_2O}}\,\mathrm{H_2O} + \nu_{\mathrm{H_2}}\mathrm{H_2} \longrightarrow \mathrm{C_aO_bH_c}$$

This method is similar to the ANL-0 approach[S5] and the method described in Section 2.1 of the manuscript, using $\mathrm{CH_4}$, $\mathrm{H_2O}$, and $\mathrm{H_2}$ as reference species. To determine the EOF of the precursor, we compute the enthalpy of reaction from the DFT energies of the gas-phase species:

$$\Delta_\mathrm{r}H^\mathrm{QM} = E_\mathrm{P} + \sum_{i\neq\mathrm{P}}^{N} \nu_i E_i \tag{4}$$

We can then calculate the EOF of the precursor with the known EOFs of the reference species from the ATcT database at 0 K assuming that $\Delta_\mathrm{r}H^\mathrm{QM} = \Delta_\mathrm{r}H$

$$\Delta_\mathrm{f}H_\mathrm{P} = \Delta_\mathrm{r}H^\mathrm{QM} - \sum_{i\neq\mathrm{P}}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{5}$$

The EOF of the gas-phase precursor can now be inserted into Equation (3) to compute the EOF of the adsorbate, leading to

$$\Delta_\mathrm{f}H_{\mathrm{P}^*} = \Delta_\mathrm{f}H_\mathrm{P} + \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} + \Delta_\mathrm{r}H^\mathrm{QM} \tag{6}$$

$$\Delta_\mathrm{f}H_{\mathrm{P}^*} = E_{\mathrm{P}^*} - E_{\mathrm{Pt(111)}} - E_\mathrm{P} + E_\mathrm{P} + \sum_{i\neq\mathrm{P}}^{N} \nu_i E_i - \sum_{i\neq\mathrm{P}}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{7}$$

$$\Delta_\mathrm{f}H_{\mathrm{P}^*} = E_{\mathrm{P}^*} - E_{\mathrm{Pt(111)}} + \sum_{i\neq\mathrm{P}}^{N} \nu_i E_i - \sum_{i\neq\mathrm{P}}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{8}$$

Through the detailed derivations, it can be seen that the DFT energies of the precursor

cancel out. Including the surface slab into the reactants results again in Equation (11).

$$\underbrace{\Delta_\mathrm{f}H_{\mathrm{P}^*}}_{\text{target}} = \underbrace{E_{\mathrm{P}^*}}_{\text{target}} + \underbrace{\sum_{i\neq\mathrm{P}}^{N} \nu_i E_i}_{\text{references}} - \underbrace{\sum_{i\neq\mathrm{P}}^{N} \nu_i \Delta_\mathrm{f}H_i}_{\text{references}} \tag{9}$$

Or in linear algebra notation

$$\underline{\mathbf{H}}_\mathrm{f} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^\mathrm{R} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_\mathrm{f}^\mathrm{R} \tag{10}$$

$$\underline{\mathbf{H}}_\mathrm{f} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\left(\underline{\mathbf{E}}^\mathrm{R} - \underline{\mathbf{H}}_\mathrm{f}^\mathrm{R}\right) \tag{11}$$

The approach adjusts all adsorption/desorption reaction enthalpies except for the reference species. Adsorption enthalpies of the three gas-phase reference species match the DFT values. To illustrate this, we derive the reaction enthalpy for CH_4.

$$\mathrm{CH_4} + {}^* \rightleftharpoons \mathrm{CH_4^*} \tag{12}$$

and the adsorption enthalpy is determined via

$$\Delta_\mathrm{r}H_\mathrm{ads} = \Delta_\mathrm{f}H_{\mathrm{CH_4^*}} - \Delta_\mathrm{f}H_{\mathrm{CH_4}} - \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} \tag{13}$$

The EOF of CH_4 is taken from the ATcT and the EOF of Pt(111) is 0 since it is assumed to represent bulk Pt. The EOF of adsorbed CH_4^* is calculated via

$$\Delta_\mathrm{f}H_{\mathrm{P}^*} = E_{\mathrm{P}^*} + \sum_{i\neq\mathrm{P}}^{N} \nu_i E_i - \sum_{i\neq\mathrm{P}}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{14}$$

$$\Delta_\mathrm{f}H_{\mathrm{CH_4^*}} = E_{\mathrm{CH_4^*}} - E_{\mathrm{Pt(111)}} - E_{\mathrm{CH_4}} + \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} + \Delta_\mathrm{f}H_{\mathrm{CH_4}} \tag{15}$$

Replacing now the EOF of CH_4^* in Equation (13) with Equation (15) leads to

$$\Delta_\mathrm{r}H_\mathrm{ads} = E_{\mathrm{CH_4^*}} - E_{\mathrm{Pt(111)}} - E_{\mathrm{CH_4}} + \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} + \Delta_\mathrm{f}H_{\mathrm{CH_4}} - \Delta_\mathrm{f}H_{\mathrm{CH_4}} - \Delta_\mathrm{f}H_{\mathrm{Pt(111)}} \tag{16}$$

$$\Delta_\mathrm{r}H_\mathrm{ads} = E_{\mathrm{CH_4^*}} - E_{\mathrm{Pt(111)}} - E_{\mathrm{CH_4}} \tag{17}$$

The EOF of the slab and gas-phase CH_4 cancel out. Thus, the adsorption enthalpy of CH_4 is equal to the DFT calculated adsorption enthalpy. The same is true for the other chosen reference species.

4 Isodesmic reactions

Here we show the matrix with the fragments of the reference species $\underline{\underline{\mathbf{F}}}^{\mathrm{R}}$ that are needed for isodesmic reactions and the manual construction of the matrix of stoichiometric coefficients.

$$\underline{\underline{\mathbf{F}}}^{\mathrm{R}} = \begin{array}{c} \begin{array}{cccccccccc} \text{C}-\text{O} & \text{C}=\text{C} & \text{C}=\text{O} & \text{C}-\text{C} & \text{C}-\text{H} & \text{O}-\text{H} & \text{Pt}-\text{C} & \text{Pt}\#\text{C} & \text{Pt}=\text{C} & \text{Pt}-\text{O} \end{array} \\ \left[\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \end{array} \begin{array}{l} \text{CH}_3\text{OH}^* \\ \text{CH}_2\text{CH}_2^* \\ \text{H}_2\text{CO}^* \\ \text{CH}_3\text{CH}_3^* \\ \text{CH}_4^* \\ \text{H}_2\text{O}^* \\ {}^*\text{CH}_3 \\ {}^*\text{CH} \\ {}^*\text{CH}_2 \\ {}^*\text{OH} \end{array} \tag{18}$$

$$\underline{\underline{\mathbf{F}}} = \begin{array}{c} \begin{array}{cccccccccc} \text{C}-\text{O} & \text{C}=\text{C} & \text{C}=\text{O} & \text{C}-\text{C} & \text{C}-\text{H} & \text{O}-\text{H} & \text{Pt}-\text{C} & \text{Pt}\#\text{C} & \text{Pt}=\text{C} & \text{Pt}-\text{O} \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 0 & 0 & 1 & 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{l} {}^*\text{CH}_2\text{CH}_3 \\ {}^*\text{CH}_2^*\text{CH}_2 \\ \text{CO}_2^* \end{array} \tag{19}$$

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{F}}}\,\underline{\underline{\mathbf{F}}}^{\mathrm{R}\,-1} = \tag{20}$$

$$\begin{array}{c} \begin{array}{cccccccccc} \text{CH}_3\text{OH}^* & \text{CH}_2\text{CH}_2^* & \text{H}_2\text{CO}^* & \text{CH}_3\text{CH}_3^* & \text{CH}_4^* & \text{H}_2\text{O}^* & {}^*\text{CH}_3 & {}^*\text{CH} & {}^*\text{CH}_2 & {}^*\text{OH} \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{l} {}^*\text{CH}_2\text{CH}_3 \\ {}^*\text{CH}_2^*\text{CH}_2 \\ \text{CO}_2^* \end{array} \tag{21}$$

5 Converting between reaction enthalpies and formation enthalpies

The reaction mechanism for the oxidative dehydrogenation of C_2H_6 with CO_2 that is assumed in the manuscript is

$$
\begin{aligned}
C_2H_6 + {}^* &\rightleftharpoons C_2H_6^* \\
CO_2 + {}^* &\rightleftharpoons CO_2^* \\
C_2H_6^* + {}^* &\rightleftharpoons {}^*CH_2CH_3 + {}^*H \\
{}^*CH_2CH_3 + 2\,{}^* &\rightleftharpoons {}^*CH_2^*CH_2 + {}^*H \\
CO_2^* + {}^* &\rightleftharpoons {}^*CO + {}^*O \\
{}^*H + {}^*O &\rightleftharpoons {}^*OH + {}^* \\
{}^*OH + {}^*H &\rightleftharpoons H_2O^* + {}^* \\
H_2O^* &\rightleftharpoons H_2O + {}^* \\
{}^*CO &\rightleftharpoons CO + {}^* \\
{}^*CH_2^*CH_2 &\rightleftharpoons C_2H_4 + 2\,{}^*
\end{aligned}
$$

The following stoichiometry matrix $\underline{\underline{\mathbf{M}}}$ can be constructed from this mechanism.

$$
\underline{\underline{\mathbf{M}}} = \begin{array}{c} \begin{matrix} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} \end{matrix} \\ \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-1 & -1 & -1 & -2 & -1 & 1 & 1 & 1 & 1 & 2
\end{bmatrix} \end{array} \begin{array}{l} C_2H_6 \\ C_2H_6^* \\ CO_2 \\ CO_2^* \\ {}^*CH_2CH_3 \\ {}^*H \\ {}^*CH_2^*CH_2 \\ {}^*CO \\ {}^*O \\ {}^*OH \\ H_2O^* \\ H_2O \\ CO \\ C_2H_4 \\ {}^* \end{array} \tag{22}
$$

$\underline{\mathbf{E}}$ is the DFT energy vector of all species in the mechanism.

$$\underline{\mathbf{E}} = \begin{bmatrix} -615.302\,\mathrm{eV} \\ -378\,231.566\,\mathrm{eV} \\ -1414.683\,\mathrm{eV} \\ -379\,030.816\,\mathrm{eV} \\ -378\,214.917\,\mathrm{eV} \\ -377\,632.664\,\mathrm{eV} \\ -378\,198.444\,\mathrm{eV} \\ -378\,453.026\,\mathrm{eV} \\ -378\,193.283\,\mathrm{eV} \\ -378\,209.805\,\mathrm{eV} \\ -378\,227.280\,\mathrm{eV} \\ -611.019\,\mathrm{eV} \\ -835.539\,\mathrm{eV} \\ -581.446\,\mathrm{eV} \\ -377\,616.072\,\mathrm{eV} \end{bmatrix} \begin{matrix} C_2H_6 \\ C_2H_6^* \\ CO_2 \\ CO_2^* \\ {}^*CH_2CH_3 \\ {}^*H \\ {}^*CH_2^*CH_2 \\ {}^*CO \\ {}^*O \\ {}^*OH \\ H_2O^* \\ H_2O \\ CO \\ C_2H_4 \\ {}^* \end{matrix} \tag{23}$$

The vector of reaction enthalpies $\underline{\mathbf{H}}_\mathrm{r}$ can be calculated via

$$\underline{\mathbf{H}}_\mathrm{r} = \underline{\underline{\mathbf{M}}}^\intercal \underline{\mathbf{E}} = \begin{bmatrix} -0.192\,\mathrm{eV} \\ -0.062\,\mathrm{eV} \\ 0.057\,\mathrm{eV} \\ -0.119\,\mathrm{eV} \\ 0.579\,\mathrm{eV} \\ 0.070\,\mathrm{eV} \\ -0.884\,\mathrm{eV} \\ 0.189\,\mathrm{eV} \\ 1.415\,\mathrm{eV} \\ 0.926\,\mathrm{eV} \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \\ r_8 \\ r_9 \\ r_{10} \end{matrix} \tag{24}$$

The stoichiometry matrix $\underline{\underline{\mathbf{M}}}$ has to be separated into the anchor species $\underline{\underline{\mathbf{M}}}_\mathrm{A}$ and non-anchor species $\hat{\underline{\underline{\mathbf{M}}}}$, leading to Equation (25).

$$\underline{\mathbf{H}}_\mathrm{r} = \underline{\underline{\mathbf{M}}}^\intercal \, \underline{\mathbf{H}}_\mathrm{A} = \begin{bmatrix} \hat{\underline{\underline{\mathbf{M}}}} \\ \\ \underline{\underline{\mathbf{M}}}_\mathrm{A} \end{bmatrix}^\intercal \begin{bmatrix} \hat{\underline{\mathbf{H}}}_\mathrm{A} \\ \\ \tilde{\underline{\mathbf{H}}}_\mathrm{A} \end{bmatrix} = \hat{\underline{\underline{\mathbf{M}}}}^\intercal \, \hat{\underline{\mathbf{H}}}_\mathrm{A} + \underline{\underline{\mathbf{M}}}_\mathrm{A}^\intercal \, \tilde{\underline{\mathbf{H}}}_\mathrm{A} \tag{25}$$

When choosing CO, H_2O, C_2H_6, Pt(111) as reference species, this separation results in:

$$\underline{\underline{\hat{\mathbf{M}}}} = \begin{array}{c} \begin{array}{cccccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} \end{array} \\ \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{array}{l} \\ \mathrm{C_2H_6^*} \\ \mathrm{CO_2} \\ \mathrm{CO_2^*} \\ {}^*\mathrm{CH_2CH_3} \\ {}^*\mathrm{H} \\ {}^*\mathrm{CH_2^*CH_2} \\ {}^*\mathrm{CO} \\ {}^*\mathrm{O} \\ {}^*\mathrm{OH} \\ \mathrm{H_2O^*} \\ \mathrm{C_2H_4} \end{array} \tag{26}$$

$$\underline{\underline{\mathbf{M}}}_{\mathrm{A}} = \begin{array}{c} \begin{array}{cccccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} \end{array} \\ \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & -1 & -1 & -2 & -1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix} \end{array} \begin{array}{l} \\ \mathrm{C_2H_6} \\ \mathrm{CO} \\ \mathrm{H_2O} \\ {}^* \end{array} \tag{27}$$

Finally, the enthalpies of formation of the target species can be calculated using the Moore-Penrose pseudo inverse of the stoichiometry matrix $\underline{\underline{\mathbf{M}}}^{+}$.

$$\underline{\hat{\mathbf{H}}}_{\mathrm{A}} = \left(\underline{\underline{\hat{\mathbf{M}}}}^{\intercal}\right)^{+} \left(\underline{\underline{\mathbf{M}}}^{\intercal}\,\underline{\mathbf{E}} - \underline{\underline{\mathbf{M}}}_{\mathrm{A}}^{\intercal}\,\underline{\tilde{\mathbf{H}}}_{\mathrm{A}}\right) \tag{28}$$

For this purpose, the singular-value decomposition of $\underline{\underline{\mathbf{M}}}$ is determined by Equation (29).

$$\underline{\underline{\mathbf{M}}} = \underline{\underline{\mathbf{U}}}\,\underline{\underline{\Sigma}}\,\underline{\underline{\mathbf{V}}}^{\dagger} \tag{29}$$

Where $\underline{\underline{\mathbf{U}}} \in \mathbb{C}^{n\times n}$, $\underline{\underline{\mathbf{V}}} \in \mathbb{C}^{m\times m}$ are orthonormal basis of $\mathbb{C}^n$ and $\mathbb{C}^m$, respectively, and $\underline{\underline{\Sigma}} \in \mathbb{C}^{n\times m}$ is a matrix with singular values of $\underline{\underline{\mathbf{M}}}$ along its diagonal entries and zeroes for the off-diagonal ones. The pseudo inverse $\underline{\underline{\mathbf{M}}}^{+}$ is computed as in Equation (30).

$$\underline{\underline{\mathbf{M}}}^{+} = \underline{\underline{\mathbf{V}}}\,\underline{\underline{\Sigma}}^{+}\,\underline{\underline{\mathbf{U}}}^{\dagger} \tag{30}$$

In which $\underline{\underline{\Sigma}}^{+}$ consists of a diagonal matrix with the reciprocal singular values of $\underline{\underline{\mathbf{M}}}$ and $\underline{\underline{\mathbf{U}}}$ is, in this case, made dimensionally-consistent, by truncating the number of rows to the dimensions of $\underline{\underline{\mathbf{V}}}$.

For continuous vector spaces defined by non-disjoint reaction networks, and when the defined reference species together contain all chemical elements in the system, $\underline{\underline{\hat{\mathbf{M}}}}^{\intercal}$ is full-rank and,

therefore,

$$\left(\underline{\underline{\mathbf{M}}}^{\intercal}\right)^{+} = \left(\underline{\underline{\mathbf{M}}}^{\intercal}\right)^{-1}. \tag{31}$$

References

(S1) Kreitz, B.; Gusmão, G. S.; Nai, D.; Sahoo, S. J.; Peterson, A. A.; Bross, D. H.; Goldsmith, C. F.; Medford, A. J. Data for Unifying thermochemistry concepts in computational heterogeneous catalysis. 2024; `https://doi.org/10.5281/zenodo.13143954`.

(S2) Kreitz, B.; Abeywardane, K.; Goldsmith, C. F. Linking Experimental and *Ab Initio* Thermochemistry of Adsorbates with a Generalized Thermochemical Hierarchy. *J. Chem. Theory Comput.* **2023**, *19*, 4149–4162.

(S3) Blöndal, K.; Jelic, J.; Mazeau, E.; Studt, F.; West, R. H.; Goldsmith, C. F. Computer-Generated Kinetics for Coupled Heterogeneous/Homogeneous Systems: A Case Study in Catalytic Combustion of Methane on Platinum. *Ind. Eng. Chem. Res.* **2019**, *58*, 17682–17691.

(S4) Wheeler, S. E.; Houk, K. N.; Schleyer, P. v. R.; Allen, W. D. A Hierarchy of Homodesmotic Reactions for Thermochemistry. *J. Am. Chem. Soc.* **2009**, *131*, 2547–2560.

(S5) Klippenstein, S. J.; Harding, L. B.; Ruscic, B. Ab Initio Computations and Active Thermochemical Tables Hand in Hand: Heats of Formation of Core Combustion Species. *J. Phys. Chem. A* **2017**, *121*, 6580–6602.

Referencing_QM_energies_to_elemental_chemical_potentials

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)

[18]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

eV_to_kJmol=96.485
```

1 Reference QM energies to elemental chemical potentials

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.

$$\nu_{\mathrm{C}}C + \nu_{\mathrm{H}}H + \nu_{\mathrm{O}}O + \nu_{*}* \rightarrow P \tag{1}$$

The relative enthalpy of the target can be determined from the DFT energies using the chemical potentials via

$$\Delta_{\mathrm{A}}H_{\mathrm{P}}(0K) = E_{\mathrm{P}} - \sum_{k}^{N_{elements}} \nu_k \mu_k^{\mathrm{A}} \tag{2}$$

In matrix notation this reads as

$$\underline{\mathbf{H}}_{\mathrm{A}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{N}}}^{\mathrm{A}} \tag{3}$$

where $\underline{\underline{\mathbf{N}}}$ is the elemental composition matrix of the target species.

Theoretically, it is possible to just use the DFT energies of the bare elements as the chemical potentials. However, it is impossible to calculate these accurately with DFT. Instead, we use calculate the elemental chemical potential from closed-shell reference species such as $[CH_4, H_2, H_2O]$ and in the case of Pt(111), the empty slab has to be added as a reference species. The chemical potentials can be derived via

$$\mu_{\mathrm{C}} = E_{CH_4} - 2E_{H_2} \tag{4}$$
$$\mu_{\mathrm{O}} = E_{H_2O} - E_{H_2} \tag{5}$$
$$\mu_{\mathrm{H}} = 0.5E_{H_2} \tag{6}$$
$$\mu_{*} = E_{Pt(111)} \tag{7}$$

It is also possible to use linear algebra to determine the elemental chemical potentials with the inverse of the elemental composition matrix of the reference species.

$$\_ = \underline{\underline{\mathbf{N}}}^{\mathrm{A}^{-1}}\ \underline{\mathbf{E}}^{\mathrm{A}} \tag{8}$$

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.

[19]:
```
#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,
    "CH4": -324.2935569,
    "H2": -32.69844421,
    "H2O": -611.0186083,
    "X": -377616.072,
}

#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

[20]:
```
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.

[21]:
```
# 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},
    "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},
}

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
```

[21]:
```
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.],
       [1., 4., 0., 0.],
       [0., 2., 0., 0.],
       [0., 2., 1., 0.],
       [0., 0., 0., 1.]])
```

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

[22]:
```
# 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
```

[22]:
```
array([[1., 4., 0., 0.],
       [0., 2., 0., 0.],
       [0., 2., 1., 0.],
       [0., 0., 0., 1.]])
```

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

$$\underline{\_}^{\mathrm{A}} = \underline{\underline{\mathbf{N}}}^{\mathrm{A}^{-1}}\,\underline{\mathbf{E}}^{\mathrm{A}} \tag{9}$$

[23]:
```
#Calculate the matrix of stoichiometric coefficients to form the target from␣
 ↪the reference species
mu=(np.linalg.inv(N_A)).dot(E_A)
elements=['C','H','O','X']

#Create a dictionary with the results
chemical_potentials = {elements[i]: mu[i] for i in range(len(elements))}
chemical_potentials
```

[23]:
```
{'C': -258.89666848,
 'H': -16.349222105,
 'O': -578.3201640899999,
 'X': -377616.072}
```

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

$$\underline{\mathbf{H}}_{\mathrm{A}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{N}}}\,\underline{\_}^{\mathrm{A}} \tag{10}$$

[24]:
```
#Determine the enthalpy of formation of the target
H_A=(E-N.dot(mu))*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
```

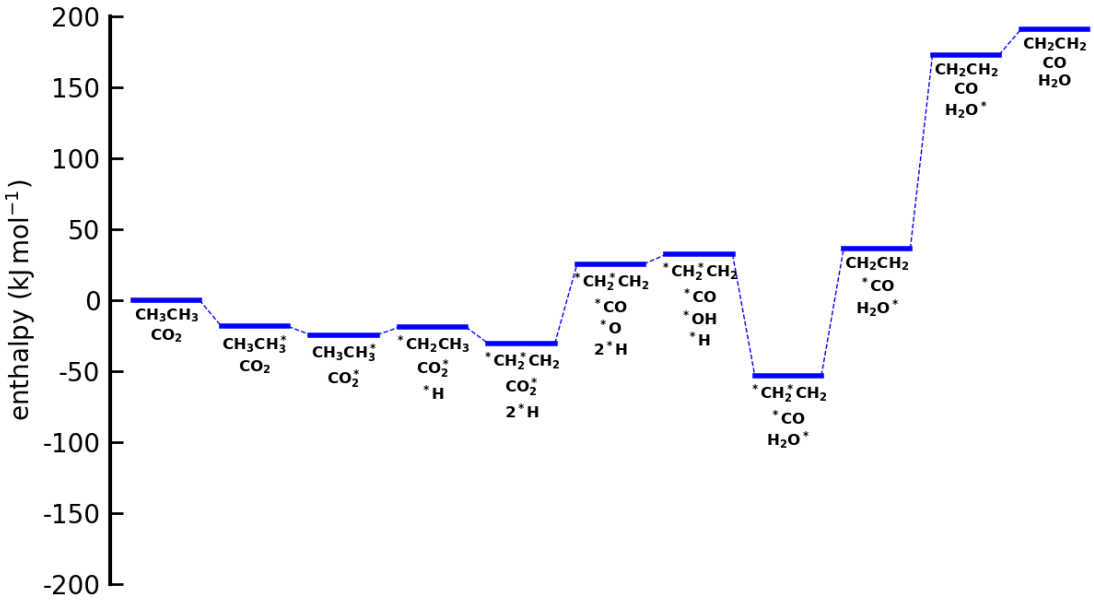
[24]:
```
{'C2H6': 56.60540394966027,
 'C2H6X': 38.12793788724724,
 'XH': -23.38583120360563,
 'XCH2CH3': 67.02306258857047,
 'XO': 106.99840022391058,
 'XCH2XCH2': 78.92717878597584,
 'XCO': 25.349752010967933,
 'CO2X': 76.47368873743049,
```

```
 'CO2': 82.41098970007852,
 'XOH': 90.39546452235372,
 'H2OX': -18.283106671951536,
 'CO': 161.88657282694535,
 'C2H4': 168.32529034728336,
 'CH4': 0.0,
 'H2': 0.0,
 'H2O': 0.0,
 'X': 0.0}
```

[25]:
```
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)
```



[ ]:

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
```

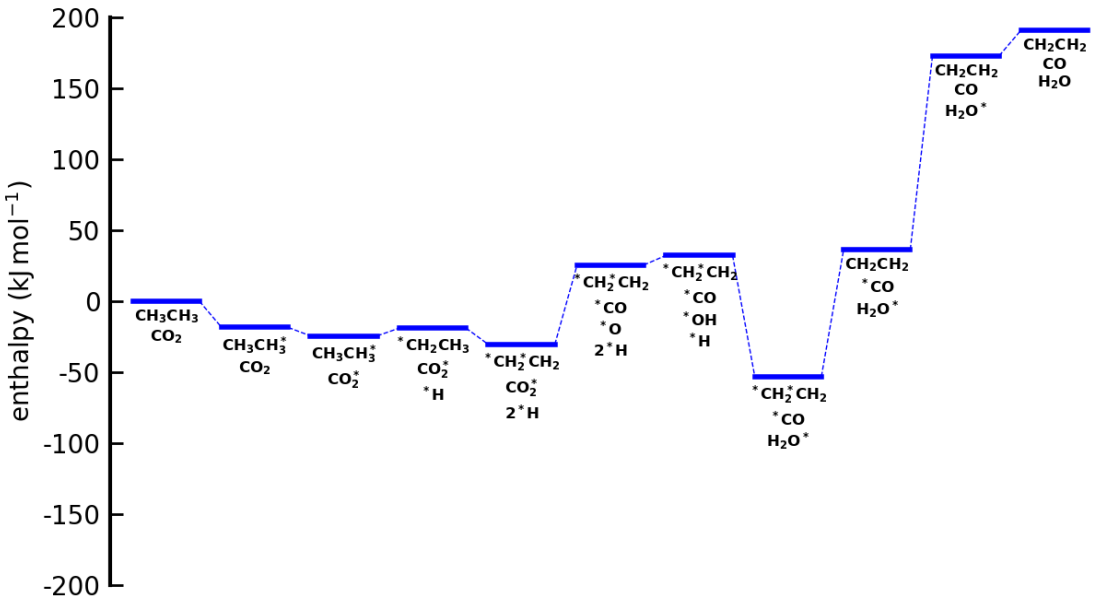
[18]:
```
{'C2H6': 56.6054039496493,
 'C2H6X': 38.12793788724724,
 'XH': -23.38583120360563,
 'XCH2CH3': 67.02306258857047,
 'XO': 106.99840022391058,
 'XCH2XCH2': 78.92717878597584,
```

```
 '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)
```



[ ]:

Referencing_QM_data_to_least_sqaures_chemical_potentials

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)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from sklearn.linear_model import LinearRegression
from scipy import linalg

eV_to_kJmol=96.485
```

1 Referencing QM energies to least-squares elemental chemical potentials

For a typical DFT dataset, there is a myriad of possible anchor species to calculate the chemical potentials. As shown in the manuscript, different anchor species lead to different elemental chemical potentials and, thus, to different relative enthalpies of the adsorbates. Instead of manually selecting the anchor species to determine the chemical potentials of the elements in a sequential approach, it is possible to exploit the invariance and select convenient numerical values for the chemical potentials. In particular, it is often convenient to select chemical potentials that minimize the (sum of squared) errors between enthalpies of formation computed using different codes, levels of fidelity, or reference sets. This can be achieved using linear algebra and least-squares regression and avoids the need to specify any explicit chemical species as anchors.

To calculate the least-squares anchor elemental potentials, we take the elemental composition matrix $\underline{\underline{\mathbf{N}}}$ that contains all the formation reactions of the target species from the constituent elements to derive the chemical potentials. Rather than selecting a set of explicit anchor species with defined stoichiometries, we seek the set of chemical potentials that minimizes the squared magnitude of the resulting relative enthalpies. The linear algebra formulation offers a convenient way to perform the minimization of squared relative enthalpy magnitudes, which is closely related to the least-squares regression problem and is given via

$$_{\text{-LS}} = \left(\underline{\underline{\mathbf{N}}}^{\top}\underline{\underline{\mathbf{N}}}\right)^{-1}\underline{\underline{\mathbf{N}}}^{\top}\underline{\mathbf{E}} = \underline{\underline{\mathbf{N}}}^{+}\underline{\mathbf{E}} \tag{1}$$

The least-squares regression result can also be written as the product of the pseudo-inverse of the elemental composition matrix $\underline{\underline{\mathbf{N}}}^{+}$ and the energy vector $\underline{\mathbf{E}}$. Calculation of the enthalpies relative to the least-squares anchors is then performed by replacing the chemical potentials in the equation above, leading to

$$\underline{\mathbf{H}}_{\text{LS}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{N}}}_{\text{-}}^{\text{LS}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{+}\,\underline{\mathbf{E}} \tag{2}$$

where we use the LS symbol to denote a least-squares anchor set that does not explicitly depend on specific molecular anchor species, but rather implicitly depends on the entire set of target species included (i.e. the species in the rows of $\underline{\underline{\mathbf{N}}}$).

We first define the vector of DFT reaction energies $\underline{\mathbf{E}}$

[2]:
```
#Target species for which we need to compute the EOFs
species_energies = {
    "X": -377616.072,
    "C2H6": -615.3019939,
    "H2O": -611.0186083,
    "CO2": -1414.682864,
    "C2H6X" : -378231.5655,
    "XCH2CH3": -378214.9168,
    "XO": -378193.2832,
    "XCH2XCH2": -378198.4442,
    "XCO": -378453.0261,
    "CO2X": -379030.8164,
    "XOH": -378209.8045,
    "H2OX": -378227.2801,
    "XH": -377632.6636,
    "CO": -835.5389907,
    "C2H4":-581.44565067567,
}

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

We then construct the elemental composition matrix $\underline{\underline{\mathbf{N}}}$

[14]:
```
# Define the species and their elemental compositions in a dictionary
species_compositions = {
    "X": {"C": 0, "H": 0, "O": 0, "X": 1},
    "C2H6": {"C": 2, "H": 6, "O": 0, "X": 0},
    "H2O": {"C": 0, "H": 2, "O": 1, "X": 0},
    "CO2": {"C": 1, "H": 0, "O": 2, "X": 0},
    "C2H6X": {"C": 2, "H": 6, "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},
    "XOH": {"C": 0, "H": 1, "O": 1, "X": 1},
    "H2OX": {"C": 0, "H": 2, "O": 1, "X": 1},
    "XH": {"C": 0, "H": 1, "O": 0, "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
```

[14]:
```
array([[0., 0., 0., 1.],
       [2., 6., 0., 0.],
       [0., 2., 1., 0.],
       [1., 0., 2., 0.],
       [2., 6., 0., 1.],
       [2., 5., 0., 1.],
       [0., 0., 1., 1.],
       [2., 4., 0., 1.],
       [1., 0., 1., 1.],
       [1., 0., 2., 1.],
       [0., 1., 1., 1.],
       [0., 2., 1., 1.],
       [0., 1., 0., 1.],
       [1., 0., 1., 0.],
       [2., 4., 0., 0.]])
```

The chemical potential that minimizes the sum of squared differences of the resulting relative enthalpies is computed using the pseudo-inverse

$$\underline{\mu}^{\mathrm{LS}} = \underline{\underline{\mathbf{N}}}^{+}\,\underline{\mathbf{E}} \tag{3}$$

[15]:
```
#The chemical potentials are calculated with the pseudo-inverse of the␣
 ↪elemental composition matrix.
```

```
mu_LS = (linalg.pinv(N)).dot(E)

elements=['C','H','O','X']

#Create a dictionary with the results
chemical_potentials = {elements[i]: mu_LS[i] for i in range(len(elements))}

chemical_potentials
```

[15]:
```
{'C': -24919.929953558047,
 'H': -1585.3403049171686,
 'O': -55770.66194749069,
 'X': -36434292.46914072}
```

It is then straightforward to calculate the relative enthalpies of the species using the least-squares chemical potentials

$$\underline{\mathbf{H}}_{\mathrm{LS}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{N}}} \cdot^{\mathrm{LS}} \tag{4}$$

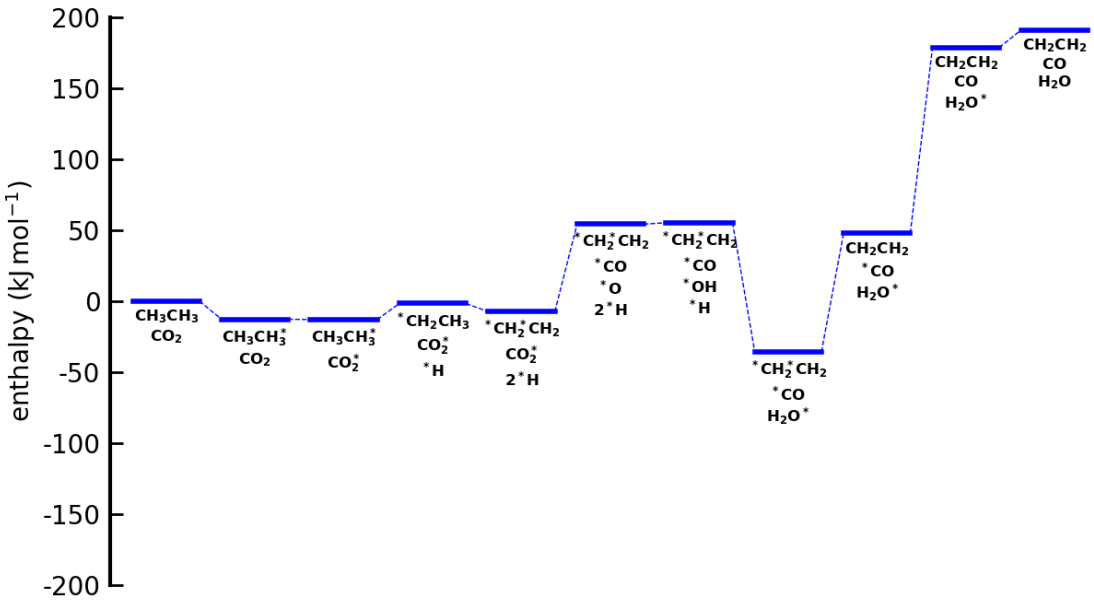
[16]:
```
#Calculate the enthalpies of formation with the chemical potentials

H_LS=E-N.dot(np.array(list(chemical_potentials.values())))

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

[16]:
```
{'X': 5.762220725417137,
 'C2H6': -15.511144822390634,
 'H2O': -12.787864500467549,
 'CO2': -34.4222845005861,
 'C2H6X': -28.226390160620213,
 'XCH2CH3': -7.216875575482845,
 'XO': 84.20153620839119,
 'XCH2XCH2': -3.1983694955706596,
 'XCO': -57.16221673041582,
 'CO2X': -34.597364738583565,
 'XOH': 75.48421062529087,
 'H2OX': -25.308750450611115,
 'XH': -9.738000363111496,
 'CO': 73.61238335924281,
 'C2H4': 80.43752134274837}
```

[17]:
```
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)
```



[18]:
```
#Alternatively, this can be formulated with a linear regression

def calculate_formation_energy(energy_dict, atoms_count_array):
    formation_en_dict = {}
    energy_array = np.array(list(energy_dict.values()))*eV_to_kJmol
    molecules = list(energy_dict.keys())
    reg = LinearRegression(fit_intercept = False)
    reg.fit(atoms_count_array, energy_array)
    predicted_energy = reg.predict(atoms_count_array)
    formation_energy = energy_array - predicted_energy

    for i, molecule in enumerate(molecules):
        formation_en_dict[molecule] = formation_energy[i]

    return formation_en_dict

formation_en = calculate_formation_energy(species_energies, N)
formation_en
```

[18]:
```
{'X': 5.762220725417137,
 'C2H6': -15.51114482034609,
 'H2O': -12.787864482830628,
 'CO2': -34.42228447421803,
 'C2H6X': -28.226390153169632,
 'XCH2CH3': -7.216875568032265,
 'XO': 84.20153622329235,
 'XCH2XCH2': -3.198369488120079,
```

```
 'XCO': -57.16221671551466,
 'CO2X': -34.59736470878124,
 'XOH': 75.48421064019203,
 'H2OX': -25.308750435709953,
 'XH': -9.738000363111496,
 'CO': 73.61238337081159,
 'C2H4': 80.43752134195529}
```

In a more generic way, this can be formulated as determining corrections to a set of energies to align them with known energies of higher fidelity. In the case of two DFT datasets obtained from two different functionals, we can either compare the EOFs or determine atomic correction factors to align the energies of, e.g., the RPBE functional to align with the BEEF energy values. In fact, it can be shown that the least-squares anchor is also the anchor set that minimizes the sum of squared deviation between two different datasets obtained from different sources. Thus, the minimum sum of squares chemical potentials can also be used directly to align data from different sources, e.g., from different functionals such as BEEF-vdW and RPBE or different levels of theory.

$$\hat{\underline{\mathbf{E}}}^{\mathrm{RPBE}} = \underline{\mathbf{E}}^{\mathrm{RPBE}} + \underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{+}\left(\underline{\mathbf{E}}^{\mathrm{BEEF\text{-}vdW}} - \underline{\mathbf{E}}^{\mathrm{RPBE}}\right) \tag{5}$$

where $\hat{\underline{\mathbf{E}}}^{\mathrm{RPBE}}$ are the aligned energies of the RPBE functional to the BEEF-vdW functional. The equation allows to align the data from various sources to the same QM zero-of-energy, which is useful in data science and machine learning for providing an unbiased error between different levels of theory.

[19]:
```
#The method can also be used to align different DFT data sets

species_energies_RPBE = {
    "X": -366518.205,
    "C2H6": -600.361,
    "H2O": -600.7389158,
    "CO2": -1389.568,
    "C2H6X" : -367118.506,
    "XCH2CH3": -367102.198,
    "XO": -367085.354,
    "XCH2XCH2": -367086.181,
    "XCO": -367339.346,
    "CO2X": -367907.753,
    "XOH": -367101.878,
    "H2OX": -367118.949,
    "XH": -366534.422,
    "CO": -819.812,
    "C2H4": -567.332,
}

#vector of DFT energies of the target species from the dictionary
E_RPBE=np.array(list(species_energies_RPBE.values()))*eV_to_kJmol
```

[20]:
```
#Calculate the DFT energies of RPBE aligned to the BEEF-vdW data with the␣
↪chemical potentials

E_RPBE_aligned=E_RPBE+N.dot((linalg.pinv(N)).dot(E-E_RPBE))

#Create a dictionary with the results
DFT_RPBE_aligned = {species[i]: E_RPBE_aligned[i] for i in range(len(species))}
DFT_RPBE_aligned
```

[20]:
```
{'X': -36434303.43239735,
 'C2H6': -59373.45291362446,
 'H2O': -58957.91270217598,
 'CO2': -136499.4476414965,
 'C2H6X': -36493671.09621097,
 'XCH2CH3': -36492056.46617737,
 'XO': -36489938.116766356,
 'XCH2XCH2': -36490469.913278766,
 'XCO': -36515044.887839854,
 'CO2X': -36570800.950338855,
 'XOH': -36491573.58755996,
 'H2OX': -36493261.83564857,
 'XH': -36435909.28229596,
 'CO': -80613.22687749541,
 'C2H4': -56104.34454141254}
```

[21]:
```
#Calculate the difference between the DFT energies of BEEF-vdW and RPBE in kJ/
↪mol

Delta_DFT_unaligned = {species[i]: E[i]-E_RPBE[i] for i in range(len(species))}
Delta_DFT_unaligned
```

[21]:
```
{'X': -1070777.6974949986,
 'C2H6': -1441.5817964414964,
 'H2O': -991.8361308624953,
 'CO2': -2423.207653040008,
 'C2H6X': -1072243.545857504,
 'XCH2CH3': -1072210.6734180003,
 'XO': -1071748.5488620028,
 'XCH2XCH2': -1072166.7148519978,
 'XCO': -1072303.4244484976,
 'CO2X': -1073208.7721489966,
 'XOH': -1071748.2883525044,
 'H2OX': -1071787.326183498,
 'XH': -1070813.840776004,
 'CO': -1517.4186976894998,
 'C2H4': -1361.7555854420207}
```

[22]:
```
#Calculate the difference between the DFT energies of BEEF-vdW and RPBE in kJ/
 ↪mol

Delta_DFT_aligned = {species[i]: E[i]-E_RPBE_aligned[i] for i in␣
 ↪range(len(species))}
Delta_DFT_aligned
```

[22]:
```
{'X': 16.72547735273838,
 'C2H6': 6.0400321829656605,
 'H2O': 3.782280350482324,
 'CO2': 3.7715084564697463,
 'C2H6X': -1.5010565295815468,
 'XCH2CH3': -9.781270630657673,
 'XO': -40.812785647809505,
 'XCH2XCH2': -6.975358232855797,
 'XCO': 4.664581350982189,
 'CO2X': 12.629984855651855,
 'XOH': 0.6003774553537369,
 'H2OX': 2.7152000665664673,
 'XH': 21.734849952161312,
 'CO': -3.752640194084961,
 'C2H4': 3.5609359705194947}
```

Deriving_enthalpies_of_formation_using_gas_phase_references

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)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

eV_to_kJmol=96.485
```

1 Deriving enthalpies of formation of adsorbates using gas-phase reference species

Enthalpies of formation of adsorbates have to be anchored to the existing global thermochemical network of gas-phase species. All species are anchored to the elements in their IUPAC standard states. For this method we are going to form an atomic reference basis set, where we select a single gas-phase species for every element e.g. $[CH_4, H_2, H_2O]$ and Pt(111) for the active site. The enthalpies of formation of the reference species can be obtained from the Active Thermochemical Tables (https://atct.anl.gov/). We assume that the enthalpy of formation of Pt(111) represents bulk Pt and, thus, its enthalpy of formation is 0 by assertion. To determine the enthalpies of formation of the target species with respect to the references in the global thermochemical network, we make a hypothetical reaction to create the target from our references.

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

The enthalpy of reaction is defined as

$$\Delta_r H = \sum_i^N \nu_i \Delta_f H_i \tag{2}$$

where we know the $\Delta_f H_i$ of the reactants (our reference species) from the ATcT. The equation can be rearranged to determine the enthalpy of formation of the target.

$$\Delta_{\mathrm{f}} H_{\mathrm{P}} = \Delta_{\mathrm{r}} H - \sum_{i \neq P}^{N} \nu_i \Delta_{\mathrm{f}} H_i \tag{3}$$

The enthalpy of reaction can also be computed from the known DFT energies of the target and the reference species.

$$\Delta_{\mathrm{f}} H_{\mathrm{P}} = E_{\mathrm{P}} + \sum_{i \neq P}^{N} \nu_i E_i \tag{4}$$

Substituting the enthalpy of reaction leads to the equation to determine the enthalpy of formation of the target with respect to the IUPAC refernence species.

$$\Delta_{\mathrm{f}} H_{\mathrm{P}} = E_{\mathrm{P}} + \sum_{i \neq P}^{N} \nu_i E_i - \sum_{i \neq P}^{N} \nu_i \Delta_{\mathrm{f}} H_i \tag{5}$$

In matrix notation this reads as

$$\underline{\mathbf{H}}_{\mathrm{f}} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^{\mathrm{R}} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_{\mathrm{f}}^{\mathrm{R}} \tag{6}$$

$$\underline{\mathbf{H}}_{\mathrm{f}} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\left(\underline{\mathbf{E}}^{\mathrm{R}} - \underline{\mathbf{H}}_{\mathrm{f}}^{\mathrm{R}}\right) \tag{7}$$

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 $\underline{\underline{\mathbf{N}}}^{\mathrm{R}}$

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{\mathrm{R}^{-1}} \tag{8}$$

When anchoring the DFT energies to the global thermochemical network, it is no longer necessary to perform DFT calculations for gas-phase species. Instead, it is possible to simply use the tabulated enthalpies of formation in the ATcT for the gas-phase species as it is in the same reference frame. In our case, we can take known enthalpies of formation for C_2H_6, C_2H_4, CO, CO_2.

This method adjusts all adsorption enthalpies of the gas-phase species that are not used as the references. The enthalpies of adsorption of the references are the DFT values and assumed to be correct.

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

[7]:
```
#Target species for which we need to compute the EOFs
#all gas-phase species are removed since we use the ATcT values for the␣
 ↪enthalpies of formation
species_energies = {
    "C2H6X" : -378231.5655,
    "XH": -377632.6636,
    "XCH2CH3": -378214.9168,
    "XO": -378193.2832,
```

```
    "XCH2XCH2": -378198.4442,
    "XCO": -378453.0261,
    "CO2X": -379030.8164,
    "XOH": -378209.8045,
    "H2OX": -378227.2801,
} #values are in eV

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

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

[10]:
```
ref_energies = {
    "CH4": -324.2935569,
    "H2": -32.69844421,
    "H2O": -611.0186083,
    "X": -377616.072,
}#values are in eV

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

For the reference species we have to collect the enthalpies of formation from the ATcT

[11]:
```
ref_EOF = {
  "CH4":-66.557,
  "H2":0,
  "H2O":-238.929,
  "X":0,
}#values are in kJ/mol

H_R=np.array(list(ref_EOF.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.

[12]:
```
# Define the species and their elemental compositions in a dictionary
species_compositions = {
    "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},
    "XOH": {"C": 0, "H": 1, "O": 1, "X": 1},
    "H2OX": {"C": 0, "H": 2, "O": 1, "X": 1},
```

```
}

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
```

[12]:
```
array([[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.],
       [0., 1., 1., 1.],
       [0., 2., 1., 1.]])
```

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

[13]:
```
# 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_R = 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_R[i, :] = [composition["C"], composition["H"], composition["O"],␣
 ↪composition["X"]]

N_R
```

[13]:
```
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 $\underline{\underline{\mathbf{N}}}^{\mathrm{R}}$ via

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{\mathrm{R}^{-1}} \tag{9}$$

[14]:
```
#Calculate the matrix of stoichiometric coefficients to form the target from␣
 ↪the reference species
M=-N.dot(np.linalg.inv(N_A))
M
```

[14]:
```
array([[-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. ],
       [-0. ,  0.5, -1. , -1. ],
       [-0. , -0. , -1. , -1. ]])
```

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

$$\underline{\mathbf{H}}_{\mathrm{f}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^{\mathrm{R}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_{\mathrm{f}}^{\mathrm{R}} \tag{10}$$

[17]:
```
#Determine the enthalpy of formation of the target
H_f=(E*eV_to_kJmol+M.dot(E_R)*eV_to_kJmol-M.dot(H_R))

#Create a dictionary with the results
enthalpies_of_formation = {species[i]: H_f[i] for i in range(len(species))}
enthalpies_of_formation
```

[17]:
```
{'C2H6X': -94.98606210941077,
 'XH': -23.385831207036972,
 'XCH2CH3': -66.09093740576506,
 'XO': -131.9305997812152,
```

```
 'XCH2XCH2': -54.186821207344536,
 'XCO': -280.13624798363446,
 'CO2X': -467.9413112559914,
 'XOH': -148.53353548282385,
 'H2OX': -257.21210667723415}
```

The enthalpies of formation of our reference species are the known values in the ATcT. Pt(111) is denoted as a generic X and assumed to have an enthalpy of formation of 0.

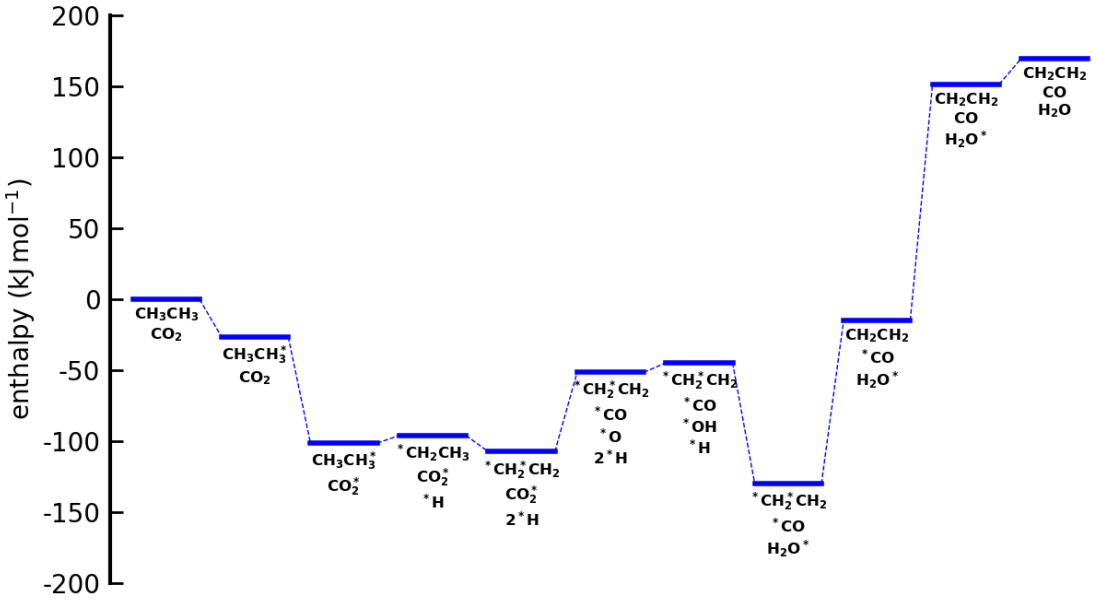
[18]:
```
#Append the enthalpies of formation of the reference species to the enthalpies␣
↪of formation dictionary
enthalpies_of_formation.update({'C2H6': -68.38, 'C2H4': 60.89, 'CO':-113.800,␣
↪'CO2':-393.110})
enthalpies_of_formation.update(ref_EOF)
enthalpies_of_formation
```

[18]:
```
{'C2H6X': -94.98606210941077,
 'XH': -23.385831207036972,
 'XCH2CH3': -66.09093740576506,
 'XO': -131.9305997812152,
 'XCH2XCH2': -54.186821207344536,
 'XCO': -280.13624798363446,
 'CO2X': -467.9413112559914,
 'XOH': -148.53353548282385,
 'H2OX': -257.21210667723415,
 'C2H6': -68.38,
 'C2H4': 60.89,
 'CO': -113.8,
 'CO2': -393.11,
 'CH4': -66.557,
 'H2': 0,
 'H2O': -238.929,
 'X': 0}
```

[19]:
```
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_of_formation)


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)
```


200
150
100
50
0
-50
-100
-150
-200
enthalpy (kJ mol^{-1})
CH_3CH_3 CO_2
$CH_3CH_3^*$ CO_2
$CH_3CH_3^*$ CO_2^*
$^*CH_2CH_3$ CO_2^* $^*H$
$^*CH_2^*CH_2$ CO_2^* 2^*H
$^*CH_2^*CH_2$ $^*CO$ $^*O$ 2^*H
$^*CH_2^*CH_2$ $^*CO$ $^*OH$ $^*H$
$^*CH_2^*CH_2$ $^*CO$ H_2O^*
CH_2CH_2 $^*CO$ H_2O^*
CH_2CH_2 CO H_2O^*
CH_2CH_2 CO H_2O

[ ]:

Deriving_enthalpies_of_formation_using_adsorbate_references

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)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

eV_to_kJmol=96.485
```

1 Deriving enthalpies of formation of adsorbates using adsorbed referenece species

Enthalpies of formation of adsorbates can be anchored to the existing global thermochemical network of gas-phase species through experimental enthalpies of formation of chosen adsorbates. All species are anchored to the elements in their IUPAC standard states. For this method we are going to form an atomic reference basis set, where we select a single adsorbate for every element e.g. $[^*H, ^*CO, ^*O, Pt(111)]$. The enthalpies of formation of the adsorbates are determined from measured enthalpies of adsorption on Pt(111) and enthalpies of formation of the gas-phase species from the ATcT database (https://atct.anl.gov/). When anchoring the DFT energies to the global thermochemical network, it is no longer necessary to perform DFT calculations for gas-phase species. Instead, it is possible to simply use the tabulated enthalpies of formation in the ATcT for the gas-phase species as it is in the same reference frame. In our case, we can take known enthalpies of formation for C_2H_6, C_2H_4, CO, CO_2.

To determine the enthalpies of formation of the target species with respect to the references in the global thermochemical network, we make a hypothetical reaction to create the target from our references.

$$\nu_H {}^*H + \nu_O {}^*O + \nu_{CO} {}^*CO + \nu_* * \rightarrow P \tag{1}$$

The enthalpy of reaction is defined as

$$\Delta_\mathrm{r}H = \Delta_\mathrm{f}H_\mathrm{P} + \sum_{i\neq P}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{2}$$

where we know the $\Delta_\mathrm{f}H_i$ of the reactants. The equation can be rearranged to determine the enthalpy of formation of the target.

$$\Delta_\mathrm{f}H_\mathrm{P} = \Delta_\mathrm{r}H - \sum_{i\neq P}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{3}$$

The enthalpy of reaction can also be computed from the known DFT energies of the target and the reference species.

$$\Delta_\mathrm{f}H_\mathrm{P} = E_\mathrm{P} + \sum_{i\neq P}^{N} \nu_i E_i \tag{4}$$

Substituting the enthalpy of reaction leads to the equation to determine the enthalpy of formation of the target with respect to the IUPAC refernence species.

$$\Delta_\mathrm{f}H_\mathrm{P} = E_\mathrm{P} + \sum_{i\neq P}^{N} \nu_i E_i - \sum_{i\neq P}^{N} \nu_i \Delta_\mathrm{f}H_i \tag{5}$$

In matrix notation this reads as

$$\underline{\mathbf{H}}_\mathrm{f} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^\mathrm{R} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_\mathrm{f}^\mathrm{R} \tag{6}$$

$$\underline{\mathbf{H}}_\mathrm{f} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\left(\underline{\mathbf{E}}^\mathrm{R} - \underline{\mathbf{H}}_\mathrm{f}^\mathrm{R}\right) \tag{7}$$

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 $\underline{\underline{\mathbf{N}}}^\mathrm{R}$

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{\mathrm{R}^{-1}} \tag{8}$$

This is the list of species with their DFT energies from BEEF-vdW for which we want to compute the enthalpies of formation. 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
#all gas-phase species are removed since we use the ATcT values for the␣
 ↪enthalpies of formation
#all adsorbates that are used as reference are removed from the list
species_energies = {
    "C2H6X" : -378231.5655,
    "XCH2CH3": -378214.9168,
    "XCH2XCH2": -378198.4442,
```

```
    "CO2X": -379030.8164,
    "XOH": -378209.8045,
    "H2OX": -378227.2801,
} #values are in eV

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

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

[5]:
```
ref_energies = {
    "XH": -377632.6636,
    "XO": -378193.2832,
    "XCO": -378453.0261,
    "X": -377616.072,
}#values are in eV

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

For the reference species we have to collect the enthalpies of formation from the ATcT

[6]:
```
ref_EOF = {
  "XH":-32.7,
  "XO":-103.7,
  "XCO":-230.9,
  "X":0,
}#values are in kJ/mol

H_R=np.array(list(ref_EOF.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.

[8]:
```
# Define the species and their elemental compositions in a dictionary
species_compositions = {
    "C2H6X": {"C": 2, "H": 6, "O": 0, "X": 1},
    "XCH2CH3": {"C": 2, "H": 5, "O": 0, "X": 1},
    "XCH2XCH2": {"C": 2, "H": 4, "O": 0, "X": 1},
    "CO2X": {"C": 1, "H": 0, "O": 2, "X": 1},
    "XOH": {"C": 0, "H": 1, "O": 1, "X": 1},
    "H2OX": {"C": 0, "H": 2, "O": 1, "X": 1},
}

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
```

[8]:
```
array([[2., 6., 0., 1.],
       [2., 5., 0., 1.],
       [2., 4., 0., 1.],
       [1., 0., 2., 1.],
       [0., 1., 1., 1.],
       [0., 2., 1., 1.]])
```

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

[9]:
```
# Define the species and their elemental compositions in a dictionary
references_compositions = {
    "XH": {"C": 0, "H": 1, "O": 0, "X": 1},
    "XO": {"C": 0, "H": 0, "O": 1, "X": 1},
    "XCO": {"C": 1, "H": 0, "O": 1, "X": 1},
    "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_R = 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_R[i, :] = [composition["C"], composition["H"], composition["O"],␣
 ↪composition["X"]]

N_R
```

[9]:
```
array([[0., 1., 0., 1.],
       [0., 0., 1., 1.],
       [1., 0., 1., 1.],
```

```
       [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 $\underline{\underline{\mathbf{N}}}^{\mathrm{R}}$ via

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{N}}}\,\underline{\underline{\mathbf{N}}}^{\mathrm{R}^{-1}} \tag{9}$$

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

[10]:
```
array([[-6.,  2., -2.,  5.],
       [-5.,  2., -2.,  4.],
       [-4.,  2., -2.,  3.],
       [-0., -1., -1.,  1.],
       [-1., -1., -0.,  1.],
       [-2., -1., -0.,  2.]])
```

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

$$\underline{\mathbf{H}}_{\mathrm{f}} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^{\mathrm{R}} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_{\mathrm{f}}^{\mathrm{R}} \tag{10}$$

[12]:
```
#Determine the enthalpy of formation of the target
H_f=(E*eV_to_kJmol+M.dot(E_R)*eV_to_kJmol-M.dot(H_R))

#Create a dictionary with the results
enthalpies_of_formation = {species[i]: H_f[i] for i in range(len(species))}
enthalpies_of_formation
```

[12]:
```
{'C2H6X': -108.85977844744923,
 'XCH2CH3': -70.65048498064277,
 'XCH2XCH2': -49.43219997435813,
 'CO2X': -390.4744634985924,
 'XOH': -129.61710450202227,
 'H2OX': -247.60984448939564}
```

The enthalpies of formation of our reference species are the known values in the ATcT. Pt(111) is denoted as a generic X and assumed to have an enthalpy of formation of 0.

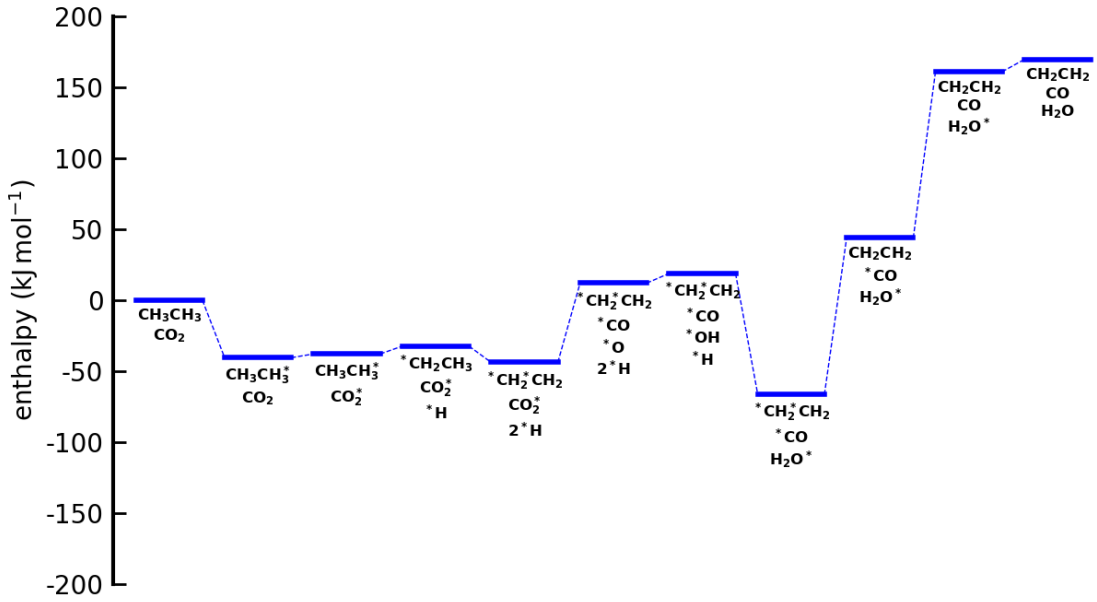
[13]:
```
#Append the enthalpies of formation of the reference species to the enthalpies␣
 ↪of formation dictionary
enthalpies_of_formation.update({'C2H6': -68.38, 'C2H4': 60.89, 'CO':-113.800,␣
 ↪'CO2':-393.110, 'H2O': -238.929})
enthalpies_of_formation.update(ref_EOF)
enthalpies_of_formation
```

[13]:
```
{'C2H6X': -108.85977844744923,
 'XCH2CH3': -70.65048498064277,
 'XCH2XCH2': -49.43219997435813,
 'CO2X': -390.4744634985924,
 'XOH': -129.61710450202227,
 'H2OX': -247.60984448939564,
 'C2H6': -68.38,
 'C2H4': 60.89,
 'CO': -113.8,
 'CO2': -393.11,
 'H2O': -238.929,
 'XH': -32.7,
 'XO': -103.7,
 'XCO': -230.9,
 'X': 0}
```

[15]:
```
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_of_formation)


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)
```



Deriving_enthalpies_of_formation_using_adsorbate_references_in_isodesm

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)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

eV_to_kJmol=96.485
```

1 Reference QM data to a set of experimental EOF of adsorbates in isodesmic reactions

Enthalpies of formation of adsorbates can be anchored to the existing global thermochemical network of gas-phase species through experimental enthalpies of formation of chosen adsorbates. Instead of using an atomic reference basis set, we can use error cancellation reactions that decompose a target into fragments. Through the conservation of bonding environment and hybridization of the target species in the references, it is possible to construct reactions that maximize the cancellation of errors. The easiest fragmentation based approach is based on a isodesmic reaction, which conserve the number of bonds of the target in the references. All adsorbates that are used in the case study can be broken down into the following 10 bond types C-O, C=C, C=O, C-C, C-H, O-H, Pt-C, Pt#C, Pt=C, Pt-O, where - is a single bond, = double bond, # triple bond. Each bond type can be represented by an adsorbate, for which we need the enthalpy of formation anchored to the global gas-phase thermochemical network. Enthalpies of formation of these adsorbates on Pt(111) were derived from experimentally measured heats of adsorption and experimental enthalpies of formation from the ATcT (https://atct.anl.gov/) by Kreitz et al. (https://doi.org/10.1021/acs.jctc.3c00112)

Bond type	species	$\Delta_f H$ / kJ mol^{-1}
Pt#C	$^*CH$	-35.8
Pt=C	$^*CH_2$	46.5
Pt-C	$^*CH_3$	-47.2

Bond type	species	$\Delta_f H$ / kJ mol^{-1}
C-H	CH_4^*	-81.3
C=C	$C_2H_4^*$	22.1
C-C	$C_2H_6^*$	-96.0
O-H	H_2O^*	-267.9
Pt-O	$^*OH$	-164.7
C=O	H_2CO^*	-159.3
C-O	CH_3OH^*	-245.0

All species are anchored to the elements in their IUPAC standard states. When anchoring the DFT energies to the global thermochemical network, it is no longer necessary to perform DFT calculations for gas-phase species. Instead, it is possible to simply use the tabulated enthalpies of formation in the ATcT for the gas-phase species as it is in the same reference frame. In our case, we can take known enthalpies of formation for C_2H_6, C_2H_4, CO, CO_2. Some of the reference species are also part of the mechanism of the case study and we can use the experimental values directly. For some adsorbates like $^*H$ or $^*CO$ it is not possible to make an isodesmic reaction and we have to use experimental values of the enthalpies of formation directly.

To determine the enthalpies of formation of the target species with respect to the references in the global thermochemical network, we make a hypothetical reaction to create the target from our references that represent the fragments.

$$\nu_{\mathrm{F}}\, F \rightarrow P \tag{1}$$

The enthalpy of reaction is defined as

$$\Delta_{\mathrm{r}} H = \Delta_{\mathrm{f}} H_{\mathrm{P}} + \sum_{i \neq P}^{N} \nu_i \Delta_{\mathrm{f}} H_i \tag{2}$$

where we know the $\Delta_{\mathrm{f}} H_i$ of the reactants. The equation can be rearranged to determine the enthalpy of formation of the target.

$$\Delta_{\mathrm{f}} H_{\mathrm{P}} = \Delta_r H - \sum_{i \neq P}^{N} \nu_i \Delta_{\mathrm{f}} H_i \tag{3}$$

The enthalpy of reaction can also be computed from the known DFT energies of the target and the reference species.

$$\Delta_{\mathrm{f}} H_{\mathrm{P}} = E_{\mathrm{P}} + \sum_{i \neq P}^{N} \nu_i E_i \tag{4}$$

Substituting the enthalpy of reaction leads to the equation to determine the enthalpy of formation of the target with respect to the IUPAC refernence species.

$$\Delta_{\mathrm{f}} H_{\mathrm{P}} = E_{\mathrm{P}} + \sum_{i \neq P}^{N} \nu_i E_i - \sum_{i \neq P}^{N} \nu_i \Delta_{\mathrm{f}} H_i \tag{5}$$

In matrix notation this reads as

$$\underline{\mathbf{H}}_\mathrm{f} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^\mathrm{R} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_\mathrm{f}^\mathrm{R} \tag{6}$$

$$\underline{\mathbf{H}}_\mathrm{f} = \underline{\mathbf{E}} + \underline{\underline{\mathbf{M}}}\left(\underline{\mathbf{E}}^\mathrm{R} - \underline{\mathbf{H}}_\mathrm{f}^\mathrm{R}\right) \tag{7}$$

where $\underline{\underline{\mathbf{M}}}$ can be determined from the bond type matrix of the target species $\underline{\underline{\mathbf{F}}}$ and the reference species $\underline{\underline{\mathbf{F}}}^\mathrm{R}$

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{F}}}\,\underline{\underline{\mathbf{F}}}^{\mathrm{R}^{-1}} \tag{8}$$

This is the list of species with their DFT energies from BEEF-vdW for which we want to compute the enthalpies of formation. 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
#all gas-phase species are removed since we use the ATcT values for the␣
 ↪enthalpies of formation
#all adsorbates that are used as reference are removed from the list
#we use experimental values for XH and XCO
species_energies = {
    "XCH2CH3": -378214.9168,
    "XCH2XCH2": -378198.4442,
    "CO2X": -379030.8164,
} #values are in eV

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

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

[3]:
```
ref_energies = {
    "CH3OHX": -378517.9494,
    "CH2CH2X": -378197.7339,
    "H2COX": -378484.4684,
    "CH3CH3X": -378231.5655,
    "CH4X": -377940.4907,
    "H2OX": -378227.2801,
    "XCH3": -377923.7364,
    "XCH": -377890.838,
    "XCH2": -377906.9113,
    "XOH": -378209.8045,
}#values are in eV

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

For the reference species we have to collect the enthalpies of formation from the ATcT

[4]:
```
ref_EOF = {
    "CH3OHX": -245.0,
    "C2H4X": 22.1,
    "H2COX": -159.3,
    "C2H6X": -96.0,
    "CH4X": -81.3,
    "H2OX": -267.9,
    "XCH3": -47.2,
    "XCH": -35.8,
    "XCH2": 46.5,
    "XOH": -164.7,
}#values are in kJ/mol

H_R=np.array(list(ref_EOF.values()))
```

We start now with constructing the molecular fragment composition matrix of our target species $\underline{\underline{\mathbf{F}}}$, which is an $m \times n$ matrix with m target species and n fragments.

[6]:
```
# Define the species and their elemental compositions in a dictionary
species_compositions = {
    "XCH2CH3": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":1, "C-H": 5, "O-H": 0,␣
 ↪"Pt-C":1, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "XCH2XCH2": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":1, "C-H": 4, "O-H": 0,␣
 ↪"Pt-C":2, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "CO2X": {"C-O": 0, "C=C": 0, "C=O": 2, "C-C":0, "C-H": 0, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
}

species=list(species_compositions.keys())

# Create a matrix to hold the elemental compositions of the target species
num_species = len(species_compositions)
num_bond_types = 10  # C-O, C=C, C=O, C-C, C-H, O-H, Pt-C, Pt#C, Pt=C, Pt-O
F = np.zeros((num_species, num_bond_types))

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

F
```

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

We construct the molecular fragment composition matrix of our reference species $\underline{\underline{\mathbf{F}}}^{\mathrm{R}}$, which is an $m \times n$ matrix with m reference species and n fragments.

[8]:
```
# Define the species and their elemental compositions in a dictionary
references_compositions = {
    "CH3OHX": {"C-O": 1, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 3, "O-H": 1,␣
 ↪"Pt-C":0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "C2H4X": {"C-O": 0, "C=C": 1, "C=O": 0, "C-C":0, "C-H": 4, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "H2COX": {"C-O": 0, "C=C": 0, "C=O": 1, "C-C":0, "C-H": 2, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "C2H6X": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":1, "C-H": 6, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "CH4X": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 4, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "H2OX": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 0, "O-H": 2, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "XCH3": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 3, "O-H": 0, "Pt-C":
 ↪1, "Pt#C":0, "Pt=C":0, "Pt-O":0},
    "XCH": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 1, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":1, "Pt=C":0, "Pt-O":0},
    "XCH2": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 2, "O-H": 0, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":1, "Pt-O":0},
    "XOH": {"C-O": 0, "C=C": 0, "C=O": 0, "C-C":0, "C-H": 0, "O-H": 1, "Pt-C":
 ↪0, "Pt#C":0, "Pt=C":0, "Pt-O":1},
}

references=list(references_compositions.keys())

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

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

F_R
```

[8]:
```
array([[1., 0., 0., 0., 3., 1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 4., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 2., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 6., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 4., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 2., 0., 0., 0., 0.],
       [0., 0., 0., 0., 3., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 2., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 1.]])
```

We can now determine $\underline{\underline{\mathbf{M}}}$ from the molecular fragment composition matrix of the target species $\underline{\underline{\mathbf{F}}}$ and the reference species $\underline{\underline{\mathbf{F}}}^{\mathrm{R}}$ via

$$\underline{\underline{\mathbf{M}}} = -\underline{\underline{\mathbf{F}}}\,\underline{\underline{\mathbf{F}}}^{\mathrm{R}^{-1}} \tag{9}$$

[9]:
```
#Calculate the matrix of stoichiometric coefficients to form the target from␣
↪the reference species
M=-F.dot(np.linalg.inv(F_R))
M
```

[9]:
```
array([[-0., -0., -0., -1.,  1., -0., -1., -0., -0., -0.],
       [-0., -0., -0., -1.,  2., -0., -2., -0., -0., -0.],
       [-0., -0., -2., -0.,  1., -0., -0., -0., -0., -0.]])
```

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

$$\underline{\mathbf{H}}_{\mathrm{f}} = \underline{\mathbf{E}} - \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{E}}^{\mathrm{R}} + \underline{\underline{\mathbf{M}}}\,\underline{\mathbf{H}}_{\mathrm{f}}^{\mathrm{R}} \tag{10}$$

[10]:
```
#Determine the enthalpy of formation of the target
H_f=(E*eV_to_kJmol+M.dot(E_R)*eV_to_kJmol-M.dot(H_R))

#Create a dictionary with the results
enthalpies_of_formation = {species[i]: H_f[i] for i in range(len(species))}
enthalpies_of_formation
```

[10]:
```
{'XCH2CH3': -72.08881599605084,
 'XCH2XCH2': -65.16864050477744,
 'CO2X': -465.99839549809695}
```

The enthalpies of formation of our reference species are the known values in the ATcT. Pt(111) is denoted as a generic X and assumed to have an enthalpy of formation of 0.

[11]:
```
#Append the enthalpies of formation of the reference species to the enthalpies␣
↪of formation dictionary
```

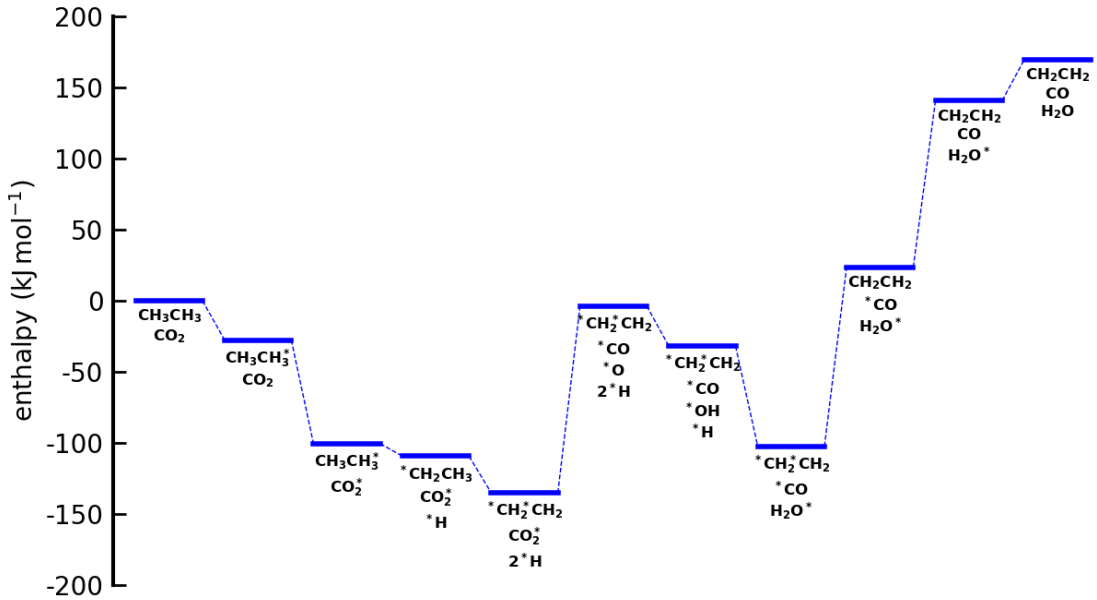
```
enthalpies_of_formation.update({'C2H6': -68.38, 'C2H4': 60.89, 'CO':-113.800,␣
 ↪'CO2':-393.110, 'H2O': -238.929})
enthalpies_of_formation.update(ref_EOF)
enthalpies_of_formation.update({'XCO': -230.9, 'XH':-32.7, 'XO':-103.7})
enthalpies_of_formation
```

[11]:
```
{'XCH2CH3': -72.08881599605084,
 'XCH2XCH2': -65.16864050477744,
 'CO2X': -465.99839549809695,
 'C2H6': -68.38,
 'C2H4': 60.89,
 'CO': -113.8,
 'CO2': -393.11,
 'H2O': -238.929,
 'CH3OHX': -245.0,
 'C2H4X': 22.1,
 'H2COX': -159.3,
 'C2H6X': -96.0,
 'CH4X': -81.3,
 'H2OX': -267.9,
 'XCH3': -47.2,
 'XCH': -35.8,
 'XCH2': 46.5,
 'XOH': -164.7,
 'XCO': -230.9,
 'XH': -32.7,
 'XO': -103.7}
```

[12]:
```
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_of_formation)


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)
```



Converting_between_reaction_and_relative_enthalpies

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)

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from sklearn.linear_model import LinearRegression
from scipy import linalg

eV_to_kJmol=96.485
```

1 Converting between reaction enthalpies and formation enthalpies

The reaction enthalpies calculated from the total DFT energies are conserved when using an atomic reference basis set, regardless of the choice of anchor/reference species. We can use this fact and calculate EOFs if only reaction enthalpies are reported for a mechanism. This problem cannot be formulated for a single species, and it directly becomes a linear algebra problem. For a chemical kinetics network involving n chemical species, a total of $n_a + n_s$ reference species must be defined, where n_a is the number of unique atomic elements of which all species consist and n_s is the number of surface sites (typically one). Let $\underline{\mathbf{E}}$ be the zero-point corrected energies of chemical species, $\underline{\mathbf{H}}_{\mathrm{A}}$ be the EOFs of the same species given defined references A and $\underline{\mathbf{H}}_{\mathrm{r}}$ be the reaction energies according to the stoichiometry matrix $\underline{\underline{\mathbf{M}}}$. The relationship between $\underline{\mathbf{H}}_{\mathrm{r}}$ and $\underline{\mathbf{E}}$ is a simple linear combination,

$$\underline{\mathbf{H}}_{\mathrm{r}} = \underline{\underline{\mathbf{M}}}^{\top} \underline{\mathbf{H}}_{\mathrm{A}} = \underline{\underline{\mathbf{M}}}^{\top} \underline{\mathbf{E}} \tag{1}$$

Since the number of elementary reactions is typically greater than or equal to the number of chemical species, i.e., $m \geq n$, there exists a direct mapping between $\underline{\mathbf{E}}$ and $\underline{\mathbf{H}}_{\mathrm{r}}$. Yet, no obvious mapping exists in the opposite direction since this is an under-constrained linear system of equations. The Moore-Penrose pseudo-inverse of the stoichiometry matrix $\underline{\underline{\mathbf{M}}}^{+}$ can be used to construct the inverse mapping. To solve this inverse mapping, it is first necessary to select anchor species for each element

from the mechanism, e.g., $A = C_2H_6, CO, H_2O, Pt(111)$. We can separate the stoichiometry matrix $\underline{\underline{\mathbf{M}}}$ into the anchor species $\underline{\underline{\mathbf{M}}}_{\mathrm{A}}$ and non-anchor species $\hat{\underline{\underline{\mathbf{M}}}}$, leading to

$$\underline{\mathbf{H}}_{\mathrm{r}} = \underline{\underline{\mathbf{M}}}^{\top} \underline{\mathbf{H}}_{\mathrm{A}} = \hat{\underline{\underline{\mathbf{M}}}}^{\top} \hat{\underline{\mathbf{H}}}_{\mathrm{A}} + \underline{\underline{\mathbf{M}}}_{\mathrm{A}}^{\top} \tilde{\underline{\mathbf{H}}}_{\mathrm{A}} \tag{2}$$

In this equation, we separate $\underline{\mathbf{H}}_{\mathrm{A}}$ into the anchor species $\tilde{\underline{\mathbf{H}}}_{\mathrm{A}}$ and the unknown EOFs $\hat{\underline{\mathbf{H}}}_{\mathrm{A}}$. All EOFs are referenced to the DFT energies of the anchor species. Thus, the EOFs of the anchor species $\underline{\mathbf{H}}_{\mathrm{A}}$ are 0.
We can now rearrange the equation to compute the unknown EOFs from the reaction enthalpies referenced to the set of anchor species.

$$\underline{\mathbf{H}}_{\mathrm{r}} = \hat{\underline{\underline{\mathbf{M}}}}^{\top} \hat{\underline{\mathbf{H}}}_{\mathrm{A}} + \underline{\underline{\mathbf{M}}}_{\mathrm{A}}^{\top} \tilde{\underline{\mathbf{H}}}_{\mathrm{A}} \tag{3}$$

$$\hat{\underline{\mathbf{H}}}_{\mathrm{A}} = \left(\hat{\underline{\underline{\mathbf{M}}}}^{\top}\right)^{+} \left(\underline{\mathbf{H}}_{\mathrm{r}} - \underline{\underline{\mathbf{M}}}_{\mathrm{A}}^{\top} \tilde{\underline{\mathbf{H}}}_{\mathrm{A}}\right) \tag{4}$$

$$\hat{\underline{\mathbf{H}}}_{\mathrm{A}} = \left(\hat{\underline{\underline{\mathbf{M}}}}^{\top}\right)^{+} \left(\underline{\underline{\mathbf{M}}}^{\top} \underline{\mathbf{E}} - \underline{\underline{\mathbf{M}}}_{\mathrm{A}}^{\top} \tilde{\underline{\mathbf{H}}}_{\mathrm{A}}\right) \tag{5}$$

We start by constructing the matrix of stoichiometric matrix $\underline{\underline{\mathbf{M}}}$.

[2]:
```
# Stoichiometry of each elementary reaction in the mechanism
reactions = [
    {"X": -1, "C2H6": -1, "C2H6X": 1},
    {"X": -1, "CO2": -1, "CO2X": 1},
    {"X": -1, "C2H6X": -1, "XH": 1, "XCH2CH3": 1},
    {"X": -1, "XCH2CH3": -1, "XH": 1, "XCH2XCH2": 1},
    {"X": -1, "CO2X": -1, "XCO": 1, "XO":1},
    {"XO": -1, "XH": -1, "X": 1, "XOH": 1},
    {"XOH": -1, "XH": -1, "H2OX": 1, "X": 1},
    {"H2OX": -1, "H2O": 1, "X": 1},
    {"XCO": -1, "CO": 1, "X": 1},
    {"XCH2XCH2": -1, "C2H4": 1, "X":1},
]

#Species in the reaction mechanism
species = ['CO2', 'C2H4', 'C2H6X', 'XCH2CH3', 'XO', 'XCH2XCH2', 'XCO',
           'CO2X', 'XOH', 'H2OX', 'XH', 'X', 'C2H6','H2O', 'CO']
#the last for species will be our reference species

# Creating the stoichiometry matrix
M = np.zeros((len(species), len(reactions)))

# Filling the matrix
for j, reaction in enumerate(reactions):
    for reactant, stoich_coeff in reaction.items():
        i = species.index(reactant)
        M[i, j] = stoich_coeff
```

```
M
```

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

We can then calculate the enthalpies of reaction from the BEEF-vdW DFT energies

$$\underline{\mathbf{H}}_{\mathrm{r}} = \underline{\underline{\mathbf{M}}}^{\top}\underline{\mathbf{E}} \tag{6}$$

[3]:
```
#We use the BEEF-vdW DFT energies to calculate the reaction enthalpies for the␣
 ↪reactions above
species_energies = {
    "X": -377616.072,
    "C2H6": -615.3019939,
    "H2O": -611.0186083,
    "CO2": -1414.682864,
    "C2H6X" : -378231.5655,
    "XCH2CH3": -378214.9168,
    "XO": -378193.2832,
    "XCH2XCH2": -378198.4442,
    "XCO": -378453.0261,
    "CO2X": -379030.8164,
    "XOH": -378209.8045,
    "H2OX": -378227.2801,
    "XH": -377632.6636,
    "CO": -835.5389907,
    "C2H4":-581.44565067567,
}

#Sort the species energies according to the defined species list
E = np.array([species_energies[_]*eV_to_kJmol for _ in species])

#Calculate the enthalpies of formation
```

```
Hr=M.T.dot(E)
Hr
```

[3]:
```
array([-18.47746606,  -5.93730097,   5.5092935 , -11.48171501,
        55.87446348,   6.78289551, -85.29273999,  18.28310668,
       136.53682082,  89.39811156])
```

We select anchor species for each element from the mechanism, in this case $A = C_2H_6, CO, H_2O, Pt(111)$. Since we are creating a local thermochemical network in the DFT energy reference frame, we set the enthalpies of formation of these anchor species to 0.

[4]:
```
# Enthalpies of formation of the reference species are assumed to be zero

H_A_dict = {
    "X": 0,
    "C2H6": 0,
    "H2O": 0,
    "CO":0,
}

ref_species=H_A_dict.keys()

#Remove the reference species from the row space
M_A=M[-4:] # these are the reference species
M_hat=M[0:-4] # this is the rest of the stoichiometry matrix

#Array of the enthalpies of formation of the anchor species
tilde_H_A =  np.array( [H_A_dict[_] for _ in H_A_dict.keys()])
tilde_H_A
```

[4]:
```
array([0, 0, 0, 0])
```

We can then determine the enthalpies of formation of the target species via

$$\underline{\hat{\mathbf{H}}}_{\mathrm{A}} = \left(\underline{\underline{\hat{\mathbf{M}}}}^{\top}\right)^{+} \left(\underline{\underline{\mathbf{M}}}^{\top}\,\underline{\mathbf{E}} - \underline{\underline{\mathbf{M}}}_{\mathrm{A}}^{\top}\,\underline{\tilde{\mathbf{H}}}_{\mathrm{A}}\right) \tag{7}$$

[5]:
```
#Calculate the enthalpies of formation of the target species using the␣
 ↪pseudo-inverse
H_A= np.linalg.pinv(M_hat.T).dot(Hr-M_A.T.dot(tilde_H_A))

targets=species-ref_species

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

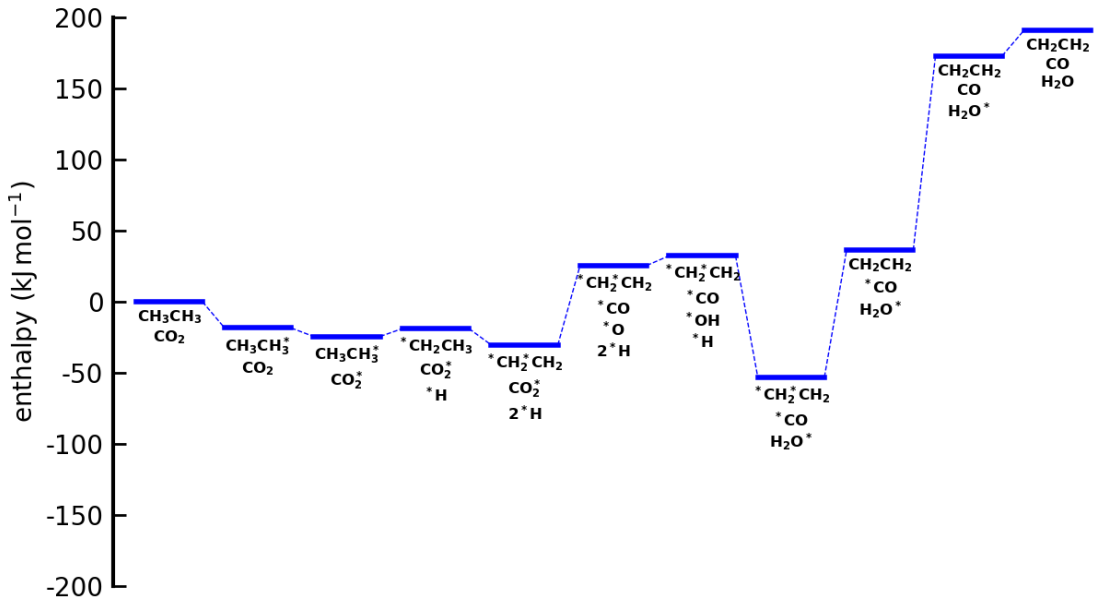
```
#Append the enthalpies of formation of the reference species to the enthalpies␣
 ↪of formation dictionary
enthalpies_relative_to_anchors.update(H_A_dict)
enthalpies_relative_to_anchors
```

[5]:
```
{'CO2': -103.51924771423779,
 'C2H4': 87.67622179997443,
 'C2H6X': -18.477466061631237,
 'XCH2CH3': -1.6041736579410335,
 'XO': 82.95473561961586,
 'XCH2XCH2': -1.7218897594761842,
 'XCO': -136.53682081535223,
 'CO2X': -109.45654867942778,
 'XOH': 78.37363221796271,
 'H2OX': -18.28310667649201,
 'XH': -11.363998907081388,
 'X': 0,
 'C2H6': 0,
 'H2O': 0,
 'CO': 0}
```

[6]:
```
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)
```



[ ]: