Introduction to pycalphad 0.7

Richard Otis and Brandon Bocklund

Sunday, May 27, 2018

CALPHAD XLVII

pycalphad = Python + CALculation of PHAse Diagrams

- pycalphad is software for designing thermodynamic models, calculating phase diagrams and investigating phase equilibria.
- Using CALPHAD-based models, pycalphad predicts properties of materials, including
  - Transition (e.g., melt) temperatures, phase fractions, solidification, degradation, corrosion, etc.
  - Anything that can be connected to a chemical or thermodynamic process
- Free and open source at pycalphad.org

Notable Features in pycalphad 0.7

- State-of-the-art global energy minimization
- Binary and ternary phase diagram plotting
- T, P, x_i conditions with multiple components
- Step/map calculation
- Support for associates and ionic liquids
- IHJ and Xiong magnetic models
- Order-disorder model, Two-state model, Einstein model
- Windows, Mac, and Linux support

R. Otis, M. Emelianenko, Z. K. Liu, “An improved sampling strategy for global energy minimization of multi-component systems”, Comp. Mat. Sci. 130 (2017) 282–291.

Current Limitations in pycalphad 0.7

- In progress: “Advanced” conditions (e.g., amount of a phase)
  - Target release: August 2018
- First calculation of a session takes a minute to warm up the cache
  - Subsequent calculations are fast
- Not yet implemented: Quasichemical model, ternary excess models other than Muggianu (the standard for alloys)
- A few TDB features are unsupported (Option “B”, Option “F”, STATUS_BITS, etc…)
  - Contact us: we can usually help you work around the issue

IP[y]: Notebook BinaryExamples Last Checkpoint: Jan 08 23:25 (autosaved)

File Edit View Insert Cell Kernel Help

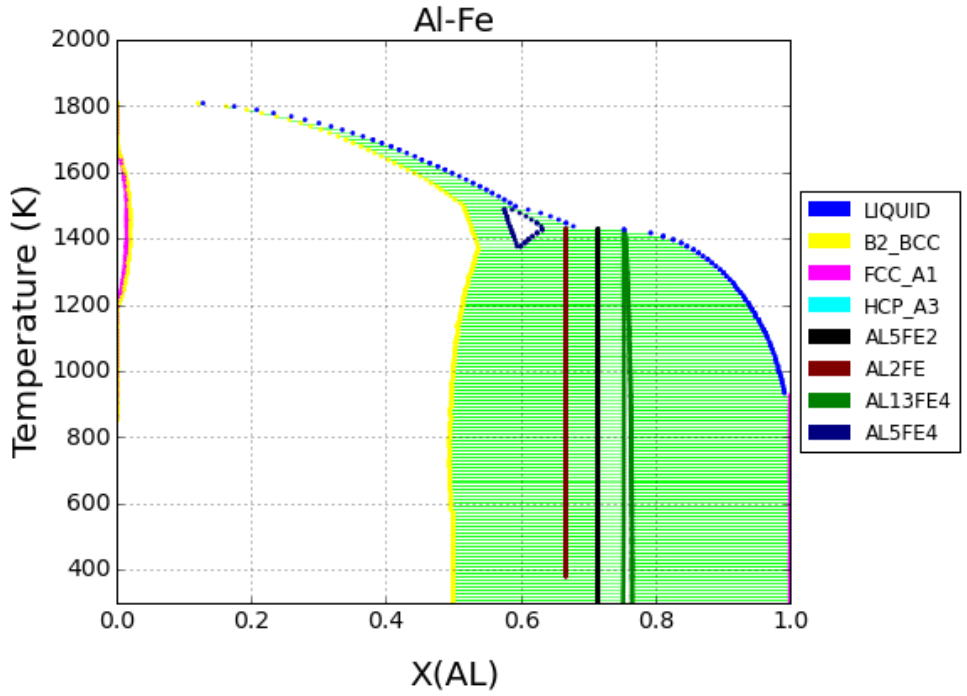
Heading 1 Cell Toolbar: None

Al-Fe (M.Seiersten et al., 1991)

In [3]:
```
db_alfe = Database('alfe_sei.TDB')
my_phases_alfe = ['LIQUID', 'B2_BCC', 'FCC_A1', 'HCP_A3', 'AL5FE2', 'AL2FE', 'AL13FE4', 'AL5FE4']

fig = plt.figure(figsize=(9,6))
pdens = [{'B2_BCC': 20000}, 2000]
%time ax = binplot(db_alfe, ['AL', 'FE', 'VA'] , my_phases_alfe, 'X(AL)', 300, 2000, pdens=pdens, ax=fig.gca())
plt.show()
```

```
CPU times: user 27.5 s, sys: 639 ms, total: 28.1 s
Wall time: 17.1 s
```

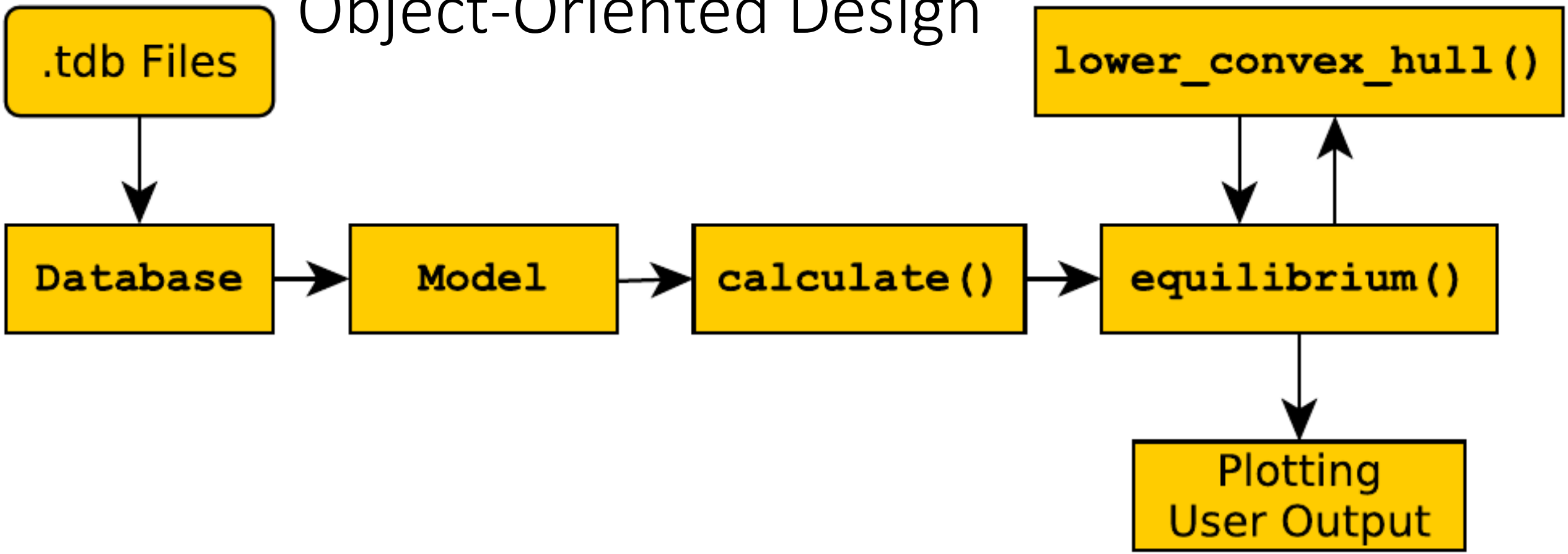


Why use pycalphad

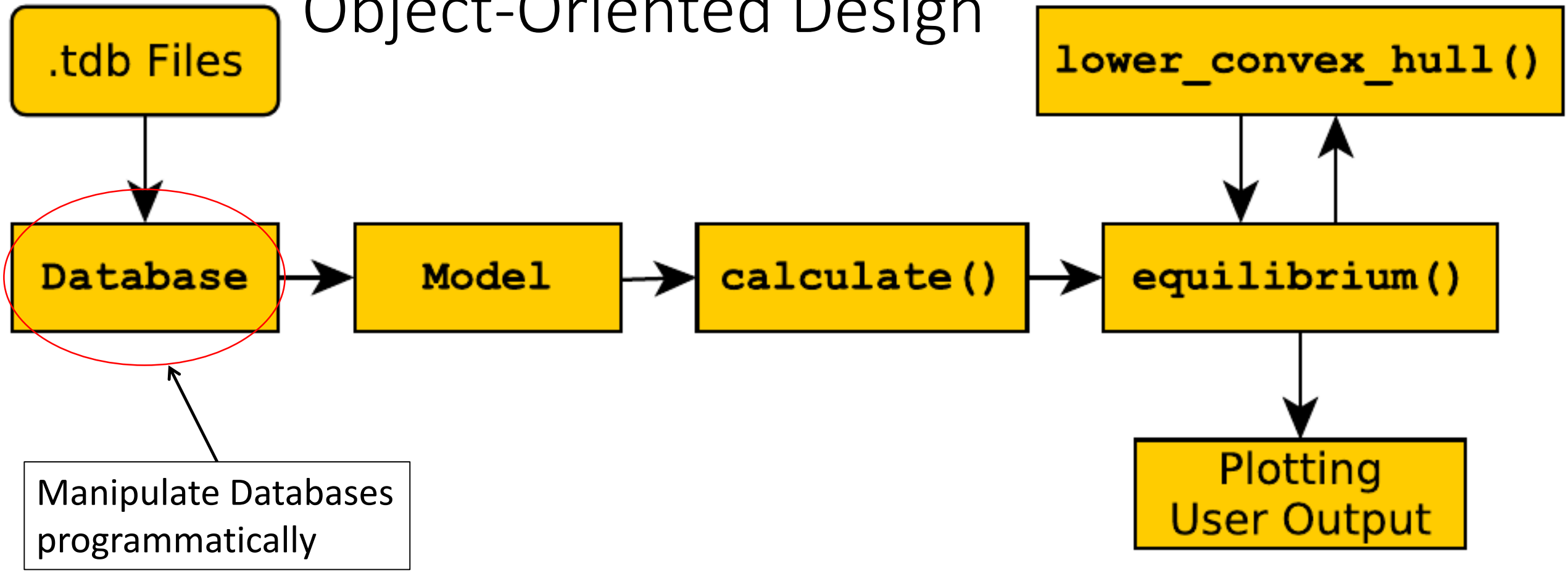
- Custom Gibbs energy models
- Custom material property models
- Digital notebooks (shareable, reproducible)
- Semi-automated database development (via ESPEI)

R. Otis and Z-K. Liu, “pycalphad: CALPHAD-based computational thermodynamics in Python”, *Journal of Open Research Software*, 5: 1 (2017)

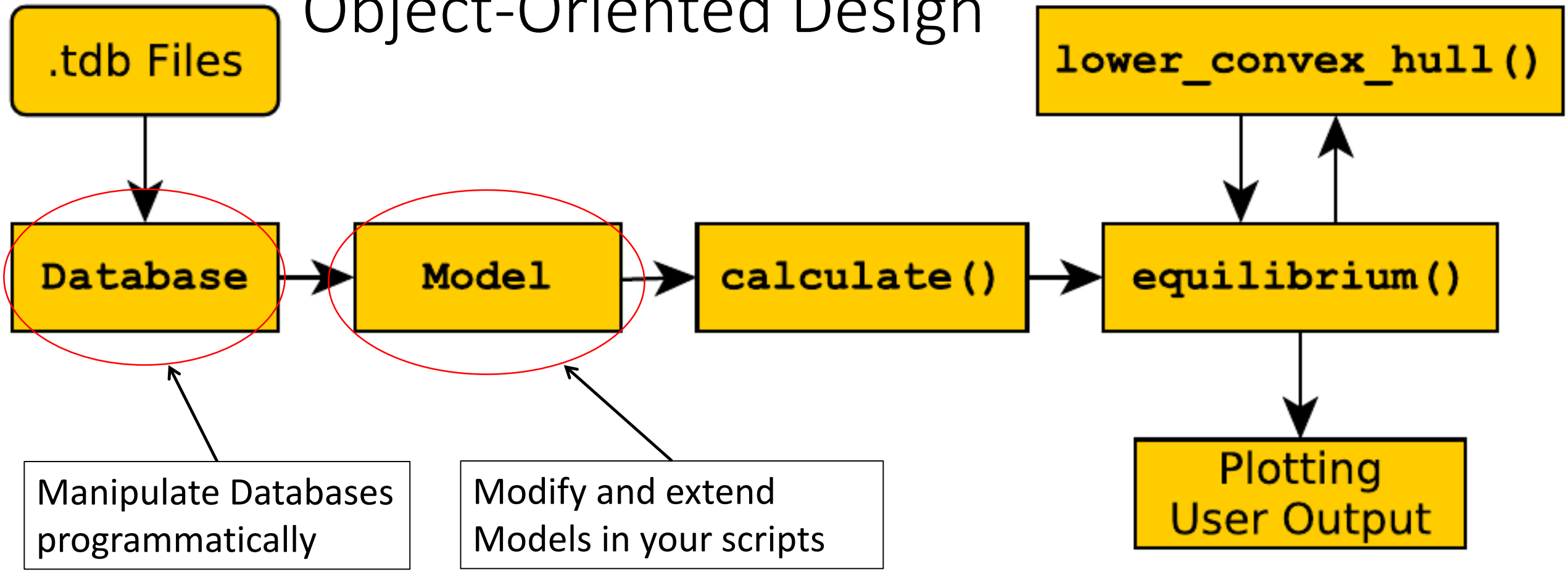
Object-Oriented Design



Object-Oriented Design



Object-Oriented Design



Custom Database parameter types

```
import pycalphad.io.tdb_keywords
pycalphad.io.tdb_keywords.TDB_PARAM_TYPES.extend(['THETA', 'THETAX', 'THETAY','THETAZ'])
```

```
thetax_param_query = (
    (where('phase_name') == phase.name) & \
    (where('parameter_type') == 'THETAX') & \
    (where('constituent_array').test(self._array_validity))
)
```

```
lntheta = self.redlich_kister_sum(phase, param_search, theta_param_query)
```

Defining custom Models

```
contributions = [('ref', 'reference_energy'), ('idmix', 'ideal_mixing_energy'),
                 ('xsmix', 'excess_mixing_energy'), ('mag', 'magnetic_energy'),
                 ('ein', 'einstein_energy'), ('2st', 'twostate_energy'),
                 ('ord', 'atomic_ordering_energy')]
```

```
def einstein_energy(self, dbe):
    phase = dbe.phases[self.phase_name]
    param_search = dbe.search
```

[function continues here]

```
einstein_cpm = property(lambda self: -v.T*self.models['ein'].diff(v.T, v.T))
einstein_gm = property(lambda self: self.models['ein'])
magnetic_gm = property(lambda self: self.models['mag'])
```

Project Sustainability

- pycalphad is developed as part of Richard’s job responsibilities
- Brandon has a fellowship supporting his development work
- We are looking to grow our team of collaborators!

Contact Us

richard.otis@outlook.com

pycalphad.org

Chat with us on Gitter: gitter.im/pycalphad/pycalphad