

## Python Group Additivity (pGrAdd) Software for Estimating Species Thermochemical Properties

Gerhard R. Wittreich<sup>a</sup> and Dionisios G. Vlachos<sup>a,\*</sup>

<sup>a</sup>Department of Chemical and Biomolecular Engineering  
Rapid Advancement in Process Intensification Deployment (RAPID) Institute  
University of Delaware, Newark, DE 19716, United States

\*Corresponding author: [vlachos@udel.edu](mailto:vlachos@udel.edu)

### Abstract

Increasingly complex chemistry models require thermochemical data for many species often estimated from costly first-principles DFT computations. We introduce the Python Group Additivity software (pGrAdd) that implements comprehensive group additivity in a simple, modular, lightweight Python package that is extensible and easy to implement. It includes 6 group additivity databases for gas species and Pt(111) adsorbates allowing users to immediately compute thermochemical properties for a wide range of molecules and build new databases.

### Program summary

Program title: pGrAdd

Licensing provisions: MIT license (MIT)

Programming language: Python

External routines: pMuTT, rdkit, ipython, numpy, pyyaml, scipy

Nature of problem: Estimation of thermochemical properties for many molecules.

Solution method: A Python package with group contribution databases and group schemes that employs first principles data estimation to estimate molecular thermodynamic properties.

### Keywords

Species thermochemistry, catalysis, Python

## Introduction

Multiscale kinetic modeling is vital for catalyst discovery, elucidation of reaction kinetics and mechanisms, and reaction optimization.[1–3] Microkinetic models (MKMs) require thermodynamic and kinetic parameters obtained from DFT data. As mechanisms grow in size and complexity, generating the necessary DFT data becomes intractable. First-principles semi-empirical methods offer an approach to parameterization at a lower cost. Bronsted-Evans-Polyani (BEP)[4–7] relationships are one such estimation method of reaction barriers from thermodynamic properties without costly transition state DFT calculations. Group additivity (GA) does the same for the thermodynamic properties of molecules. GA, first proposed by Benson and coworkers,[8,9] represents molecules with groups whose thermodynamic contribution is summed to estimate the thermodynamic property. A set of training molecules is selected, the thermochemistry computed via DFT and statistical mechanics, and the values of the groups are regressed. The GA method was initially introduced for gas molecules and later extended to heterogeneous systems[10]. Once training is complete, the thermochemistry of a broad set of adsorbates can be computed at nearly no cost.

The Python Group Additivity (pGrAdd) software introduced here facilitates the parameterization of thermodynamic properties given the molecule's SMILES string. pGrAdd is a Python-based series of libraries and group value databases designed in modules to expand the code functionality. Group definitions are codified in a scheme file and associated with group values in a database to determine each group's frequency in a molecule and its thermodynamic value.

Ease of use, parametrization speed, and extensibility were our goals in building pGrAdd. Python is well supported and used extensively in the scientific community. Functionality can easily be extended through new Python modules and new GA databases (user defined schemes and group values) can also be added. The software also comes preconfigured with the Benson's [8,9] gas database and 5 additional databases for Pt(111) adsorbates. pGrAdd is part of the VLab suite: RAPID Reaction Software Ecosystem of kinetic modeling tools. (<https://dei.udel.edu/rapid/research/rapid-reaction-software-ecosystem/>)

## Comparison to Other Software and Dependencies

Table 1 shows software packages typically used to determine the thermodynamic properties of molecules along with their pros and cons.

**Table 1: Available software packages employed in determining thermodynamic properties of molecules.**

Software	Pros	Cons	Reference
Python Multiscale Thermochemistry Toolbox (pMuTT)	<ul style="list-style-type: none"><li>- Open-source</li><li>- Python-based; libraries can be added to Python code</li><li>- Computes thermodynamic and kinetic parameters</li><li>- Modular and easily expanded</li><li>- Good integration with kinetic packages</li></ul>	<ul style="list-style-type: none"><li>- No libraries to estimate thermochemical properties</li><li>- Requires DFT data or empirical relationships (NASA or Shomate polynomials)</li></ul>	[11]

---

	- Integrated with pGrAdd		
Atomic Simulation Environment (ASE)	<ul style="list-style-type: none"> <li>- Open-source</li> <li>- Python-based; libraries can be added to Python code</li> <li>- Modular and easily expanded</li> <li>- Long history, well supported</li> <li>- Integration with DFT packages</li> <li>- Adsorbate visualization</li> </ul>	<ul style="list-style-type: none"> <li>- No libraries to estimate thermochemical properties</li> <li>- Requires DFT data</li> </ul>	[12]
CanTherm	<ul style="list-style-type: none"> <li>- Open-source</li> <li>- Python-based</li> <li>- Computes thermodynamic and kinetic parameters using statistical mechanical models</li> <li>- High-pressure limit kinetics calculations</li> <li>- <b>Bundled with RMG-3.0</b></li> </ul>	<ul style="list-style-type: none"> <li>- No libraries of data to estimate thermochemical properties</li> <li>- Requires DFT data</li> <li>- Integrated primarily with Gaussian</li> <li>- Integrated with RMG for reaction mechanism generation</li> </ul>	[13]
Reaction Mechanism Generator (RMG-3.0)	<ul style="list-style-type: none"> <li>- Open-source</li> <li>- Python-based</li> <li>- Computes thermodynamic and kinetic parameters</li> <li>- Integration with multiple DFT packages</li> <li>- Contains libraries of species (gas and surface) thermochemical and group additivity data</li> <li>- ~24k gas species in 55 databases</li> <li>- 2.6k GA functional groups and 3.2k</li> <li>- Corrections in 8 databases</li> <li>- 91 surface species in 2 databases (Ni and Pt)</li> <li>- 5.8k gas group contribution and correction values</li> <li>- 82 surface group correction values Pt(111)</li> <li>- Supports limited surface chemistry-merged with RMG-Cat project</li> </ul>	<ul style="list-style-type: none"> <li>- Not lightweight: RMG has many features beyond GA, adding complexity if GA is the only need</li> <li>- Limited surface species data</li> <li>- No GA model/data for surface species</li> </ul>	[13]
Open Catalyst Project/ Open Catalyst 2020 (OC20) Dataset	<ul style="list-style-type: none"> <li>- <b>Open-source library of relaxed adsorbate-surface pairs.</b></li> <li>- <b>Machine learning implemented to predict</b></li> </ul>	<ul style="list-style-type: none"> <li>- <b>Limited to the adsorbate/surface pairs included in the database</b></li> <li>- <b>Does not include a GA scheme. User would need</b></li> </ul>	[14]

- future adsorbate-surface pairs without costly DFT to develop a scheme from the OC20 data.
- Consists of ~1.3 M Density Functional Theory (DFT) simulations across a wide range of materials, surfaces, and adsorbates
- Great source to train GA or machine learning schemes

pGrAdd offers several advantages over existing thermodynamic estimators. First, it includes group value databases and schemes (ruleset defining atom connectivity and corrections) that enable a user to immediately parametrize the thermodynamic properties of molecules without performing DFT calculations. Second, the databases focus on heterogeneous catalysis filling a gap of other software packages. Third, the software is lightweight, quickly providing thermodynamic values for molecules, resulting in a short learning curve and rapid results. Finally, the software is easily expanded using a common scientific coding language (Python), and data and schemes can easily be added customized to the end user's needs.

pGrAdd depends on several additional Python libraries:

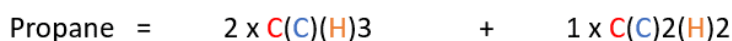
- **Python 3:** The latest release of the Python coding language
- **pMuTT:** Python multiscale thermodynamic toolbox is used for unit conversions, input of molecule SMILES, output to various empirical relationships (NASA polynomial, Shomate polynomial, etc., and MKM software input (e.g., Chemkin thermdat)[11]
- **RDKit:** Adds the ability to parse SMILES string in determining constituent groups by converting SMILES strings to molecule objects[15]
- **iPython:** Shell and interactive use[16]
- **NumPy:** Linear algebra and vector/matrix manipulation[17]
- **pyYaml:** Reads and writes YAML formatted files
- **SciPy:** Extends NumPy scientific computations[18]

pGrAdd resides on the Python Package Index (PyPI) software repository (<https://pypi.org/project/pypi/>) and Github repository (<https://github.com/VlachosGroup/PythonGroupAdditivity>) and can be installed using pip (Python package management software).

## Theory

The general limiting law for solutions dictates that when very dilute, the interactions between components are negligible, and solution properties are additive. The values of enthalpy, entropy, free energy, and heat capacity can be obtained from sums of groups (combination of atoms and bonds) and can be treated similarly due to the short-range nature of atomic forces. Therefore, given the thermodynamic contribution of a group, we can sum the properties of all the constituent groups to obtain the molecular property. Groups are defined as a central atom (graph theory vertex) and peripheral atoms (graph theory neighbors) bonded to the central atom. Figure 1 shows an example for a propane molecule with the two groups and their frequency of occurrence defined using the Benson group scheme along with their thermodynamic

contributions. The group includes a central atom (first atom listed without parentheses) and all atoms bonded to it (peripheral atoms in parentheses followed by a number representing the frequency of occurrence). In Figure 1, the first group in propane starts with a central carbon (the C1 carbon) bonded to a peripheral carbon (C2) and three peripheral hydrogens. Atoms can be central in one group and peripheral in another, as in the propane example. The C1 and C3 carbons are central in the first group and C2 is a peripheral atom, whereas C2 is the central atom in the second with C1 and C3 as peripheral. Summing these contributions multiplied with the frequencies of occurrence estimates the thermodynamic properties. We will later show that these thermodynamic values agree very well ( $\pm < 1$  kJ/mol in enthalpy) with experimental values.



Benson Gas Group Contributions [298.15 K]			
Benson Group	$C_p$ [J mol <sup>-1</sup> K <sup>-1</sup> ]	H [kJ/mol]	S [J mol <sup>-1</sup> K <sup>-1</sup> ]
C(C)(H) <sub>3</sub>	25.90	-42.68	127.24
C(C) <sub>2</sub> (H) <sub>2</sub>	23.01	-20.63	39.41
<b>Propane</b>	<b>2x25.90 + 1x23.01 = 74.81</b>	<b>2x-42.68 + 1x-20.63 = -105.99</b>	<b>2x127.24 + 1x39.41 = 293.89</b>

**Figure 1: Group additivity example using the Benson scheme for propane gas molecule at 298.15 K.**

In practice, interactions between a molecule's functional groups, such as alkane gauche conformations, hydrogen bonding, cis-/trans- stereo isomers, and ring-strain, are often added to GA schemes as corrections to improve its accuracy. Additionally, heterogeneous catalysts also require surface atoms to be included in the groups, as shown in Figure 2. pGrAdd assigns the number of connected surface atoms based on the number of free valence electrons. For example, in Figure 2, each carbon has a valency of 4 each with 3 single bonds and, therefore, binds atop to the same Pt atom. A larger molecule could bind to different/multiple Pt atoms. Corrections to the valency rule will be reported elsewhere.



regressed to 'g' groups.

The OLS regression results in group value variance. The unbiased variance for the GA scheme is computed as shown in Eq. 1.

$$\hat{\sigma}^2 = \frac{\| (I_{N_T} - H_T) Y_T \|^2}{m - g - 1}; H_T = X_T (X_T' X_T)^{-1} X_T' \quad \text{Eq. 1}$$

Here  $\hat{\sigma}^2$  is the unbiased model variance estimator,  $Y_T$  is the training set vector of thermodynamic values,  $X_T$  is the training set configuration matrix, and  $(m - g - 1)$  is the model degrees-of-freedom (DOF) ( $m$  = number of molecules,  $g$  = number of groups, 1 = additional DOF to account for computing the intercept). A better metric of the model's predictive ability is computed via cross-validation, eliminating the bias from a non-independent testing set. For example, in 10-fold cross-validation, 90% of the original training molecules are used to train the model while the remaining 10% to validate the model's performance. The group values computed from the 90% are used to predict the thermodynamic properties of the remaining 10% validation set. These values are compared to their known values, and a mean absolute error is calculated. This process is repeated many times, each using a different combinatorial training and validation set and the mean value for all trials representing the predictive variance of the model.

### Program Features

A GA database minimally contains three files: library.yaml: Contains a list of group contribution value files. While at least one is required, it can be useful to separate different types of group contributions in separate files.

scheme.yaml: It contains three sections.

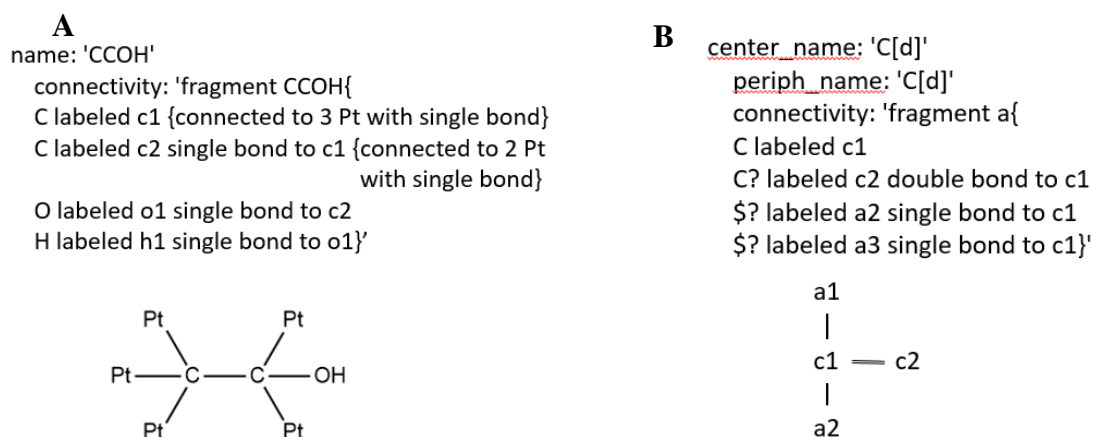
1. patterns: Ruleset used to parse molecules into an adjacency list discovering atom interconnectivity and their role as a central atom and/or peripheral atom.
2. remaps: Identifies groups with different notation but equivalent contribution.
3. other\_descriptors: Rulesets to identify corrections. Corrections do not follow graph theory defined groups but have sufficient rulesets to describe a molecule configuration that requires a correction.

<group contribution>.yaml: One or more files containing the group thermodynamic contribution values identified by a group SMILES string or correction pattern name. File names must appear in the library.yaml file. Each contribution file contains two sections: group: Molecule group contributions; other\_descriptors: Correction contributions.

### Scheme Files/Group Value Files

Scheme files coupled with group value files provide the database that pGrAdd employs to convert an **input** molecule SMILES string (rdkit is used to convert SMILES strings to parsable molecule objects) to a collection of groups and corrections. **Our parser couples** rulesets **with** the Silver extensible attribute grammar system[29] implemented similarly to the ruleset used in the Rule-based Reaction Network Generator (RING)[30]. **First**, the rulesets describe the connectivity

between atoms in a molecule, as in Figure 4, allowing our parser to identify the groups present in the molecule. pGrAdd considers only a central atom and its first nearest neighbors when generating groups. Second, additional rulesets define corrections by fully describing the atom connectivity pattern required to define an applicable correction. These groups, corrections, and their associated thermodynamic contribution values are in the group values file. Scheme and group value files are user-definable and selectable with six preconfigured databases included with pGrAdd.



**Figure 4: Examples of Silver language rulesets used to identify fragments (an atom's connectivity that will become the basis of a group) and correction groups in the pGrAdd scheme file.** (A) Correction rule identifying the presence of a CCOH group. (B) Molecule rule identifying a carbon (c1) double bonded to a second carbon (c2) and single bonded to two other undefined atoms (a1 and a2). c1 is identified as both a central and peripheral atom named C[d].

### Molecule Parser

The parser uses the ruleset in the scheme file to create an adjacency list (describing the connectivity of the atoms in the molecule) and identify their role as central and/or peripheral atoms in a group. The parser's ability to define groups is limited to the atoms/corrections prepopulated in the scheme file and the groups defined in the group value file. Additionally, the graph theory used by pGrAdd considers only central atoms and first-nearest neighbors. Hydrogen is not considered a central atom.

### Thermodynamic Estimator

The thermodynamic estimator takes the results of the molecule parser (groups and frequency of occurrence) and sums the contribution values to determine the thermodynamic value for a molecule. The estimator object includes the methods in Table 2 with available units in Table 3. The group values file and scheme file are in YAML format (a human readable markup language). A typical entry for a group in the groups value file follows:

```
groups:
  'C(C) (CO) (H) (O) ':
    'thermochem':
      T_ref: 298 K
      H_ref: -5.073 kcal/mol
      S_ref: 1.094 cal/(mol*K)
      Cp_data:
        - [100 K, -0.035 cal/(mol*K)]
        - [200 K, 2.118 cal/(mol*K)]
```

```

- [300 K, 3.619 cal/(mol*K)]
- [400 K, 4.934 cal/(mol*K)]
- [500 K, 5.944 cal/(mol*K)]
- [600 K, 6.782 cal/(mol*K)]
- [700 K, 7.440 cal/(mol*K)]
- [800 K, 7.971 cal/(mol*K)]
- [900 K, 8.413 cal/(mol*K)]
- [1000 K, 8.792 cal/(mol*K)]
- [1100 K, 9.127 cal/(mol*K)]
- [1200 K, 9.428 cal/(mol*K)]
- [1300 K, 9.695 cal/(mol*K)]
- [1400 K, 9.918 cal/(mol*K)]
- [1500 K, 10.079 cal/(mol*K)]
range: [100K, 1500K]

```

The entry consists of the SMILES string of the group, the reference temperature, and the enthalpy and entropy contribution at that reference temperature, the constant pressure heat capacity contribution at various temperatures, and the Cp temperature range over which the data has been regressed. Energy units are listed with each entry.

**Table 2: Methods included in pGrAdd's thermodynamic estimator object. Units are defined in pGrAdd's constants file (See Table 3 for a current list).**

Methods	Description
get_CpOR(T)	Dimensionless constant pressure heat capacity at temperature = T [K]
get_HoRT(T)	Dimensionless enthalpy at temperature = T [K]
get_SoR(T)	Dimensionless entropy at temperature = T [K]
get_GoRT(T)	Dimensionless Gibb's free energy at temperature = T [K]
get_Cp(T, units = )	Constant pressure heat capacity at temperature = T [K] and units =
get_H(T, units = )	Enthalpy at temperature = T [K] and units =
get_S(T, units = )	Entropy at temperature = T [K] and units =
get_G(T, units = )	Gibb's free energy at temperature = T [K] and units =

These methods are consistent with the thermodynamic methods used in pMuTT, allowing easy integration of the two. An empirical object (NASA or Shomate polynomial) can be created in pGrAdd and used interchangeably in pMuTT.

**Table 3: Current list of available units in pMuTT v 1.2.21. Omit the /K for enthalpy and Gibb's free energy.**

Units	Description
J/mol/K	Joule per mole per kelvin
kJ/mol/K	kiloJoule per mole per kelvin
L kPa/mol/K	Liter kilopascal per mole per kelvin
cm <sup>3</sup> kPa/mol/K	Cubic centimeter kilopascal per mole per kelvin
m <sup>3</sup> Pa/mol/K	Cubic meter pascal per mole per kelvin
cm <sup>3</sup> MPa/mol/K	Cubic centimeter megapascal per mole per kelvin
m <sup>3</sup> bar/mol/K	Cubic meters bar per mole per kelvin
L bar/mol/K	Liter bar per mole per kelvin

L torr/mol/K	Liter torr per mole per kelvin
cal/mol/K	Calorie per mole per kelvin
kcal/mol/K	Kilocalorie per mole per kelvin
L atm/mol/K	Liter atmosphere per mole per kelvin
cm <sup>3</sup> atm/mol/K	Cubic centimeter atmosphere per mole per kelvin
eV/K	Electron volt per molecule per kelvin
Eh/K	Hartree per molecule per kelvin
Ha/K	Hartree per molecule per kelvin

---

### Uncertainty quantification (UQ) Data

Four of the six included GA databases (see Table 4) also have an additional UQ file that contains the variance for each thermodynamic value (heat capacity is given at a number of temperatures, and enthalpy and entropy at the listed reference temperature), a list of groups in the scheme, an inverse covariance matrix, and the total degrees of freedom (DOF).

We demonstrate pGrAdd v2.2 providing examples from the Benson gas GA scheme and the Vlachos Group Pt(111) heterogeneous catalysis scheme. We compare the Benson results to NIST experimental data and the Vlachos results to DFT data for the same species.

### Installation

The pGrAdd Python libraries installation package is maintained on both Github and PyPi.org (Python Package Index). PyPi is streamlined for simple installation. To install pGrAdd use the pip Python package installer.

```
pip install pgradd
```

### **Application Case 1: Gas-phase Thermochemistry (Propanol)**

The first example is propanol using the Benson gas GA[8,9] database (scheme.yaml and group contributions files) which we then compare to NIST data

### **Python code**

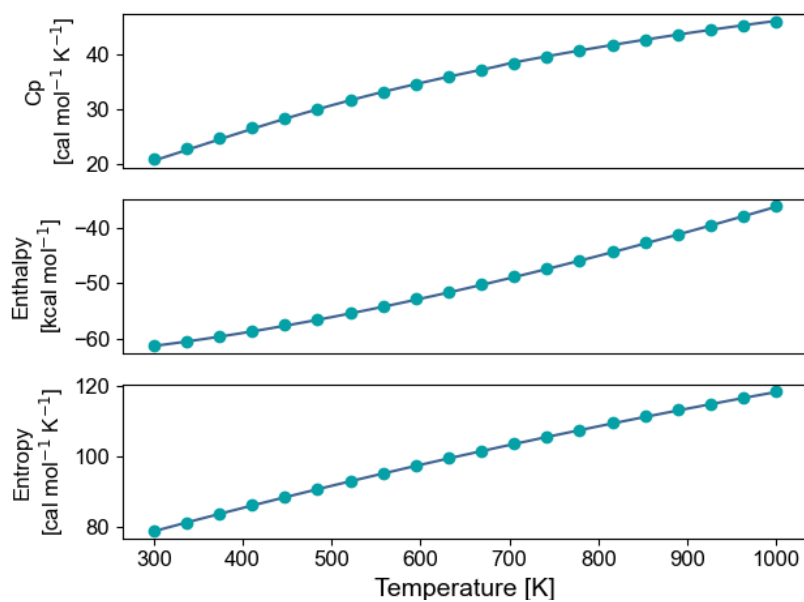
<pre>from pgradd.GroupAdd.Library import GroupLibrary import pgradd.ThermoChem  lib = GroupLibrary.Load('BensonGA')  descriptors = lib.GetDescriptors('CCCO')  print(descriptors)  therm = lib.Estimate(descriptors, 'thermochem') H = therm.get_H(T=500, units='kcal/mol') print('H(500 K) = {0:5.2f} [kcal/mol]'.format(H))</pre>	<p>Import pGrAdd libraries</p> <p>Specify GA database (BensonGA)</p> <p>Determine and print groups for selected molecule SMILES (CCCO)</p> <p>Initialize thermodynamic estimator Call enthalpy method and print results</p>
---	---

### **Results for Case 1**

```
defaultdict(<class 'int'>, {'C(C)(H)3': 1, 'C(C)2(H)2': 1, 'C(C)(H)2(O)': 1, 'O(C)(H)': 1})
```

$H(500\text{ K}) = -55.89\text{ [kcal/mol]}$

pGrAdd parsed the SMILES for propanol (CCCO) into 4 groups listed in the above results along with their frequency of occurrence. The format of the group names (first atom [central] without parentheses followed by atoms in parentheses) indicates these are all molecule groups and not corrections. The absence of corrections indicates that no Benson correction patterns were found in this molecule. Calling the get\_H method, with temperature and units specified, resulted in the enthalpy of -55.89 kcal/mol at 500 K. The above code can be repeated at various temperatures. The results are plotted vs. the same data from NIST. Figure 5 shows the comparison of heat capacity, enthalpy, and entropy. The GA and NIST values agree very well with a mean absolute error (MAE) of 0.12 [cal/mol/K], 0.10 [kcal/mol], and 0.12 [cal/mol/K] in heat capacity, enthalpy, and entropy, respectively. This is consistent with the overall MAE reported by Benson of 0.5 cal/mol/K for heat capacity and entropy and 0.6 kcal/mol for enthalpy.



**Figure 5: Thermodynamic properties (heat capacity, enthalpy, and entropy) for propanol gas as reported by NIST (—) and computed via the Benson GA scheme (●).**

### Application Case 2: Surface Species on Pt(111)

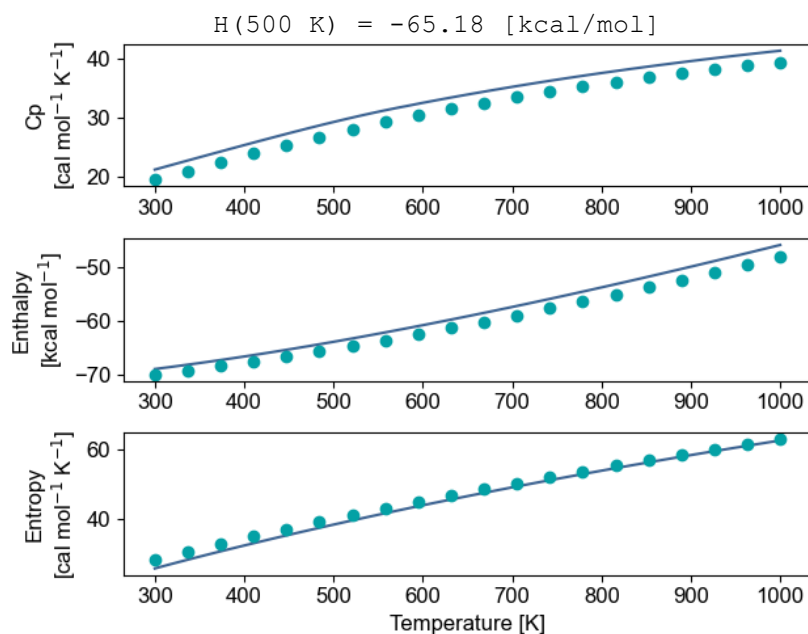
The second example uses a GA scheme for surface species on Pt(111) parameterized with 164 training molecules (GRWSurface2018). The thermochemistry was determined via DFT calculations processed via pMuTT's statistical mechanics models. The data was regressed to 66 groups.

## Python Code

<pre>from pgradd.GroupAdd.Library import GroupLibrary import pgradd.ThermoChem  lib = GroupLibrary.Load('GRWSurface2018')  descriptors = lib.GetDescriptors('CCO') print(descriptors)  therm = lib.Estimate(descriptors, 'thermochem') H = therm.get_H(T=500, units='kcal/mol') print('H(500 K) = {0:5.2f} [kcal/mol]'.format(H))</pre>	<p>Import pGrAdd libraries</p> <p>Specify GA database (GRWSurface2018)</p> <p>Determine and print groups for selected molecule SMILES (CCO)</p> <p>Initialize thermodynamic estimator</p> <p>Call enthalpy method and print results</p>
---	---

## Results for Case 2

```
defaultdict(<class 'int'>, {'C(C)(H)3': 1, 'C(C)(H)2(O)': 1, 'O(C)(H)': 1, 'CC': 1})
```



**Figure 6: Thermodynamic properties (heat capacity, enthalpy, and entropy) for ethanol chemisorbed on a Pt(111) surface as computed from DFT (—) and computed via the surface GA scheme-GRWSurface2018 (●).**

pGrAdd parsed the SMILES for ethanol physisorbed on Pt(111) (CCO) into 4 groups along with their frequency of occurrence. The name pattern of the first three groups indicates these are molecule groups while the fourth group (not using the central atom/neighbor pattern) is a correction accounting for the nonlinearity arising from the increased length of the alkane chain. Calling the `get_H` method with temperature and units computes an enthalpy (-65.18 kcal/mol at 500 K). The above code is repeated at various temperatures and the results are plotted vs. DFT data of our group. Figure 6 compares the heat capacity, enthalpy, and entropy. The values agree very well with a mean absolute error (MAE) of 1.87 [cal/mol/K], 1.61 [kcal/mol], and 1.29 [cal/mol/K] respectively. This is consistent with the overall GA scheme MAE we report of 0.23

[cal/mol/K] for heat capacity, 4.53 [kcal/mol] for enthalpy, and 2.07 [cal/mol/K] for entropy in this work.

**Table 4: GA databases included with pGrAdd**

Name	Details	Reference
BensonGA	Subset of the original Benson gas GA scheme Includes correction files for all 256 groups Benson identified but implements only hydrocarbons	[8,9]
SaliccioliGA2012	Adsorbates on Pt(111) 75 groups, <b>support for [C.] radicals</b>	[31]
GuSolventGA2017Aq, GuSolventGA2017Vac	Adsorbates on Pt(111) 75 groups. Species include C, H, and O. Trained on 200 molecules (163 adsorbates, 37 gas molecules) Vac: Adsorbates w/o solvation effects Aq: Adsorbates with solvation effects Includes model uncertainty data in uq.yaml file <b>Support for double/triple carbon bonds not bonded to the surface, no support for radicals.</b>	[22,32]
GRWSurface2018, GRWAqueous2018	Adsorbates on Pt(111): Subset of GuSolventGA2017 datasets excludes most gas training molecules to reduce OLS fit error. 66 groups; Species include C, H, and O. <b>No support for double/triple carbon bonds not bonded to the surface; no support for radicals.</b> Trained on 164 molecules (163 adsorbates, 1 gas molecule) including chemisorbed, physisorbed, and hybrid (chemisorbed with functional groups extending above the active surface) species. Surface: Adsorbates w/o solvation effects Aqueous: Adsorbates with solvation effects Includes model uncertainty data in uq.yaml file	[22,32]

### pMuTT Integration and Multiple Molecules

Parameterizing a kinetic model often requires thermodynamic data for many molecules and output in a specific format. In this example, we demonstrate how multiple molecules can be parsed, thermodynamic properties estimated, temperature-dependent thermodynamic properties expressed as NASA polynomials, and results output in the commonly used thermdat format.

#### Python Code

```
from pgradd.GroupAdd.Library import GroupLibrary
import pgradd.ThermoChem

lib = GroupLibrary.Load('GRWSurface2018')
```

```
Import pGrAdd libraries interactively or into the
user's code
Specify GA database (GRWSurface2018) for
adsorbates
```

<pre> from pmutt.empirical.nasa import Nasa from pmutt.io.thermdat import write_thermdat from pmutt import parse_formula  molecules = ['CH4', 'C2H4', 'CH3OH', 'C3H8', 'CH2COH'] SMILES = ['C', 'C[Pt]C[Pt]', 'CO', 'CCC',           'C(C[Pt])([Pt])([Pt])O']  nasa_obj = [] T_ref = 298.15 T = np.linspace(300, 1500, 13) for smile, mol in zip(SMILES, molecules):     descriptors = lib.GetDescriptors(smile)     thermochem = lib.Estimate(descriptors, 'thermochem')     elements = parse_formula(mol)     nasa_obj.append(Nasa.from_data(name=mol, T=T,                                   CpoR=thermochem.get_CpoR(T),                                   T_ref=T_ref,  HoRT_ref=thermochem.get_HoRT(T_ref),     SoR_ref=thermochem.get_SoR(T_ref),     elements=elements, phase='S'))  write_thermdat(nasa_species=nasa_obj, write_date=True) </pre>	<p>Import pMuTT libraries that build NASA empirical objects, write thermdat files, and determine elements in a molecular formula</p> <p>List of molecule names and SMILES to parameterize.</p> <p>This code parses the list of molecules, determines the elements in the molecules, and generates a NASA empirical object.</p> <p>Use NASA object, containing data for the target molecules, and convert to thermdat format</p>
--	---

## Results

```

THERMO ALL
      100      500      1500
CH4      20210707C      1H      4      S300.0      1500.0      800.0      1
3.99950592E+00 6.85458298E-03 2.59863036E-06-3.80082810E-09 1.01025020E-12
-1.21094055E+04-1.04509843E+01 7.47986468E+00-1.87055076E-02 6.98625180E-05
-8.00188047E-08 3.26043955E-11-1.24600001E+04-2.50195754E+01
C2H4      20210707C      2H      4      S300.0      1500.0      800.0      1
9.84152520E-01 3.15964000E-02-2.81257467E-05 1.40901310E-08-2.93544397E-12
-9.37882132E+03-5.93768291E+00-2.16253558E+00 4.89601235E-02-6.34027940E-05
4.52524314E-08-1.29998233E-11-8.92868669E+03 8.20660811E+00
CH3OH      20210707C      1H      4O      1      S300.0      1500.0      800.0      1
1.68494919E+00 2.20534568E-02-1.57841109E-05 6.46178899E-09-1.16274079E-12
-3.01345605E+04-4.87882167E+00 4.36862356E+00 3.54609619E-03 3.08074844E-05
-4.46327266E-08 1.94997349E-11-3.04328345E+04-1.63172717E+01
C3H8      20210707C      3H      8      S300.0      1500.0      800.0      1
-2.04151360E+00 5.06149690E-02-3.85057815E-05 1.65553957E-08-3.08793456E-12
-2.29570046E+04 1.38342718E+01 3.87558879E+00 1.43534756E-02 4.46131570E-05
-6.78600094E-08 2.89350905E-11-2.37271643E+04-1.21803878E+01
CH2COH      20210707C      2H      3O      1      S300.0      1500.0      800.0      1
8.77826631E-01 3.08431553E-02-2.87086771E-05 1.39556852E-08-2.73827363E-12
-3.01612087E+04-6.22995990E+00-1.42120826E+00 4.08708183E-02-4.23781323E-05
1.86816127E-08-1.25637002E-12-2.97789657E+04 4.53198567E+00
END

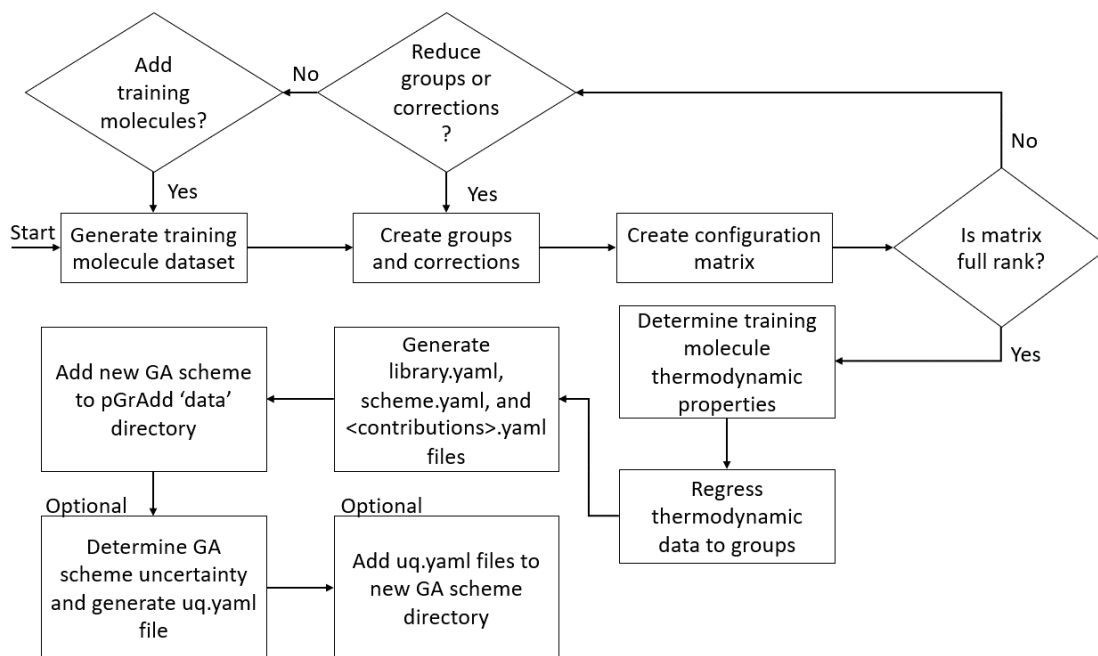
```

The example shows five molecules and associated SMILES used as input, but there is no limit to the number of molecules. The molecule and SMILES list could also have been read from a spreadsheet instead of explicitly stated as a list variable in Python. A list of dimensionless heat capacities and associated temperatures, dimensionless enthalpy and entropy at a reference temperature, a dictionary of elements, and the species phase ('S', surface, is specified for all given these are adsorbates) are input to pMuTT's NASA Python object. pMuTT converts these

to a two-temperature range NASA polynomial (a low and high-temperature range polynomial bounded by the  $C_p$  temperature range and optimized mid-point computed by pMuTT). Finally, pMuTT's write\_thermdat method converts the NASA objects to the thermdat format and writes to a file or standard output.

## Custom GA Database

pGrAdd does not provide any functionality to create a custom GA database but can implement a new database once it is created. Nonetheless, a custom GA database can be created for pGrAdd using the flowchart in Figure 7:



**Figure 7: Flowchart outlining the key steps in creating a customized GA database.**

1. Create a set of training molecules. The total number must exceed the total number of groups plus corrections (to, minimally, form a square matrix) but, ultimately, must meet the matrix rank requirements outlined in step 3.
2. Generate a list of groups and corrections to be used in the GA. This includes all central atom and nearest neighbor groups and any corrections.
3. Build a configuration matrix (as shown in Figure 3) and determine if the matrix is full rank. Determine if there are a sufficient number of linearly independent training molecules to create a unique regression to the desired number of groups plus corrections. If not, remedy by adding training molecules or reducing the number of groups. Adding training molecules requires the least effort, but the molecule that solves the rank deficiency can be elusive. Reducing the dimensionality will usually work but can increase model variance due to information loss (minimized by selecting groups with the lowest influence, as in Lasso[25]) or increase complexity by shifting the model basis

from physically meaningful groups to mathematically relevant principal components (as in PCA).

4. Determine the training molecules' thermodynamic properties ( $C_p$  at various temperatures and enthalpy and entropy at a reference temperature). Using  $C_p(T)$  we can integrate the relationships in Eq. 2 and apply the enthalpy and entropy values as integration constants to yield  $H(T)$  and  $S(T)$ :

$$C_p = \left( \frac{\partial H}{\partial T} \right)_p \quad C_p = T \left( \frac{\partial S}{\partial T} \right)_p \quad \text{Eq. 2}$$

5. Regress the thermodynamic properties of the groups to those of the training set.  $C_p$  at each temperature, enthalpy, and entropy must be regressed.
6. Create the three required pGrAdd files: library.yaml (List of group and correction contribution values), scheme.yaml (rulesets required to parse for groups and corrections), <group contribution>.yaml (one or more of these as listed in the library.yaml file) containing the group contributions and corrections.
7. Deposit these files in a directory under the pGrAdd 'data' directory with a unique directory name for the GA scheme.
8. [Optional] Use cross-validation to compute the model uncertainty and add a uq.yaml file to the GA scheme.

Thermodynamic properties can be generated from experimental or DFT data. It is important to verify that the configuration matrix is full rank (matrix rank equals the total number of groups plus corrections) before performing the OLS regression (Figure 3). If the group scheme follows the same graph theory (central atom and all first nearest neighbors) and uses no additional corrections than the current pGrAdd GA schemes, then one of the existing scheme.yaml files can be used. Otherwise, a new scheme file will need to be written and possible code modifications could be required.

## Open Source

pGrAdd is open source and welcomes both community development and feedback to improve the software continuously. We use GitHub as a version repository and encourage users to leave feedback via the 'Issues' page or fork a copy of our code to contribute additions or fixes to the existing code.

## Conclusions

Thermochemical and kinetic parameters are key inputs to MKM's. Complex heterogeneous reaction mechanisms have a large number of adsorbates and reactions requiring a commensurately large number of thermochemical parameters. These are computationally costly to compute via DFT. pGrAdd presents a simple, lightweight Python toolbox to parameterize thermochemical data for MKMs without costly DFT. We include 6 GA databases for gas and adsorbates on Pt(111), enabling users to immediately generate thermochemical data but also allow for users to generate a custom GA scheme using their own data.

Development on pGrAdd is ongoing with plans to add GA databases on other transition metals, thermodynamic corrections for gas DFT computed thermochemistry, and improved integration with pMuTT.

### **Acknowledgments**

The authors acknowledge support from the Department of Energy's Office of Energy Efficient and Renewable Energy's Advanced Manufacturing Office under Award Number DE-EE0007888-9.5. The authors thank Dr. Jeffrey Frey for the development of the pip installation package, Dr. Geun Ho Gu for coding contributions and GA databases, and all those that contributed to the code in the past.

## References

- [1] A.H. Motagamwala, J.A. Dumesic, *Microkinetic Modeling: A Tool for Rational Catalyst Design* *Chem. Rev.* 121 (2021) 1049–1076 <https://pubs.acs.org/doi/10.1021/acs.chemrev.0c00394>.
- [2] V. Prasad, D.G. Vlachos, *Multiscale Model and Informatics-Based Optimal Design of Experiments: Application to the Catalytic Decomposition of Ammonia on Ruthenium Ind.* *Eng. Chem. Res.* 47 (2008) 6555–6567 <https://pubs.acs.org/doi/10.1021/ie800343s>.
- [3] V. Prasad et al., *High throughput multiscale modeling for design of experiments, catalysts, and reactors: Application to hydrogen production from ammonia* *Chem. Eng. Sci.* 65 (2010) 240–246 <http://dx.doi.org/10.1016/j.ces.2009.05.054>.
- [4] T. Bligaard et al., *The Bronsted-Evans-Polanyi relation and the volcano curve in heterogeneous catalysis* *J. Catal.* 224 (2004) 206–217.
- [5] J.K. Nørskov et al., *Universality in heterogeneous catalysis* *J. Catal.* 209 (2002) 275–278.
- [6] S. Wang et al., *Universal Brønsted-Evans-Polanyi relations for C-C, C-O, C-N, N-O, N-N, and O-O dissociation reactions* *Catal. Letters* 141 (2011) 370–373.
- [7] A. Michaelides et al., *Identification of general linear relationships between activation energies and enthalpy changes for dissociation reactions at surfaces* *J. Am. Chem. Soc.* 125 (2003) 3704–3705.
- [8] S.W. Benson et al., *Additivity rules for the estimation of thermochemical properties* *Chem. Rev.* 69 (1969) 279–324 <http://pubs.acs.org/doi/abs/10.1021/cr60259a002>.
- [9] S.W. Benson et al., *Additivity rules for the estimation of molecular properties. Thermodynamic Properties* *Chem. Rev.* 69 (1969) 279–324.
- [10] J. Kua et al., *Thermochemistry for hydrocarbon intermediates chemisorbed on metal surfaces: CH(n-m)(CH3)(m) with n = 1, 2, 3 and m ≤ n on Pt, Ir, Os, Pd, Rh, and Ru J.* *Am. Chem. Soc.* 122 (2000) 2309–2321.
- [11] J. Lym et al., *A Python Multiscale Thermochemistry Toolbox (pMuTT) for thermochemical and kinetic parameter estimation* *Comput. Phys. Commun.* 247 (2020) 106864 <https://doi.org/10.1016/j.cpc.2019.106864>.
- [12] A. Larsen et al., *The Atomic Simulation Environment—A Python library for working with atoms* *J. Phys. Condens. Matter* 29 (2017) 273002.
- [13] M. Liu et al., *Reaction Mechanism Generator v3.0: Advances in Automatic Mechanism Generation* *J. Chem. Inf. Model.* 61 (2021) 2686–2696.
- [14] L. Chanussot et al., *Open Catalyst 2020 (OC20) Dataset and Community Challenges* *ACS Catal.* 11 (2021) 6059–6072.
- [15] G. Landrum, *RDKit: Open-Source Cheminformatics Software* (2008) <https://www.rdkit.org/>.
- [16] F. Pérez, B.E. Granger, *IPython: A system for interactive scientific computing* *Comput. Sci. Eng.* 9 (2007) 21–29.
- [17] C.R. Harris et al., *Array programming with NumPy* *Nature* 585 (2020) 357–362 <http://dx.doi.org/10.1038/s41586-020-2649-2>.
- [18] P. Virtanen et al., *SciPy 1.0: fundamental algorithms for scientific computing in Python* *Nat. Methods* 17 (2020) 261–272.
- [19] R. Penrose, *A generalized inverse for matrices* *Math. Proc. Cambridge Philos. Soc.* 51 (1955) 406–413.
- [20] A. Dresden, *The fourteenth western meeting of the american mathematical society* *Bull. Am. Math. Soc.* 26 (1920) 385–396.

- [21] V. Vorotnikov et al., Group additivity for estimating thermochemical properties of furanic compounds on Pd(111) *Ind. Eng. Chem. Res.* 53 (2014) 11929–11938.
- [22] G.H. Gu et al., Group additivity for aqueous phase thermochemical properties of alcohols on Pt(111) *J. Phys. Chem. C* 121 (2017) 21510–21519.
- [23] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space London, Edinburgh, Dublin *Philos. Mag. J. Sci.* 2 (1901) 559–572  
<https://doi.org/10.1080/14786440109462720>.
- [24] I.T. Jolliffe, J. Cadima, Principal component analysis: a review and recent developments *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* 374 (2016) 20150202  
<https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202>.
- [25] J. Lever et al., Points of Significance: Principal component analysis *Nat. Methods* 14 (2017) 641–642 <http://dx.doi.org/10.1038/nmeth.4346>.
- [26] Q. Li et al., Accurate Thermochemistry of Complex Lignin Structures via Density Functional Theory, Group Additivity, and Machine Learning *ACS Sustain. Chem. Eng.* 9 (2021) 3043–3049.
- [27] G.H. Gu et al., Thermochemistry of gas-phase and surface species via LASSO-assisted subgraph selection *React. Chem. Eng.* 3 (2018) 454–466  
<http://xlink.rsc.org/?DOI=C7RE00210F>.
- [28] R.J. Tibshirani, The lasso problem and uniqueness *Electron. J. Stat.* 7 (2013) 1456–1490.
- [29] E. Van Wyk et al., Silver: An extensible attribute grammar system *Sci. Comput. Program.* 75 (2010) 39–54 <http://dx.doi.org/10.1016/j.scico.2009.07.004>.
- [30] S. Rangarajan et al., Language-oriented rule-based reaction network generation and analysis: Description of RING *Comput. Chem. Eng.* 45 (2012) 114–123  
<http://dx.doi.org/10.1016/j.compchemeng.2012.06.008>.
- [31] M. Saliccioli et al., Density functional theory-derived group additivity and linear scaling methods for prediction of oxygenate stability on metal catalysts: Adsorption of open-ring alcohol and polyol dehydrogenation intermediates on pt-based metals *J. Phys. Chem. C* 114 (2010) 20155–20166.
- [32] G.H. Gu, D.G. Vlachos, Group Additivity for Thermochemical Property Estimation of Lignin Monomers on Pt(111) *J. Phys. Chem. C* 120 (2016) 19234–19241.