pypsbuilder Documentation

Release 2.4.2

Ondrej Lexa

Apr 22, 2024

CONTENTS

CHAPTER

ONE

INSTALLATION

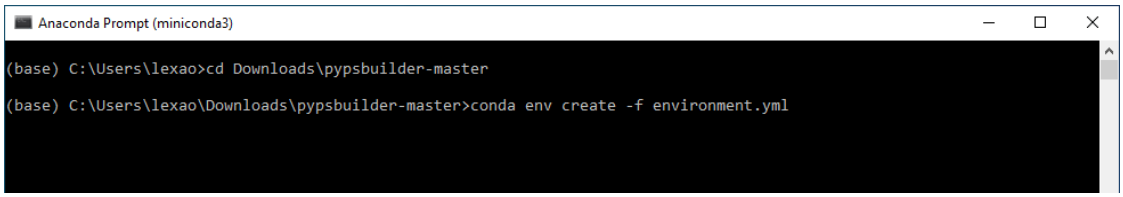
In the moment you need to have Python and needed modules installed locally and **pypsbuilder** must be installed from source. Just follow these steps:

1. Easiest way to install Python is to use Anaconda/ Miniconda/ Miniforge distribution. Download it and follow installation steps.

2. Download latest version of pypsbuilder and unzip to folder of your choice.

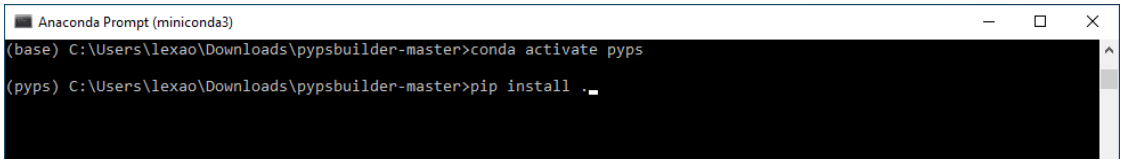
3. Use conda to create an environment from an `environment.yml` file. Open the terminal, change directory where you unzip the source and execute following command:

```
$ conda env create -f environment.yml
```



4. Activate the new environment and install from current directory:

```
$ conda activate pyps
(pyps) $ pip install .
```



Alternatively, you can use `pip` to install pypsbuilder:

```
(pyps) $ pip install pypsbuilder
```

1.1 Upgrade to latest version

You can anytime upgrade your existing pypsbuilder to the latest released version:

```
(pyps) $ pip install --upgrade pypsbuilder
```

or to latest master version at github:

```
(pyps) $ pip install --upgrade https://github.com/ondrolexa/pypsbuilder/archive/master.
↪zip
```

CHAPTER

TWO

PSEUDOSECTION BUILDERS TUTORIAL

The **pypsbuilder** provides three builders `ptbuilder`, `txbuilder` and `pxbuilder` to create P-T, T-X and P-X pseudosections.

Before you can successfully run any builder, you have to prepare working directory, which contain *THERMOCALC* and *drawpd* executables, preferences file, thermodynamic dataset and a-x file. Builders will work only with certain setting, like `calcmode` must be set to 1, no ask for scripts etc. Builders validate settings and gives warning if some action is needed.

The only special need is to place special tags for bulk composition, ptguesses and dogmin in your scriptfile, to manage starting guesses and dogmin runs.

You have to add following comment lines to your script file to line where normally starting guesses should be placed (definitely before last * and before standard or samecoding guesses).:

```
%{PSBGUESS-BEGIN}
%{PSBGUESS-END}
```

2.1 Scriptfile modifications for TC3.4

For older version of THERMOCALC you need add two other commented blocks. For dogmin replace existing `dogmin` script with:

```
%{PSBDOGMIN-BEGIN}
dogmin no
%{PSBDOGMIN-END}
```

and for bulk composition place before and after existing `setbulk` script(s) these tags:

```
%{PSBBULK-BEGIN}
setbulk ....
%{PSBBULK-END}
```

2.2 Scriptfile modifications for TC3.5

For latest THERMOCALC the tags are slightly different. You should enclosed calculation part as:

```
%{PSBCALC-BEGIN}
calcP 4 12
calcT 600 1050 9
calctatp no
with  plc q aug hb sph g ep
zeromodeisopleth g
%{PSBCALC-END}
```

and for bulk composition place before and after existing `bulk` scripts these tags:

```
%{PSBBULK-BEGIN}
bulk  H2O   SiO2   Al2O3 CaO    MgO    FeOt  K2O   Na2O  TiO2  O
bulk  4.781 50.052 9.106 12.391 11.192 8.943 0.139 2.078 0.705 0.612
%{PSBBULK-END}
```

If you are not sure, which scripts should be set on and off, you can check example scriptfiles in `examples/avgpelite` or `examples/avgpelite_34` directory.

2.3 Using tcinit to setup working directory

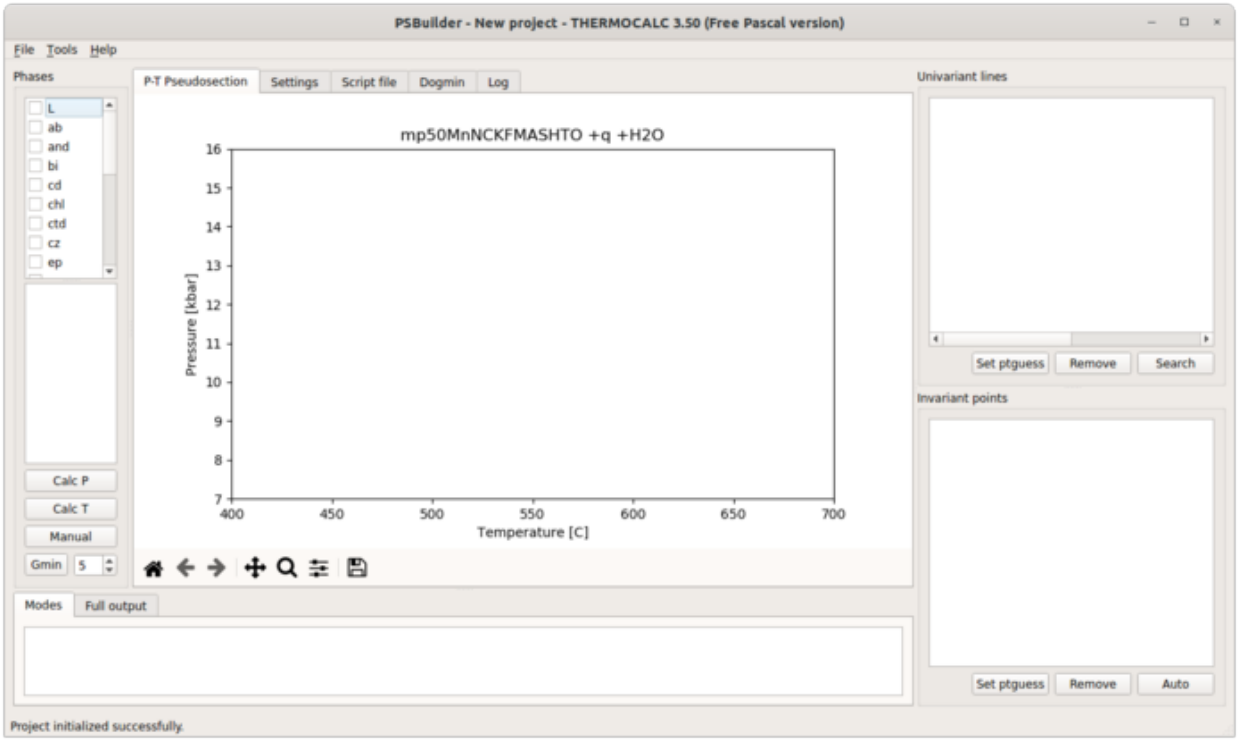
For TC3.5 you can use `tcinit` script to initialize working directory. It will download the latest 3.5 version of THERMOCALC, appropriate dataset and a-x file according to user selection.

2.4 New P-T pseudosection project

Use the terminal or an Anaconda Prompt, activate the `pyps` environment and run `ptbuilder`:

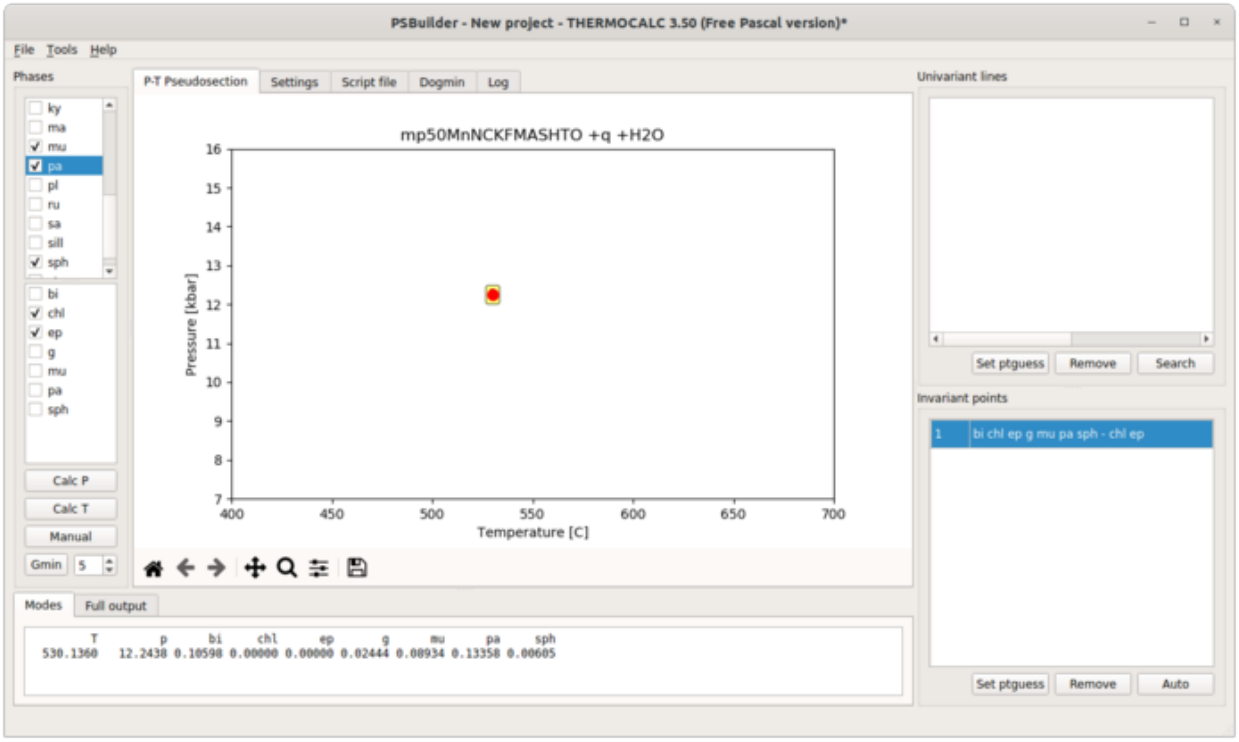
```
$ conda activate pyps
(pyps) $ ptbuilder
```

To create the new project (File->New project), you have to select working directory. `ptbuilder` automatically execute *THERMOCALC*, check settings in your script file and initialize project. Available phases are automatically populated to *Phases* list and default P-T range from scriptfile is set.



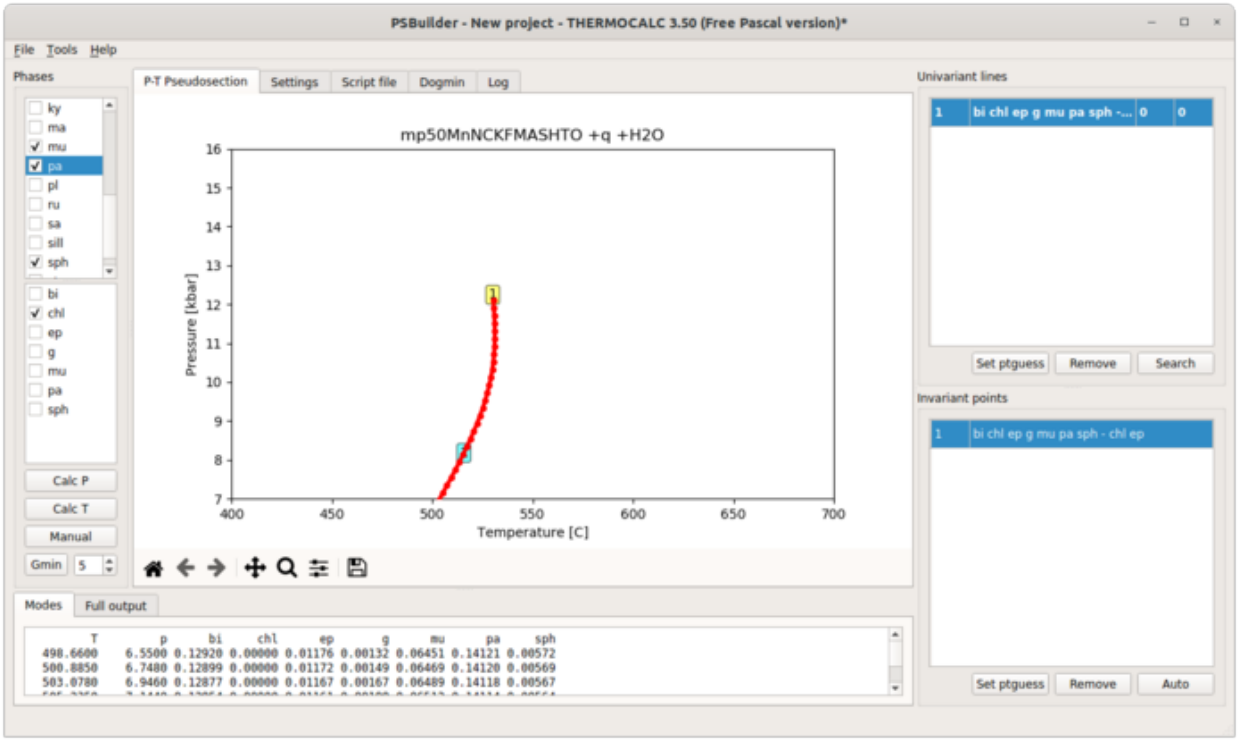
2.5 Create invariant point

In *Phases* list you select phases which should be in stable assemblage, while in lower pane you select two phases for which modal proportion should be zero. Than just click either `Calc P` or `Calc T` and invariant point will appear on diagram and in the list of invariant points in lower right part of window.



2.6 Create univariant line

Similarly, you can create univariant line, when only one phase is selected to have zero modal proportion. In addition `ptbuilder` allows you to create univariant lines based on already calculated invariant points. Right-click on invariant points will show context menu with possible choices of univariant lines passing trough this point and which are not yet calculated. Note, that selecting offered univariant line phases from context menu, the starting guesses from invariant point will be used in subsequent calculation. Hit **Calc T** or **Calc P** according to dp/dT of univariant line. Once calculated, result is added to diagram and to the list of univariant lines in upper right part of the window. Within this list you can define begin and end by selecting appropriate invariant points. If you allow autoconnection on *Settings* pane and both invariants points are already calculated, the begin and end is set automatically.

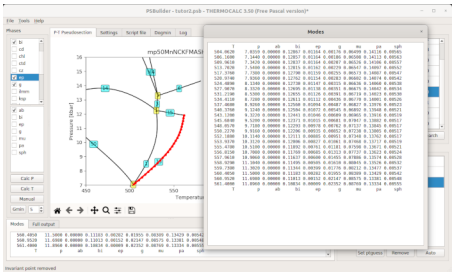


By default, `ptbuilder` use 50 steps to calculate univariant lines. You can change it in *Settings* pane. When you need to calculate some short univariant lines you can zoom into this part of pseudosection and hit one of the `Calc` buttons. Active region (possibly extended, check *Extend view range to calculation range* setting) will be used as computational P-T range. Moreover, you can manually add univariant line to simply connect two invariant points by straight line. For ''Manual" addition of both invariant point or univariant line present phases and zero mode phases have to be properly selected. Manually added lines or points are shown in italics in lists. Unconnected univariant lines are shown in bold.

When univariant line cannot be calculated by single THERMOCALC calculation (either due to starting guesses or high curvature), you can merge partially calculated segments using ''+" button before calculation. Instead of overwriting of existing line, newly calculated part will be merged with existing one. To remove selected part of already calculated line, you can zoom to that part and right-click highlighted name and choose ''Remove nodes".

Double-clicking any univariant line or invariant point in the list will highlight that line/point on diagram marked by calculated points.

> Note that double-click name of univariant (or invariant) line will populate *Modes* and *Full output* panes at the bottom of application, so you can always check what is going on along lines. Double-clicking of tabs heading open outputs in larger separate window.

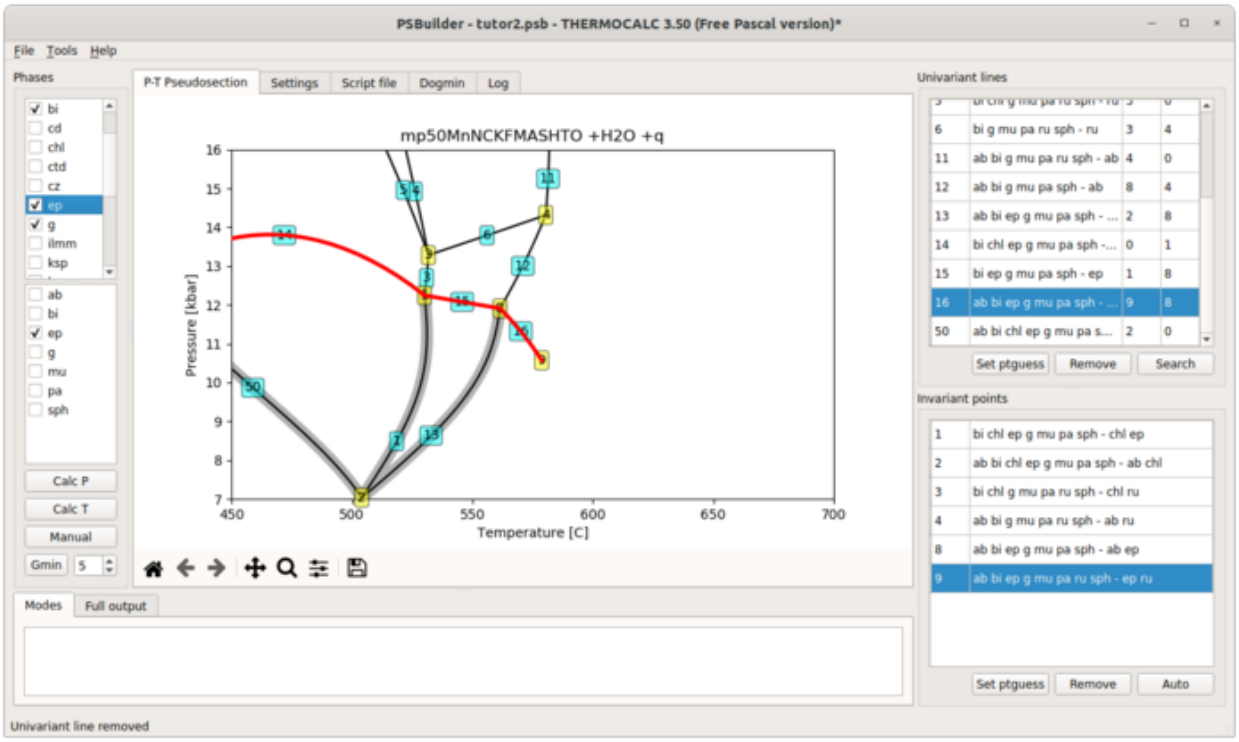


2.7 Starting guesses

`ptbuilder` stores all relevant information for each point or line already calculated. If you need to update starting guesses during construction of pseudosection, just choose invariant point or univariant line from which the starting guesses should be copied and click `Set ptguess` button. `ptbuilder` stores new starting guesses to your script file, so next calculation will use it. You can any time check and/or modify your script file with integrated editor on *Script file* pane. The *Log* pane always shows standard output of last *THERMOCALC* execution.

2.8 Phase out and phase stable lines

Double click on any phase in *Phases* list will highlight all univariant lines with zero modal proportion of selected phase and all phase present univariant lines.

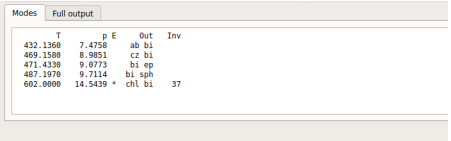


2.9 Manual invariant points or univariant lines

`Manual` button allows to add user-defined point or line. You need to select stable phases and zero mode phases accordingly. For manual univariant line begin and end invariant point must be specified. For manual invariant point, you can either specify position of point by clicking on diagram by mouse or when more than two univariant lines passing trough that point already exists, calculated intersection could be used.

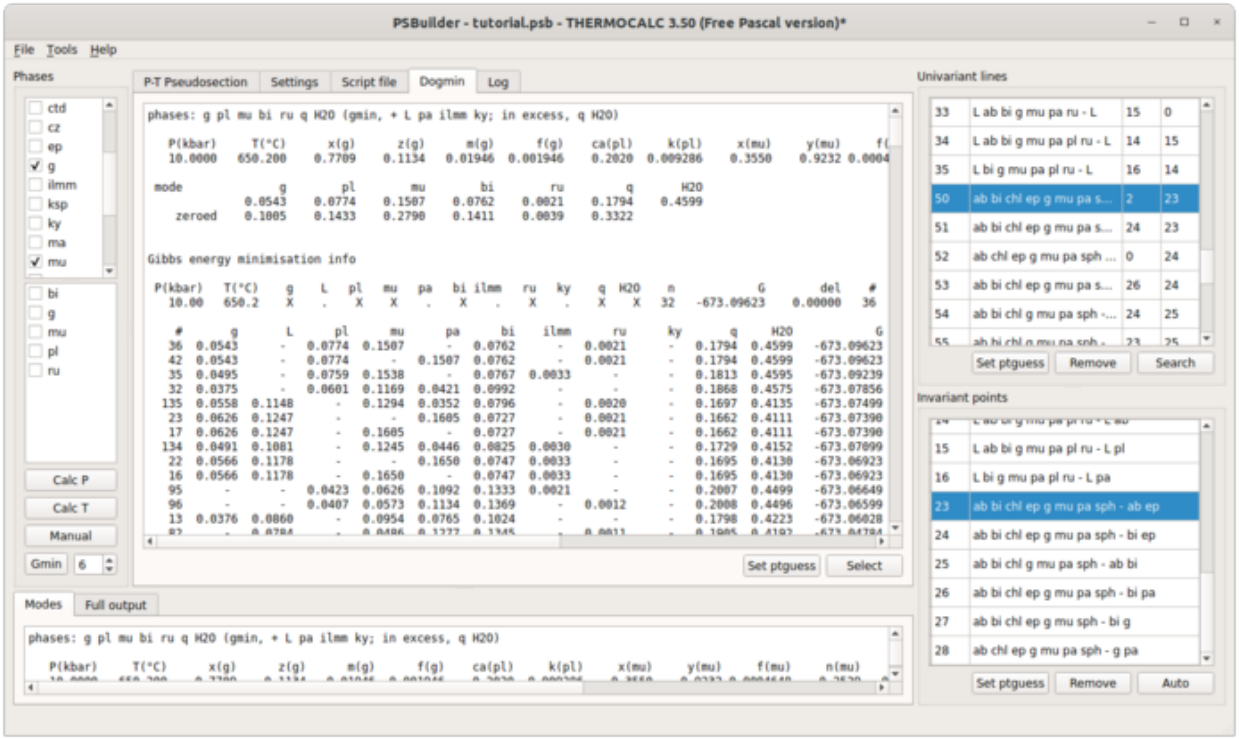
2.10 Searching for invariant points

To find out, what phase is appearing or disappearing along exiting univariant line, you can activate that line and click the `Search` button. Builder list possible (only found ones, if ptguesses are not appropriate, only metastable invariant points could be offered) solutions ordered along univariant line direction. The already calculated invariant points are marked. If there is one already calculated invariant point, the ptguesses from that point are used.

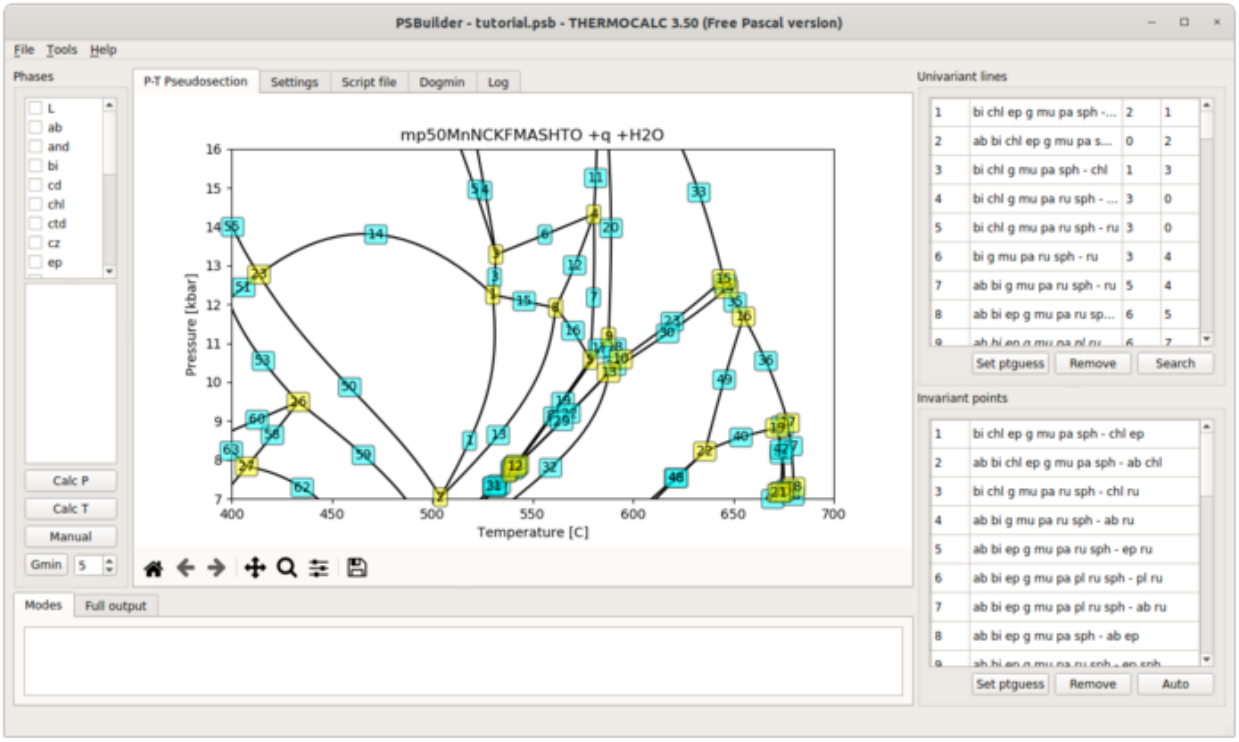


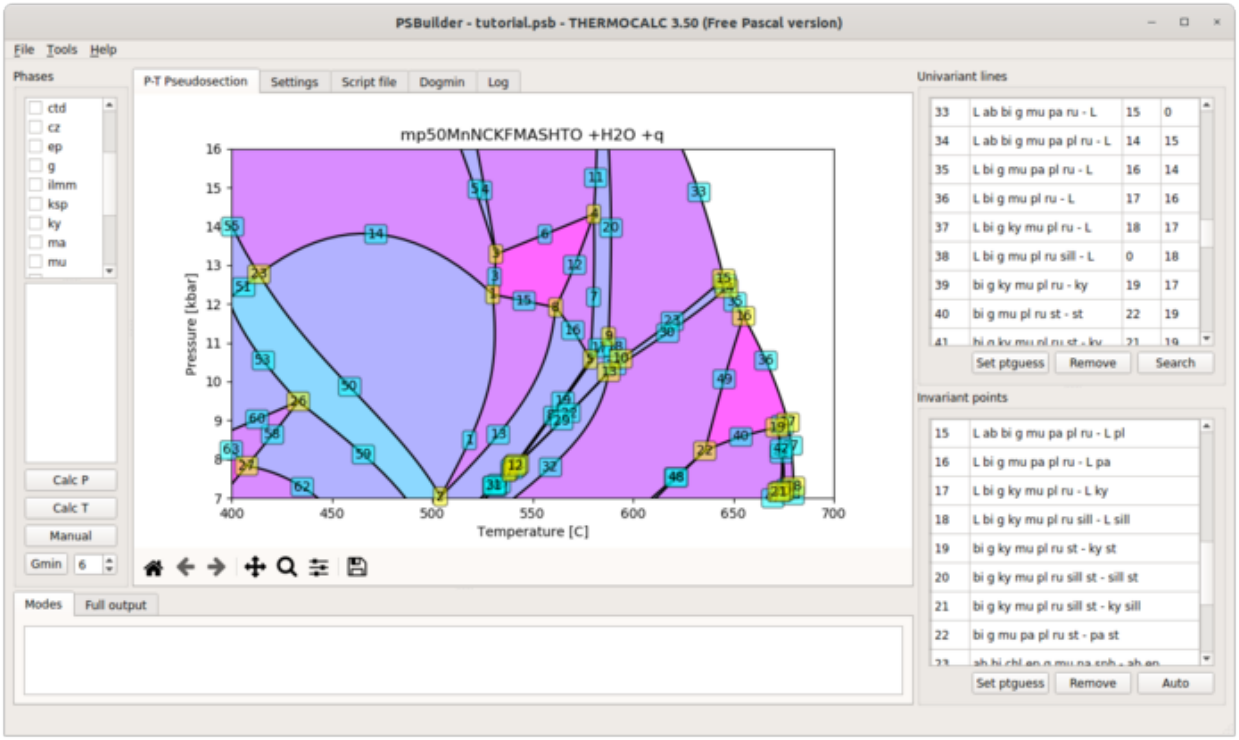
2.11 Dogmin

`Gmin` button runs THERMOCALC dogmin script, which tries to calculate phase equilibria between all possible subsets of a list of selected phases. The pressure and temperature is indicated by clicking on the diagram. Maximum variance to be considered (higher max variance -> fewer phases in smallest assemblage) is set in spin widget next to `Gmin` button. Ranked the equilibria in order of stability by comparing the Gibbs energies of each assemblage are shown in *Modes* pane. On *Dogmin* pane you can use `Select` button to select found assemblage in *Phases* and `Set guesses` to use ptguess of found solution.



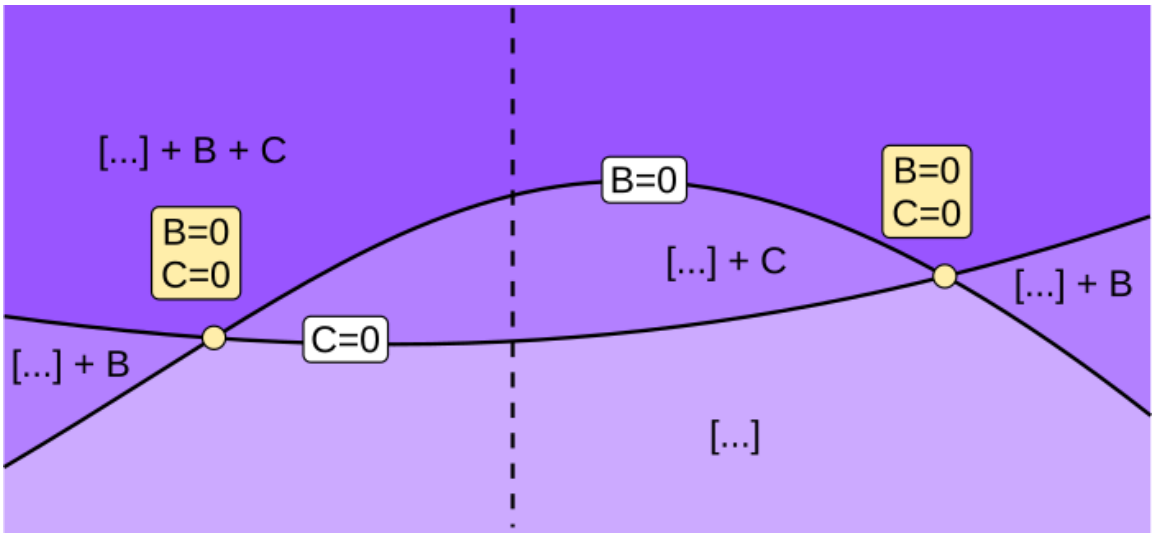
Finished pseudosection should contain topologically correct set of univariant lines and invariant points. Topology could be checked by creating areas (Tools>Show areas or Ctrl-A) of stable assemblages. If there are some topological or geometrical problems to create areas, corresponding IDs are reported on ‘’Mode” tab.


PSBuilder - tutorial.psb - THERMOCALC 3.50 (Free Pascal version)
File Tools Help
Phases
L
ab
and
bi
cd
chl
ctd
cz
ep
P-T Pseudosection
Settings
Script file
Dogmin
Log
mp50MnNCKFMASHTO +q +H2O
Pressure [kbar]
Temperature [C]
Univariant lines
1 bi chl ep g mu pa sph -... 2 1
2 ab bi chl ep g mu pa s... 0 2
3 bi chl g mu pa sph - chl 1 3
4 bi chl g mu pa ru sph - ... 3 0
5 bi chl g mu pa ru sph - ru 3 0
6 bi g mu pa ru sph - ru 3 4
7 ab bi g mu pa ru sph - ru 5 4
8 ab bi ep g mu pa ru sp... 6 5
Set ptguess
Remove
Search
Invariant points
1 bi chl ep g mu pa sph - chl ep
2 ab bi chl ep g mu pa sph - ab chl
3 bi chl g mu pa ru sph - chl ru
4 ab bi g mu pa ru sph - ab ru
5 ab bi ep g mu pa ru sph - ep ru
6 ab bi ep g mu pa pl ru sph - pl ru
7 ab bi ep g mu pa pl ru sph - ab ru
8 ab bi ep g mu pa sph - ab ep
Set ptguess
Remove
Auto
Calc P
Calc T
Manual
Gmin 5
Modes
Full output


PSBuilder - tutorial.psb - THERMOCALC 3.50 (Free Pascal version)
File Tools Help
Phases
ctd
cz
ep
g
ilmm
ksp
ky
ma
mu
P-T Pseudosection
Settings
Script file
Dogmin
Log
mp50MnNCKFMASHTO +H2O +q
Pressure [kbar]
Temperature [C]
Univariant lines
33 L ab bi g mu pa ru - L 15 0
34 L ab bi g mu pa pl ru - L 14 15
35 L bi g mu pa pl ru - L 16 14
36 L bi g mu pl ru - L 17 16
37 L bi g ky mu pl ru - L 18 17
38 L bi g mu pl ru sill - L 0 18
39 bi g ky mu pl ru - ky 19 17
40 bi g mu pl ru st - st 22 19
Set ptguess
Remove
Search
Invariant points
15 L ab bi g mu pa pl ru - L pl
16 L bi g mu pa pl ru - L pa
17 L bi g ky mu pl ru - L ky
18 L bi g ky mu pl ru sill - L sill
19 bi g ky mu pl ru st - ky st
20 bi g ky mu pl ru sill st - sill st
21 bi g ky mu pl ru sill st - ky sill
22 bi g mu pa pl ru st - pa st
Set ptguess
Remove
Auto
Calc P
Calc T
Manual
Gmin 6
Modes
Full output

2.12 Double crossing univariant lines

Common problem is that you realize during construction of pseudosection, that some univariant lines are double crossing and therefore two invariant points and two separated segments of univariant lines with identical assemblage and zero mode phase(s) exists. The **pypsbuilders** cannot have those "identical" points and lines in single project an pseudosection has be split in two parts by defining axes limits on *Settings* pane. Split line should be placed approximately like on the figure.



The all parts of single pseudosection could be post-processed and visualized by **psexplorers** simulataneously as single pseudosection.

When you have already calculated invariant points and lines and you want to use them in separate project, create new project, set axes limits properly and use *File>Import > Import into range* to read it.

2.13 T-X and P-X pseudosections

You can create T-X and P-X pseudosection using exactly same steps like for P-T pseudosection using *txbuilder* or *pxbuilder*.

CHAPTER

THREE

PSEUDOSECTION EXPLORERS TUTORIAL

psexplorers provides several post-processing methods and visualizations of already constructed pseudosections. You can also create isopleths diagrams or do calculations along paths.

It provides four command-line scipts *psgrid*, *psshow* and *psiso* for quick visualizations. For options, check built-in help, e.g.:

```
$ psshow -h
```

To show pseudosection, you need to provide project file:

```
$ psshow -b /some/path/project.ptb
```

3.1 Using psexplorers in Python (or Jupyter notebook)

To access all options of **psexplorer** it is suggested to use Python API. You need to activate the pyps environment and run python interpreter:

```
$ conda activate pyps
$ python
```

To use **psexplorer** we need to import appropriate class, which contains most of the methods to work with pseudosection.

- `PTPS` class for P-T pseudosection constructed with `ptbuilder`
- `TXPS` class for T-X pseudosection constructed with `txbuilder`
- `PXPS` class for P-X pseudosection constructed with `pxbuilder`

All following commands must be executed in Python interpreter.

```
from pypsbuilder import PTPS
```

The second step is to create instance of pseudosection using existing project file.

```
pt = PTPS('/some/path/project.ptb')
```

We can check, whether the pseudosection already contains gridded calculations. If not, we can use `calculate_composition` method to calculate compositional variations on grid. The resulting data are stored in project file. Note that any new modifications of the project by *pypsbuilder** will discard compositional variations on grid and must be calculated again.

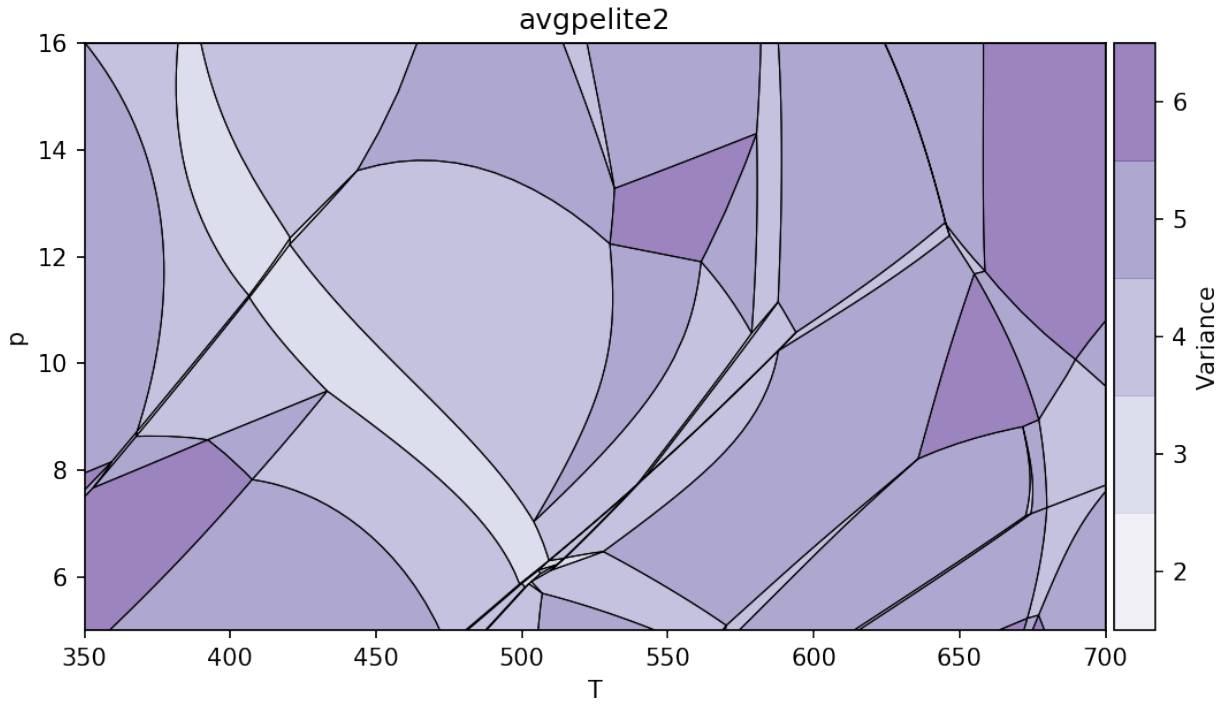
```
if not pt.gridded:
    pt.calculate_composition(nx=50, ny=50)
```

```
Gridding 1/1: 100%|| 2500/2500 [03:10<00:00, 13.12it/s]
Grid search done. 0 empty points left.
```

3.2 Visualize pseudosection

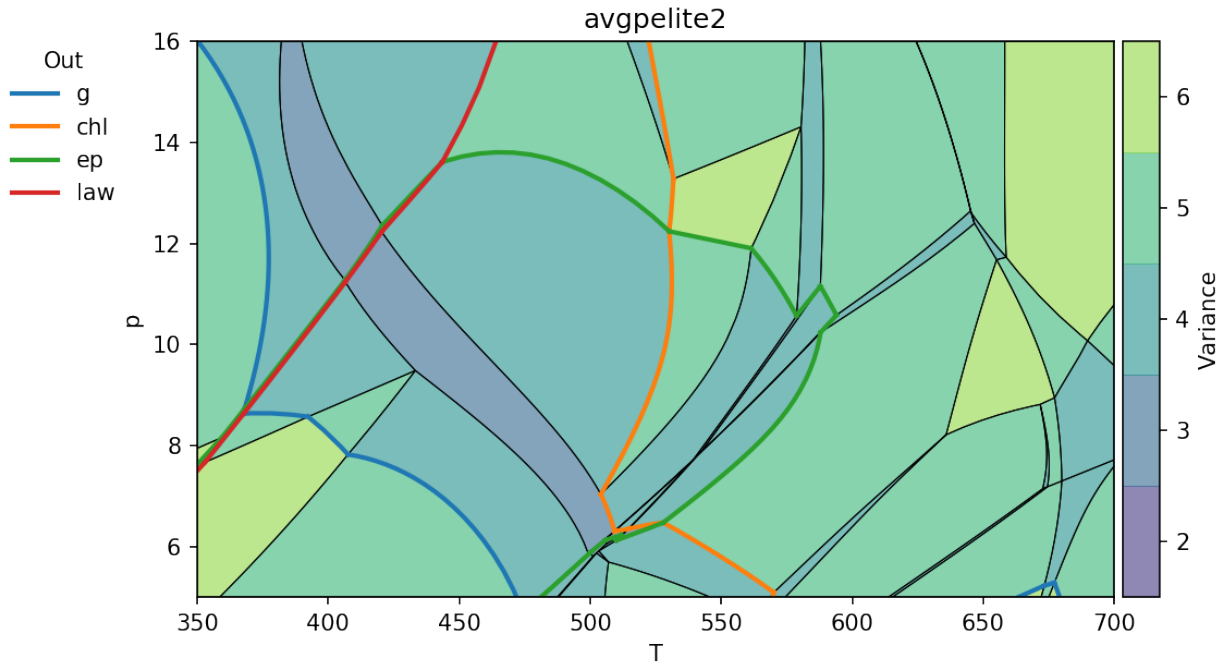
To show pseudosection, we can use `show` method

```
pt.show()
```



The keyword arguments `cmap` and `out` could be used to modify colormap and highlight zero mode lines across pseudosection.

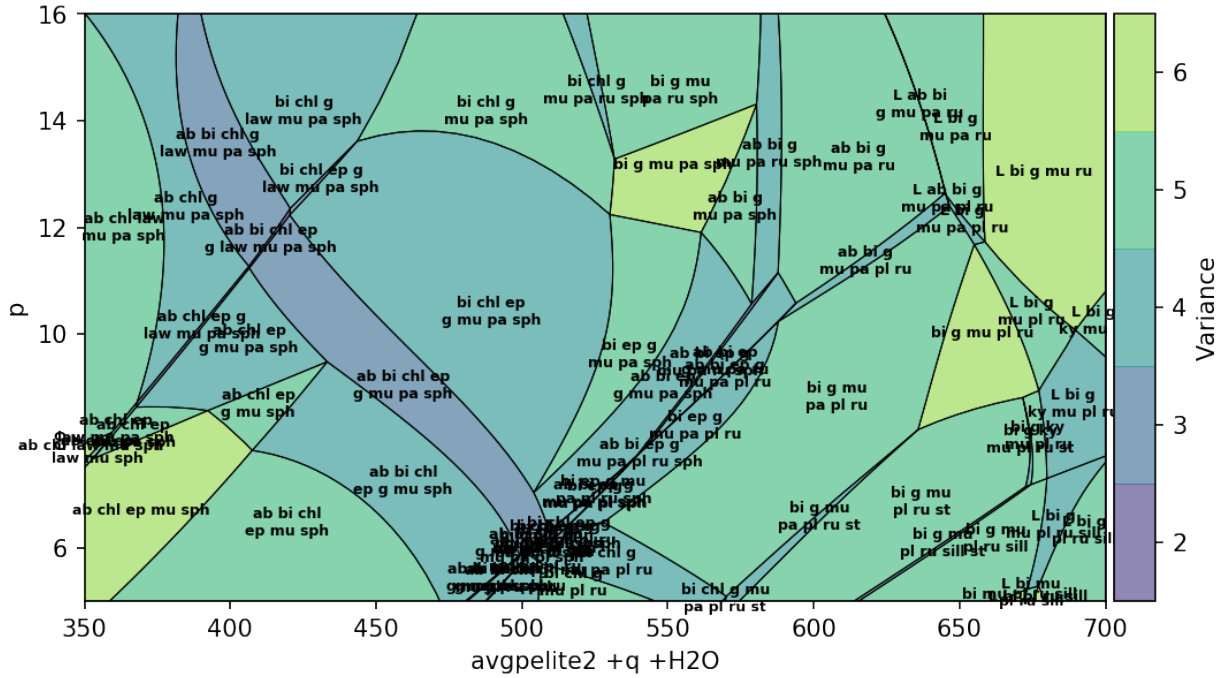
```
pt.show(cmap='viridis', out=['g', 'chl', 'ep', 'law'])
```



The keyword arguments `bulk` and `label` set whether the bulk composition is shown on figure and whether the fields are labeled by phases.

```
pt.show(cmap='viridis', bulk=True, label=True)
```

H2O	SiO2	Al2O3	CaO	MgO	FeO	K2O	Na2O	TiO2	MnO	O
50	35.6	5.8	0.695	2.33	2.89	1.34	0.995	0.335	0.05	0.005



The pseudosection `identify` method could be used to identify stable assemblage for given p and T conditions. Note that returned key (Python frozenset) is used to identify stable assemblage in many PTPS methods.

```
key = pt.identify(550, 13)
print(key)
```

```
frozenset({'sph', 'pa', 'q', 'g', 'mu', 'H2O', 'bi'})
```

3.3 Access data and variables stored in project

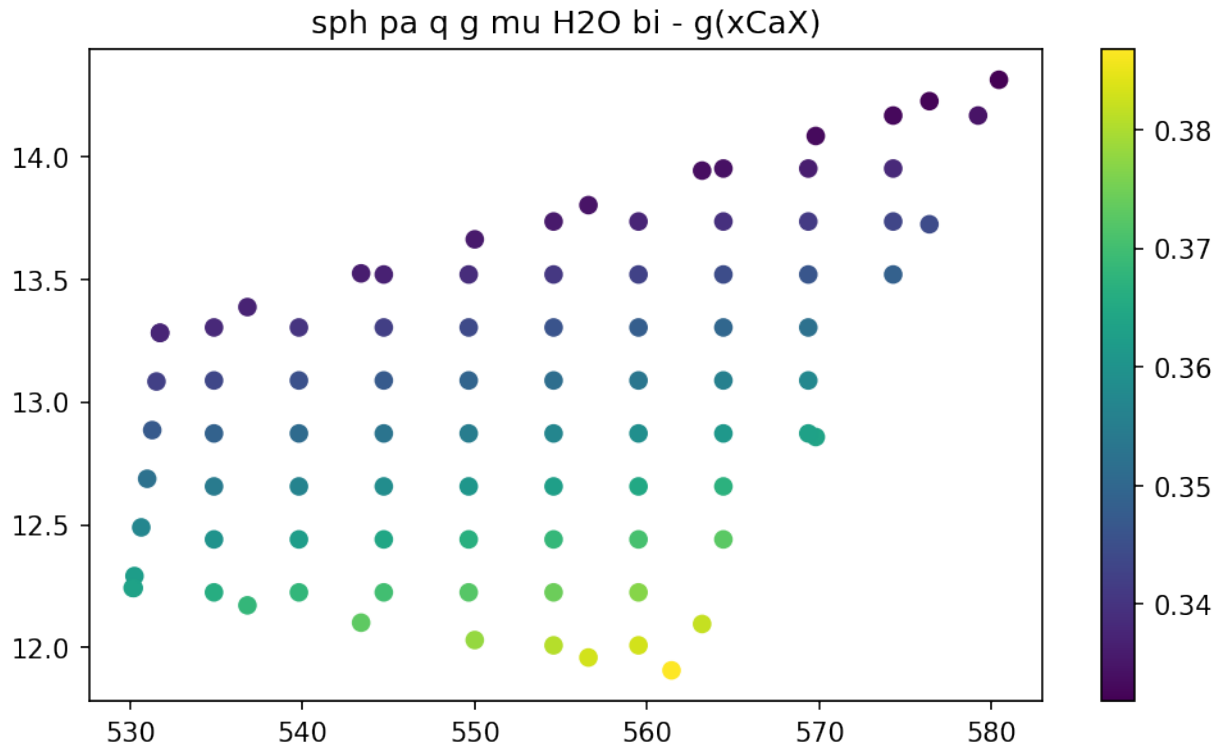
The calculated data are usually accessed using stable assemblage key (see above). Theera are three groups of data stored 1) at invariant points, 2) along univariant lines and 3) on grid covering multivariate fields. To see data coverage and all available variables, you can use `show_data` method. When no variable (or expression) is provided, method will show available variables.

```
pt.show_data(key, 'g')
```

```
Missing expression argument. Available variables for phase g are:
mode x z m f xMgX xFeX xMnX xCaX xAlY xFe3Y H2O SiO2 Al2O3 CaO MgO FeO K2O Na2O TiO2 MnO␣
↪O factor G H S V rho
Available end-members for g: kho gr alm py spss
```

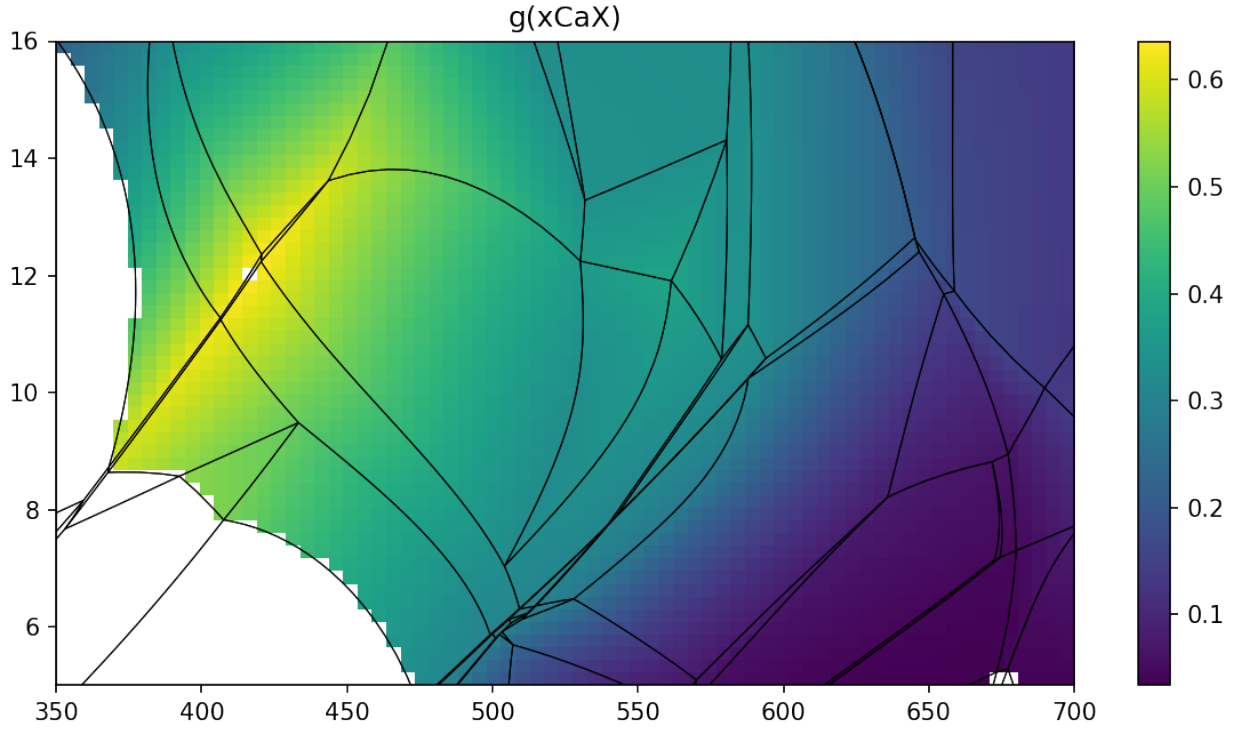
Once variable is provided, the all available data are shown.

```
pt.show_data(key, 'g', 'xCaX')
```



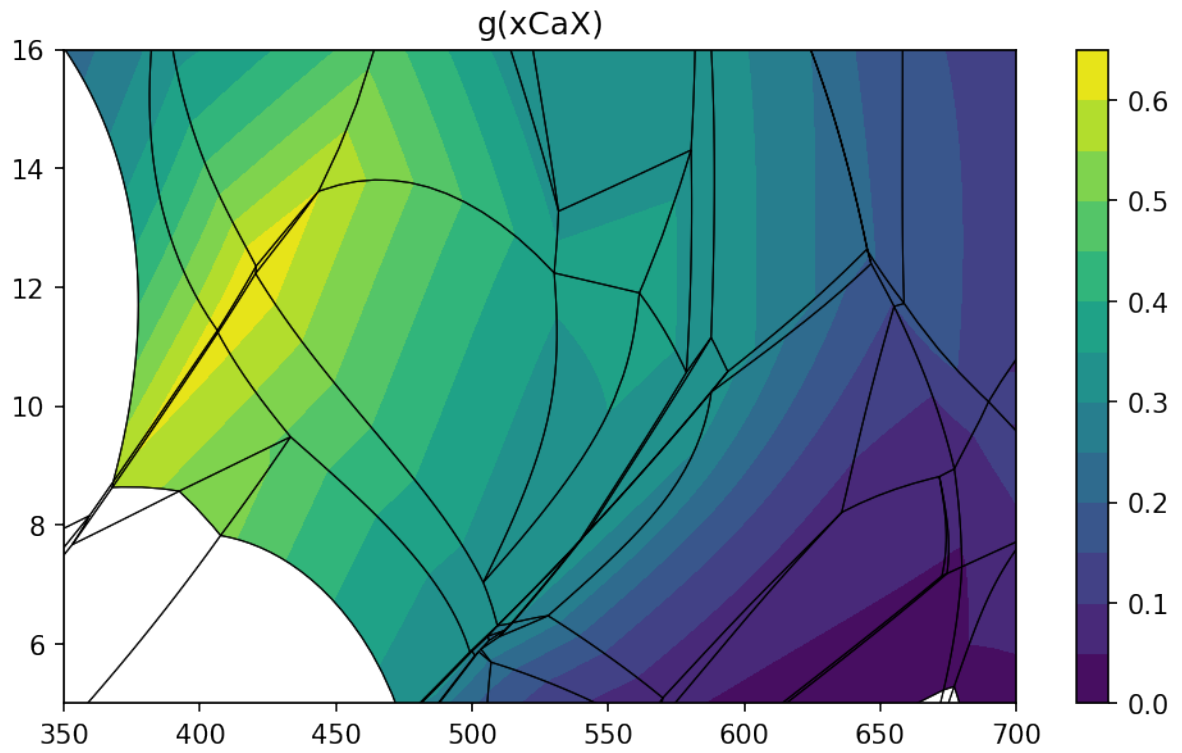
For data on the grid you can visualize them for all diagram in once using `show_grid` method.

```
pt.show_grid('g', 'xCaX')
```



To create isopleths diagram you can use `isopleths` method. Note that contours are created separately for each stable assemblage allowing proper geometry of isopleths.

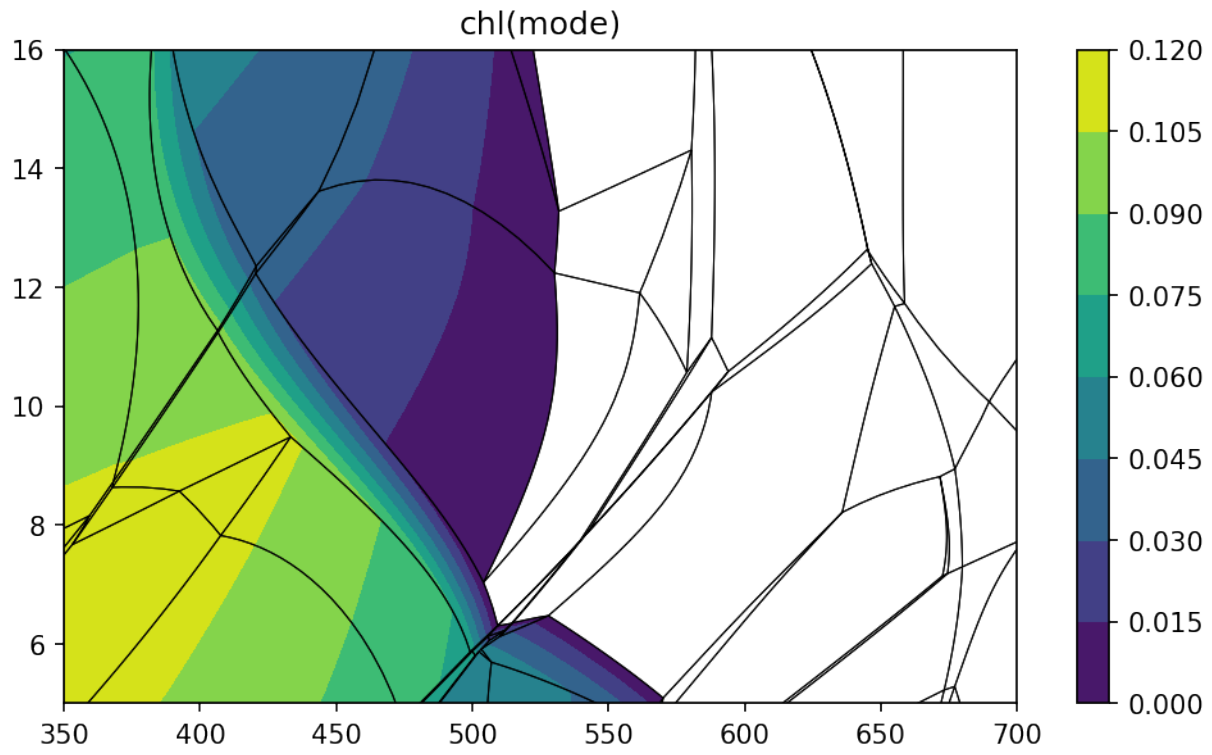
```
pt.isopleths('g', 'xCaX', N=14)
```




```
pt.isopleths('chl')
```

```
Missing expression argument. Available variables for phase chl are:
mode x y f m QAl Q1 Q4 xMgM1 xMnM1 xFeM1 xAlM1 xMgM23 xMnM23 xFeM23 xMgM4 xFeM4 xFe3M4␣
↪xAlM4 xSiT2 xAlT2 H2O SiO2 Al2O3 CaO MgO FeO K2O Na2O TiO2 MnO O factor G H S V rho
Available end-members for chl: ames mmchl ochl1 f3clin afchl ochl4 clin daph
```

```
pt.isopleths('chl', 'mode')
```



3.4 Calculations along PT paths

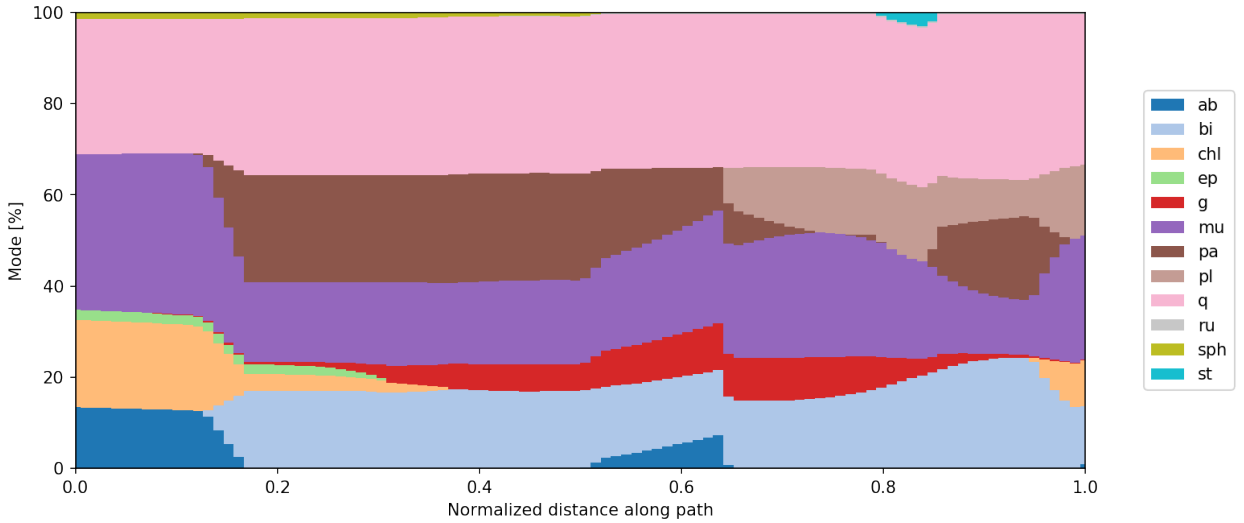
PTPS allows you to evaluate equilibria along user-defined PT path. PT path is defined by series of points (path is interpolated) and method `collect_ptpath` do actual calculations. It runs THERMOCALC with ptguesses obtained from existing calculations.

```
t = [380, 480, 580, 640, 500]
p = [7, 12, 15, 9, 5.5]
pa = pt.collect_ptpath(t, p)
```

```
Calculating: 100%|| 100/100 [00:03<00:00, 25.86it/s]
```

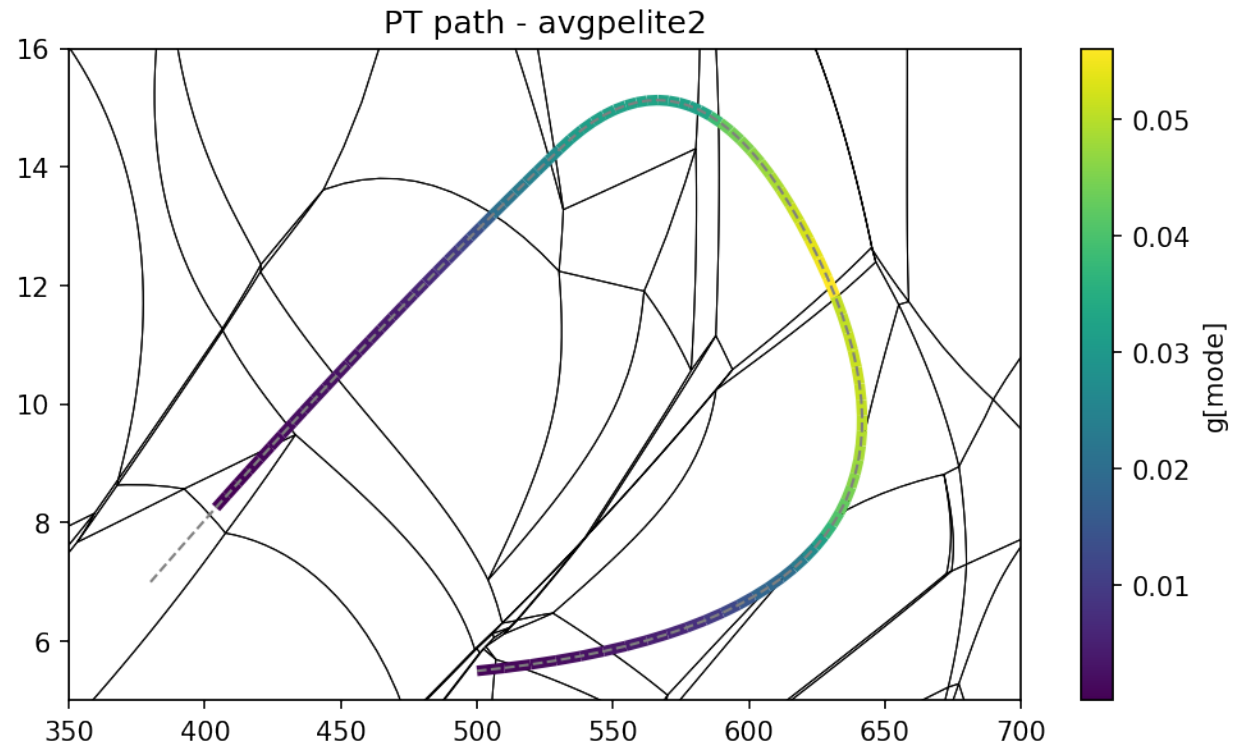
You can see phase modes along PT path using `show_path_modes` method.

```
pt.show_path_modes(pa, exclude=['H2O'])
```



or show value of user-defined expression shown as colored strip on PT space.

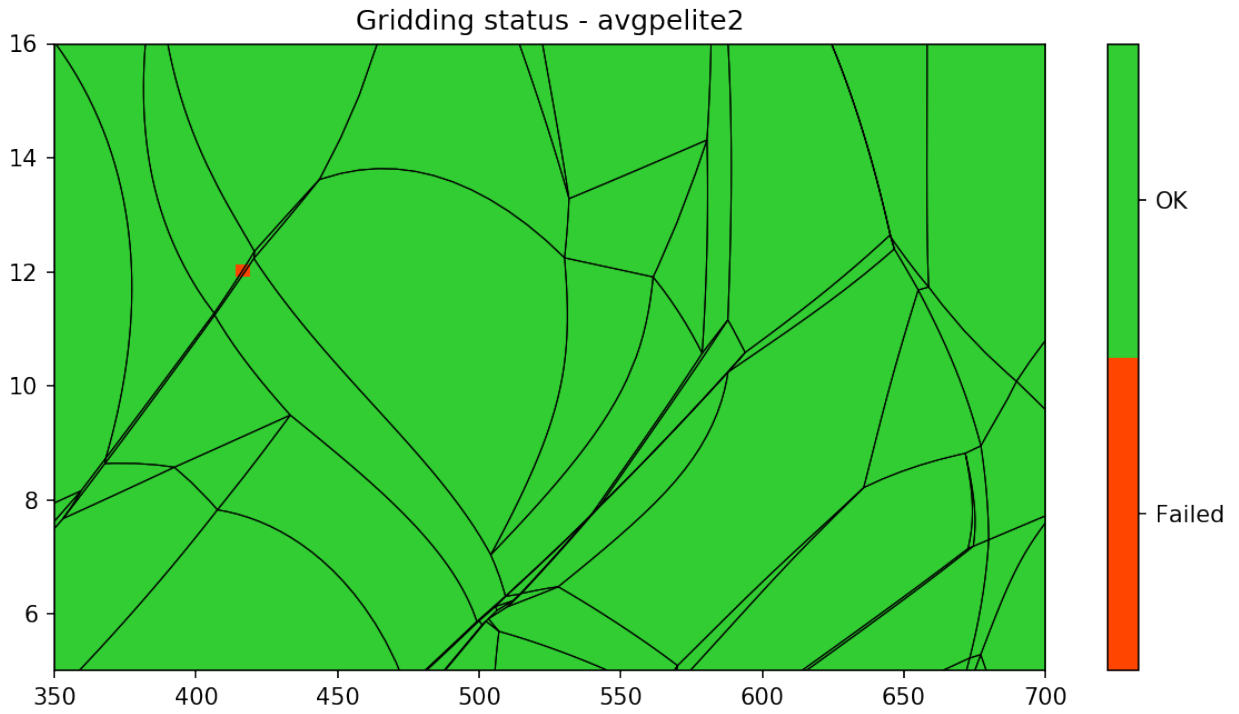
```
pt.show_path_data(pa, 'g', 'mode')
```



3.5 Extra

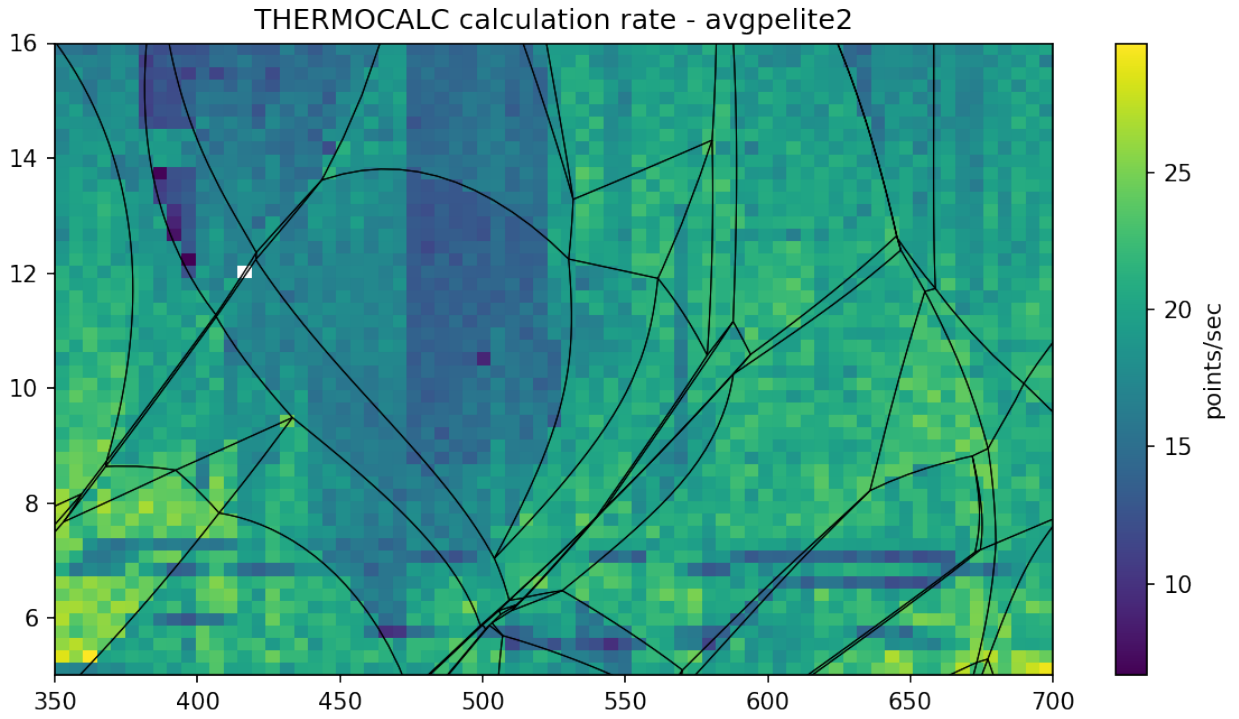
`show_status` method shows status of calculations on the grid. Possible failed calculations are shown.

```
pt.show_status()
```



Do you want to know execution time of THERMOCALC on individual grid points? Check `show_delta` method.

```
pt.show_delta(pointsec=True)
```



For full description of Python API check: *Python API*.

CHAPTER

FOUR

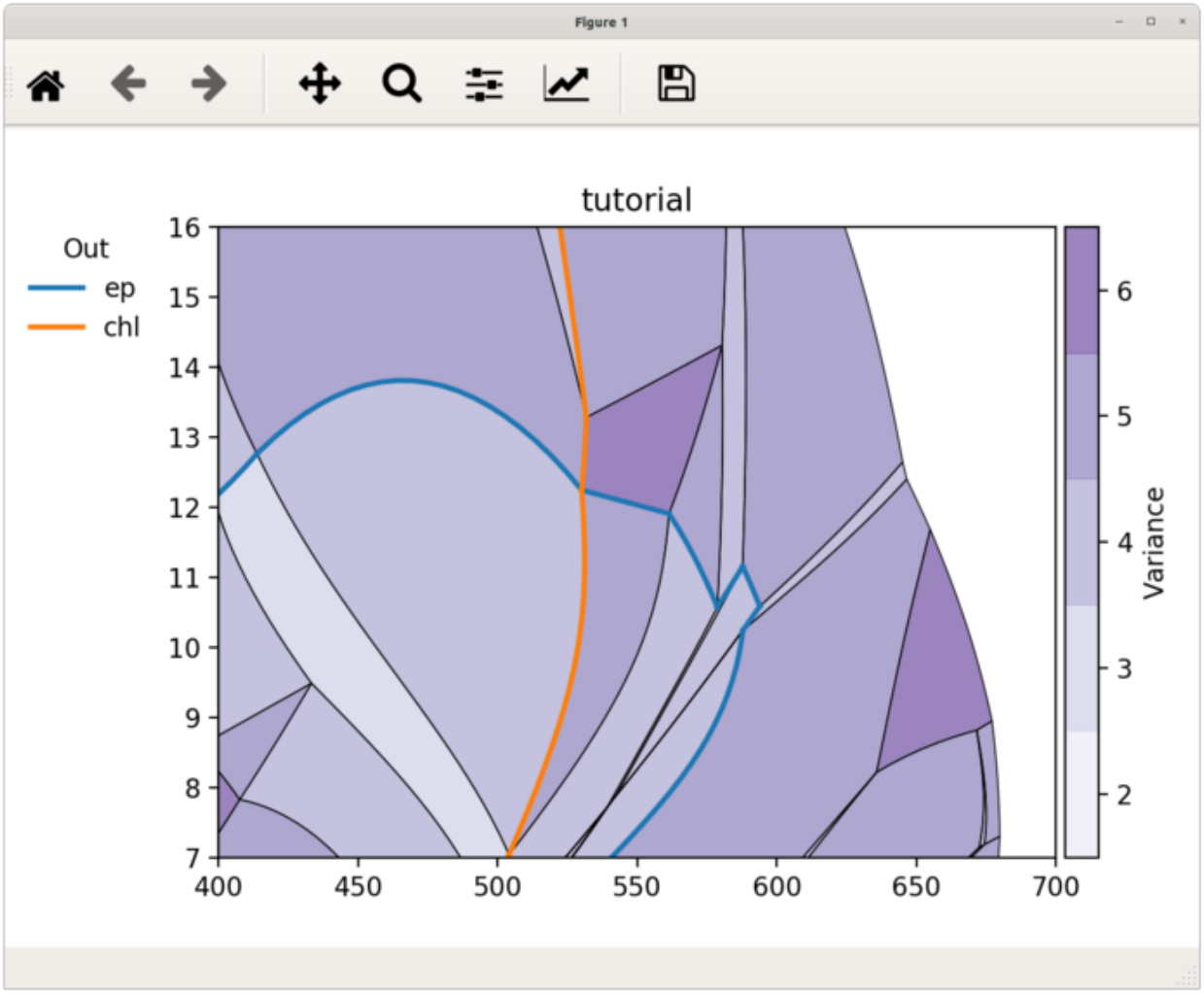
COMMAND LINE SCRIPTS

Before any further calculations you can check and draw your pseudosection using *psshow* command which construct finished areas within your project. It has few options to label pseudosection with assamblages or highlight out phase lines.

```
$ psshow -h
usage: psshow [-h] [-o OUT [OUT ...]] [-l] [--origwd] [-b] [--cmap CMAP]
          [--alpha ALPHA] [--connect] [--high HIGH]
          [--tolerance TOLERANCE]
          project [project ...]

Draw pseudosection from project file

positional arguments:
  project               builder project file(s)

optional arguments:
  -h, --help            show this help message and exit
  -o OUT [OUT ...], --out OUT [OUT ...]
                        highlight out lines for given phases
  -l, --label           show area labels
  --origwd              use stored original working directory
  -b, --bulk            show bulk composition on figure
  --cmap CMAP           name of the colormap
  --alpha ALPHA         alpha of colormap
  --connect             whether mouse click echo stable assemblage
  --high HIGH           highlight field defined by set of phases
  --tolerance TOLERANCE
                        tolerance to simplify univariant lines
```

To draw pseudosection with marked epidote-out and chlorite-out lines execute:

```
$ psshow '/path/to/project.ptb' -o ep chl
```



4.1 Draw isopleths diagrams

To create isopleths diagrams the pseudoction should be gridded at first (In other case only values from univariant lines and invariant points are used and interpolated accross areas). Command *psgrid* will do all calculations and result are saved afterwards, so next time results are automatically loaded. Be aware that calculations takes some time.

```
$ psgrid -h
usage: psgrid [-h] [--nx NX] [--ny NY] [--origwd] [--tolerance TOLERANCE]
              project [project ...]

Calculate compositions in grid

positional arguments:
  project               builder project file(s)

optional arguments:
  -h, --help            show this help message and exit
  --nx NX               number of T steps
```

(continues on next page)

(continued from previous page)

```
  --ny NY               number of P steps
  --origwd              use stored original working directory
  --tolerance TOLERANCE
                        tolerance to simplify univariant lines
```

For gridding pseudosection with grid 50x50 run following command:

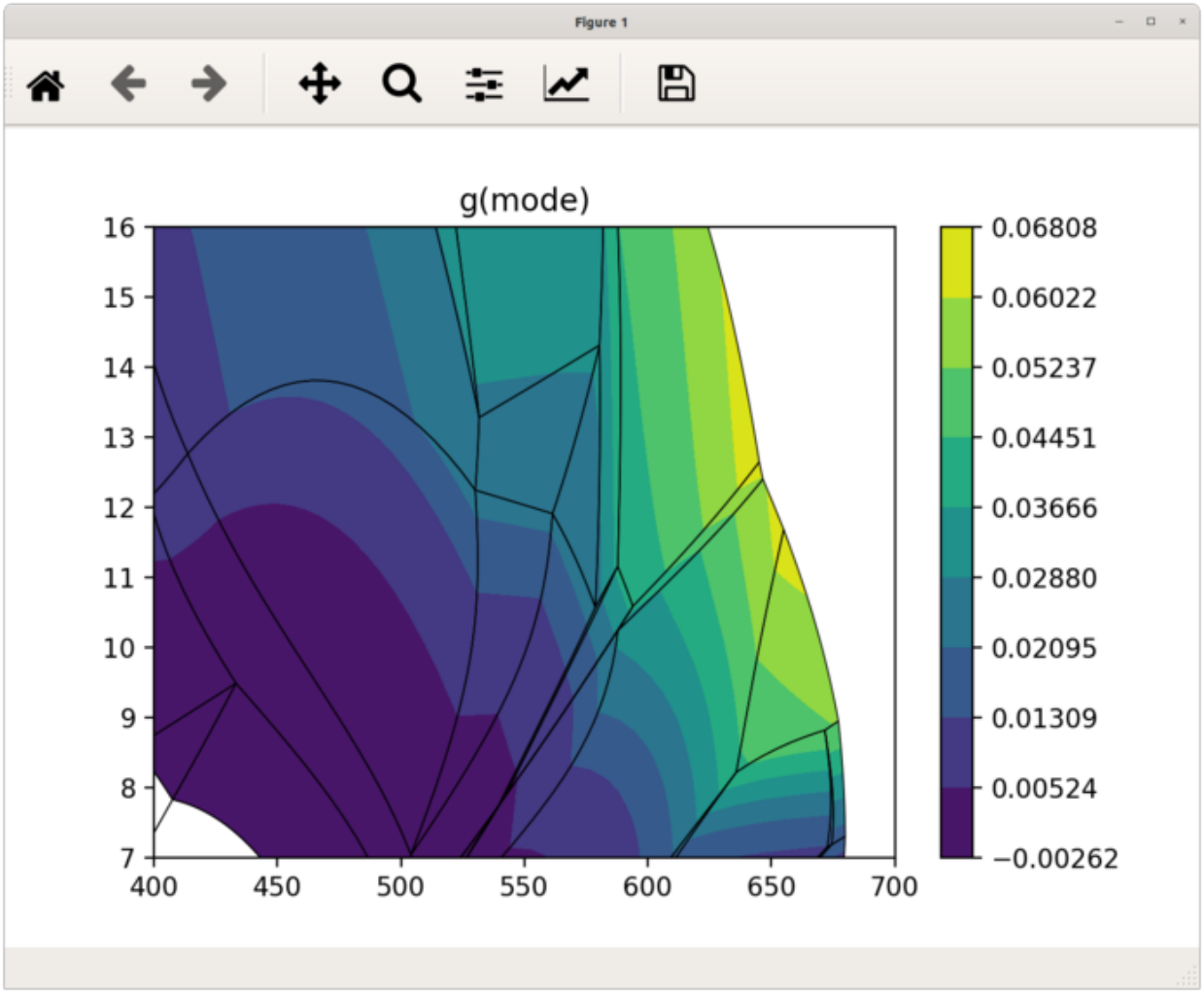
```
$ psgrid '/path/to/project.ptb' --nx 50 --ny 50
            Gridding: 100%|| 2500/2500 [01:30<00:00, 27.62it/s]
            Grid search done. 0 empty grid points left.
```

Once gridded you can draw isopleths diagrams using *psiso* command:

```
            $ psiso -h
usage: psiso [-h] [-e EXPR] [-f] [--origwd] [-o OUT [OUT ...]] [--nosplit]
             [-b] [--step STEP] [--ncont NCONT] [--colors COLORS]
             [--cmap CMAP] [--smooth SMOOTH] [--labelkey LABELKEY]
             [--high HIGH] [--tolerance TOLERANCE]
             project [project ...] phase

Draw isopleth diagrams

positional arguments:
  project               builder project file(s)
  phase                 phase used for contouring

optional arguments:
  -h, --help            show this help message and exit
  -e EXPR, --expr EXPR  expression evaluated to calculate values
  -f, --filled          filled contours
  --origwd              use stored original working directory
  -o OUT [OUT ...], --out OUT [OUT ...]
                        highlight out lines for given phases
  --nosplit             controls whether the underlying contour is removed or
                        not
  -b, --bulk            show bulk composition on figure
  --step STEP           contour step
  --ncont NCONT         number of contours
  --colors COLORS       color for all levels
  --cmap CMAP           name of the colormap
  --smooth SMOOTH       smoothness of the approximation
  --labelkey LABELKEY   label contours in field defined by set of phases
  --high HIGH           highlight field defined by set of phases
  --tolerance TOLERANCE
                        tolerance to simplify univariant lines
```

Following example shows isopleths of garnet mode:

```
$ psiso '/path/to/project.ptb' -f g -e mode
```



If the *expression* argument is not provided, the `psexplorer` shows list of all calculated variables available for given phase.

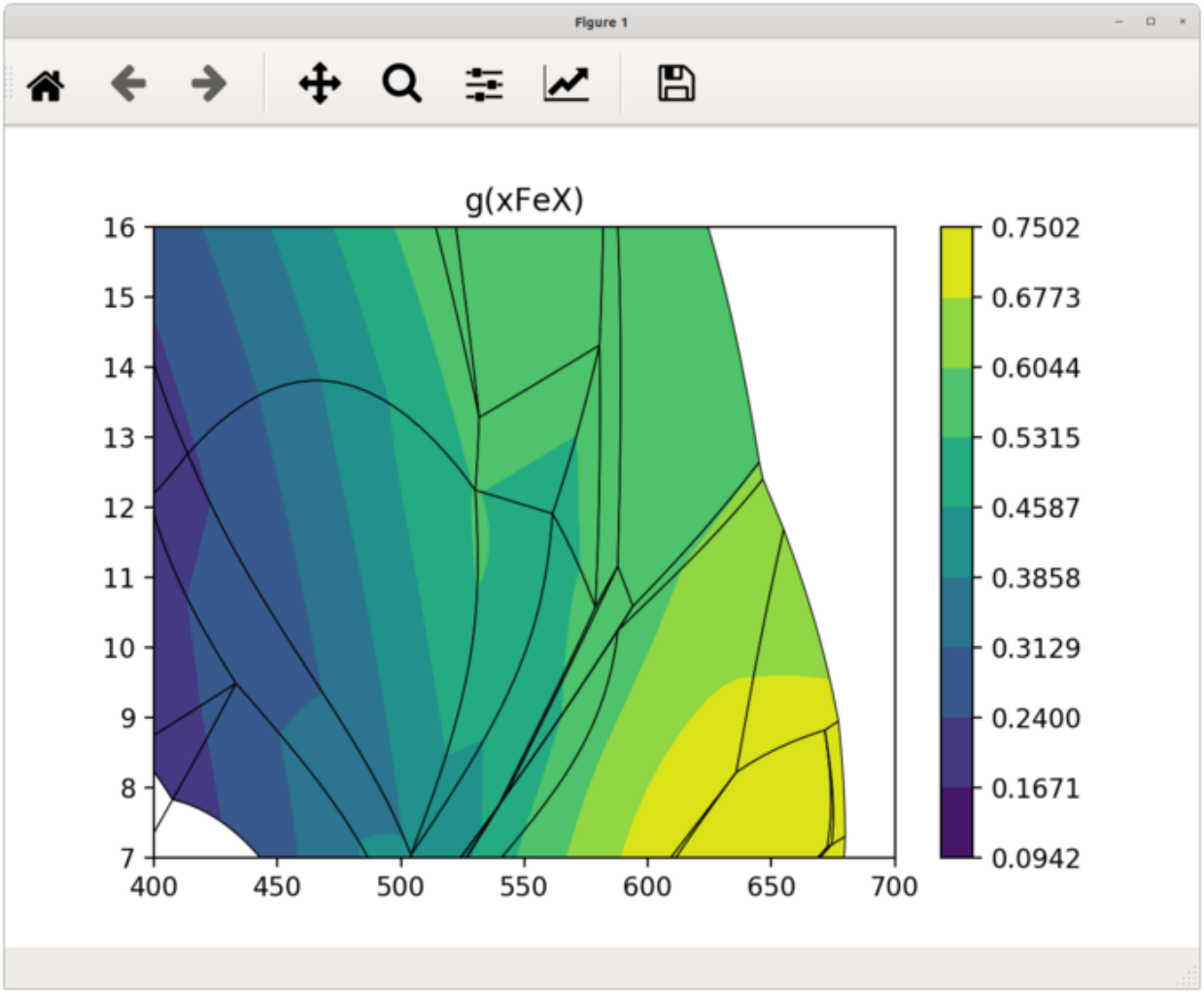
```
$ psiso '/path/to/project.ptb' -f g
Missing expression argument. Available variables for phase g are:
mode x z m f xMgX xFeX xMnX xCaX xAlY xFe3Y H2O SiO2 Al2O3 CaO MgO FeO K2O
Na2O TiO2 MnO O factor G H S V rho
```

To draw isopleths of almandine garnet proportion you can use expression from a-x file $alm = x + (-m)\,x + (-x)\,z$:

```
$ psiso '/path/to/project.ptb' -f g -e 'x-m*x-x*z'
```

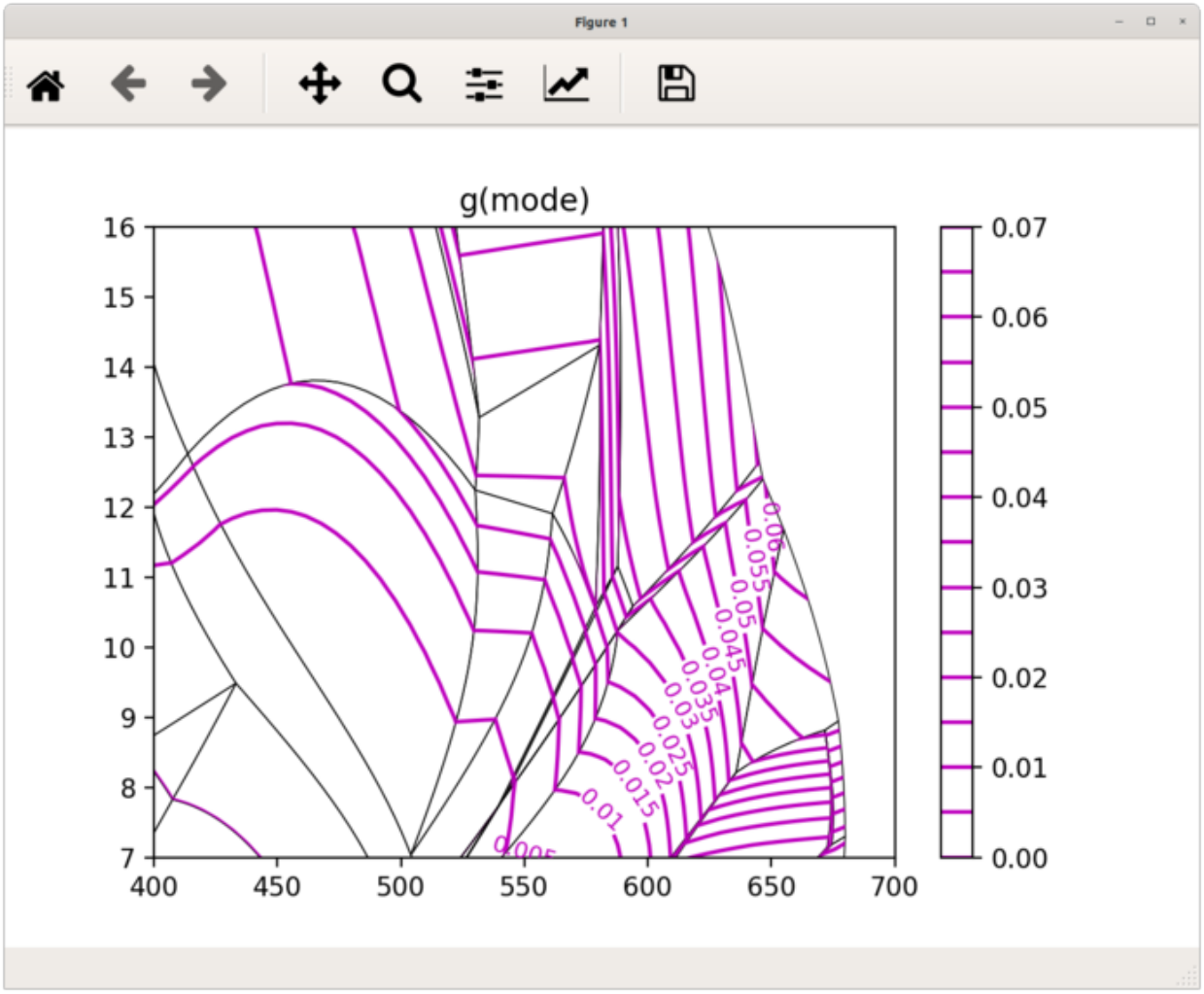
or use variable *xFeX*:

```
$ psiso tutorial.ptb -f g -e xFeX
```



If you need to label contour lines, you can use labelkey option to define field, where contour labels are plotted.

```
$ psiso '/path/to/project.ptb' g -e mode --labelkey "H2O bi g mu pa pl q ru"
          --step 0.005 --colors m
```



Another example of some other options.

```
$ psiso tutorial.ptb -f g -e mode --step 0.005 --high "H2O bi g mu pa pl q ru"
--out chl ep --cmap YlGnBu_r
```

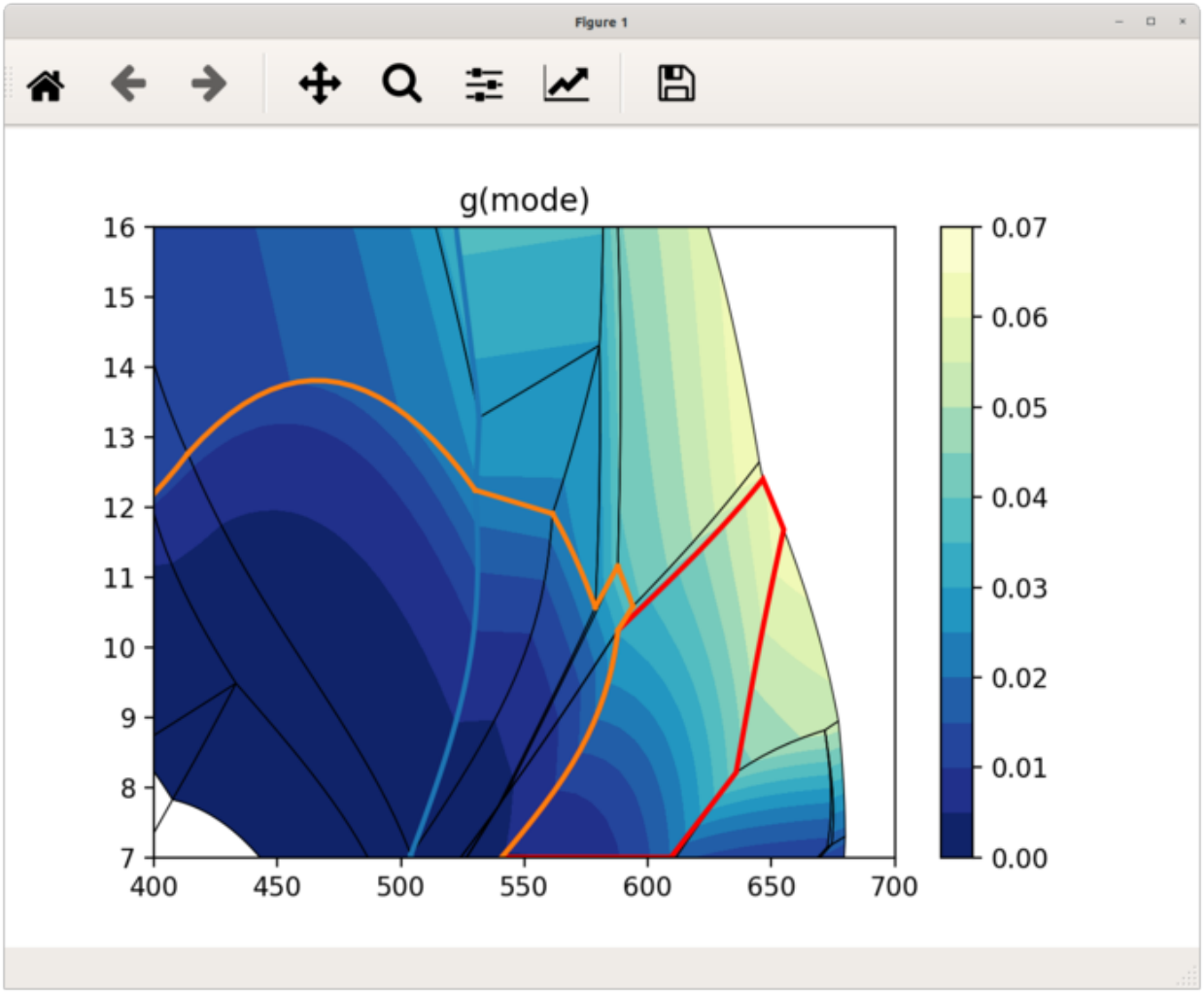


Figure 1
g(mode)
16
15
14
13
12
11
10
9
8
7
400
450
500
550
600
650
700
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00

CHAPTER

FIVE

PYTHON API

pypsbuilder expose several classes which could be used in Python sctipts, interactively or in Jupyter notebooks. Check included jupyter notebook to see how it works.

Here you can find auto-generated documentation of main classes, methods and functions.

5.1 psexplorer module

PTPS	Class to postprocess ptbuilder project
TXPS	Class to postprocess txbuilder project
PXPS	Class to postprocess pxbuilder project

5.1.1 pypsbuilder.psexplorer.PTPS

class pypsbuilder.psexplorer.**PTPS**(**args*, ***kwargs*)

Bases: PS

Class to postprocess ptbuilder project

__init__(**args*, ***kwargs*)

Create PTPS class instance from builder project file.

Parameters

- **projfile** (*str, Path*) – psbuilder project file or files
- **tolerance** (*float*) – if not None, simplification tolerance. Default None
- **origwd** (*bool*) – If True TCAPI uses original stored working directory Default False.

Methods

__init__(*args, **kwargs)	Create PTPS class instance from builder project file.
add_overlay(ax[, fc, ec, label, skiplabels, ...])	
calculate_composition([nx, ny])	Method to calculate compositional variations on grid.
check_phase_expr(phase, expr)	

continues on next page

Table 1 – continued from previous page

collect_all_data_keys()	Collect all phases and variables calculated on grid.
collect_data(key, phase, expr[, which])	Convinient function to retrieve values of expression for given phase for user-defined combination of results of divariant field identified by key.
collect_grid_data(key, phase, expr)	Retrieve values of expression for given phase for all GridData points within divariant field identified by key.
collect_inv_data(key, phase, expr)	Retrieve value of variables based expression for given phase for all invariant points surrounding divariant field identified by key.
collect_ptpath(tpath, ppath[, N, kind])	Method to collect THERMOCALC calculations along defined PT path.
collect_uni_data(key, phase, expr)	Retrieve values of expression for given phase for all univariant lines surrounding divariant field identified by key.
common_grid_and_masks(**kwargs)	Initialize common grid and mask for all partial grids
create_masks()	Update grid masks from existing divariant fields
fix_solutions()	Method try to find solution for grid points with failed status.
format_coord(x, y)	
gendrawpd([export_areas])	Method to write drawpd file
get_gridded(phase[, expr, which, smooth])	
get_grids(phase[, expr])	Convinient function to get dictionary of grids.
get_nearest_grid_data(x, y)	Retrieve nearest results from GridData to point.
get_section_id(x, y)	Return index of pseudosection and grid containing point
gidentify([label, skiplabels, labelfs])	Visual version of *identify* method.
ginput_path([label, skiplabels, labelfs])	Collect Path data by mouse digitizing.
glabel([label, skiplabels, labelfs])	Return formatted string of assamblage at PT point provided by mouse click.
identify(x, y)	Return key (frozenset) of divariant field for given temperature and pressure.
invs_from_unilist(ix, unilist)	Return set of IDs of invariant points associated with unilines.
isopleths(phase[, expr])	Method to draw compositional isopleths.
isopleths_vector(phase[, expr])	Method to draw vectorized compositional isopleths.
merge_data(phase, expr[, which])	Returns merged data obtained by *collect_data* method for all divariant fields.
onclick(event)	
overlap_isopleths(*args, **kwargs)	Function to overlaped isopleths ranges.
pointcalc([label, skiplabels, labelfs])	
remove_grid_data(key, phase, expr)	Remove calculated data from grid and mark as failed
save()	Save gridded copositions and constructed divariant fields into psbuilder project file.
save_tab(comps[, tabfile])	Export gridded values to Perple_X tab format.
search_composition(*args[, interpolation, ...])	Function to plot root squared error of calculated and estimated values.
show(**kwargs)	Method to draw PT pseudosection.

continues on next page

Table 1 – continued from previous page

show_data(key, phase[, expr, which])	Convinient function to show values of expression for given phase for user-defined combination of results of divariant field identified by key.
show_delta([label, pointsec, skiplabels, ...])	Shows THERMOCALC execution time for all grid points.
show_grid(phase[, expr, interpolation, ...])	Convinient function to show values of expression for given phase only from Grid Data.
show_path_data(ptpath, phase[, expr, label, ...])	Show values of expression for given phase calculated along PTpath.
show_path_modes(ptpath[, exclude, cmap])	Show stacked area diagram of phase modes along PT path
show_status([label, skiplabels, labelfs])	Shows status of grid calculations

Attributes

endmembers	Returns dictionary with phases and their end-members names
gridded	True when compositional grid(s) is calculated, otherwise False
gridxstep	
gridystep	
keys	Returns set of all existing multivariant fields.
name	Get project directory name.
phases	Returns set of all phases present in pseudosection
ratio	
variance	Returns dictionary of variances
x_var	
xrange	
y_var	
yrange	

5.1.2 pypsbuilder.psexplorer.TXPS

class pypsbuilder.psexplorer.**TXPS**(**args*, ***kwargs*)

Bases: PS

Class to postprocess txbuilder project

__init__(**args*, ***kwargs*)

Create PTPS class instance from builder project file.

Parameters

- **projfile** (*str, Path*) – psbuilder project file or files

- **tolerance** (*float*) – if not None, simplification tolerance. Default None
- **origwd** (*bool*) – If True TCAPI uses original stored working directory Default False.

Methods

__init__(*args, **kwargs)	Create PTPS class instance from builder project file.
add_overlay(ax[, fc, ec, label, skiplabels, ...])	
calculate_composition([nx, ny])	Method to calculate compositional variations on grid.
check_phase_expr(phase, expr)	
collect_all_data_keys()	Collect all phases and variables calculated on grid.
collect_data(key, phase, expr[, which])	Convinient function to retrieve values of expression for given phase for user-defined combination of results of divariant field identified by key.
collect_grid_data(key, phase, expr)	Retrieve values of expression for given phase for all GridData points within divariant field identified by key.
collect_inv_data(key, phase, expr)	Retrieve value of variables based expression for given phase for all invariant points surrounding divariant field identified by key.
collect_uni_data(key, phase, expr)	Retrieve values of expression for given phase for all univariant lines surrounding divariant field identified by key.
common_grid_and_masks(**kwargs)	Initialize common grid and mask for all partial grids
create_masks()	Update grid masks from existing divariant fields
fix_solutions()	Method try to find solution for grid points with failed status.
format_coord(x, y)	
gendrawpd([export_areas])	Method to write drawpd file
get_gridded(phase[, expr, which, smooth])	
get_grids(phase[, expr])	Convinient function to get dictionary of grids.
get_nearest_grid_data(x, y)	Retrieve nearest results from GridData to point.
get_section_id(x, y)	Return index of pseudosection and grid containing point
gidentify([label, skiplabels, labelfs])	Visual version of *identify* method.
ginput_path([label, skiplabels, labelfs])	Collect Path data by mouse digitizing.
glabel([label, skiplabels, labelfs])	Return formatted string of assamblage at PT point provided by mouse click.
identify(x, y)	Return key (frozenset) of divariant field for given temperature and pressure.
invs_from_unilist(ix, unilist)	Return set of IDs of invariant points associated with unilines.
isopleths(phase[, expr])	Method to draw compositional isopleths.
isopleths_vector(phase[, expr])	Method to draw vectorized compositional isopleths.
merge_data(phase, expr[, which])	Returns merged data obtained by *collect_data* method for all divariant fields.
onclick(event)	

continues on next page

Table 2 – continued from previous page

overlap_isopleths(*args, **kwargs)	Function to overlaped isopleths ranges.
pointcalc([label, skiplabels, labelfs])	
remove_grid_data(key, phase, expr)	Remove calculated data from grid and mark as failed
save()	Save gridded copositions and constructed divariant fields into psbuilder project file.
save_tab(comps[, tabfile])	Export gridded values to Perple_X tab format.
search_composition(*args[, interpolation, ...])	Function to plot root squared error of calculated and estimated values.
show(**kwargs)	Method to draw PT pseudosection.
show_data(key, phase[, expr, which])	Convinient function to show values of expression for given phase for user-defined combination of results of divariant field identified by key.
show_delta([label, pointsec, skiplabels, ...])	Shows THERMOCALC execution time for all grid points.
show_grid(phase[, expr, interpolation, ...])	Convinient function to show values of expression for given phase only from Grid Data.
show_status([label, skiplabels, labelfs])	Shows status of grid calculations

Attributes

endmembers	Returns dictionary with phases and their endmembers names
gridded	True when compositional grid(s) is calculated, otherwise False
gridxstep	
gridystep	
keys	Returns set of all existing multivariant fields.
name	Get project directory name.
phases	Returns set of all phases present in pseudosection
ratio	
variance	Returns dictionary of variances
x_var	
xrange	
y_var	
yrange	

5.1.3 pypsbuilder.psexplorer.PXPS

class pypsbuilder.psexplorer.**PXPS**(**args*, ***kwargs*)

Bases: PS

Class to postprocess pxbuilder project

__init__(**args*, ***kwargs*)

Create PTPS class instance from builder project file.

Parameters

- **projfile** (*str, Path*) – psbuilder project file or files
- **tolerance** (*float*) – if not None, simplification tolerance. Default None
- **origwd** (*bool*) – If True TCAPI uses original stored working directory Default False.

Methods

__init__(*args, **kwargs)	Create PTPS class instance from builder project file.
add_overlay(ax[, fc, ec, label, skiplabels, ...])	
calculate_composition([nx, ny])	Method to calculate compositional variations on grid.
check_phase_expr(phase, expr)	
collect_all_data_keys()	Collect all phases and variables calculated on grid.
collect_data(key, phase, expr[, which])	Convinient function to retrieve values of expression for given phase for user-defined combination of results of divariant field identified by key.
collect_grid_data(key, phase, expr)	Retrieve values of expression for given phase for all GridData points within divariant field identified by key.
collect_inv_data(key, phase, expr)	Retrieve value of variables based expression for given phase for all invariant points surrounding divariant field identified by key.
collect_uni_data(key, phase, expr)	Retrieve values of expression for given phase for all univariant lines surrounding divariant field identified by key.
common_grid_and_masks(**kwargs)	Initialize common grid and mask for all partial grids
create_masks()	Update grid masks from existing divariant fields
fix_solutions()	Method try to find solution for grid points with failed status.
format_coord(x, y)	
gendrawpd([export_areas])	Method to write drawpd file
get_gridded(phase[, expr, which, smooth])	
get_grids(phase[, expr])	Convinient function to get dictionary of grids.
get_nearest_grid_data(x, y)	Retrieve nearest results from GridData to point.
get_section_id(x, y)	Return index of pseudosection and grid containing point
gidentify([label, skiplabels, labelfs])	Visual version of *identify* method.
ginput_path([label, skiplabels, labelfs])	Collect Path data by mouse digitizing.

continues on next page

Table 3 – continued from previous page

glabel([label, skiplabels, labelfs])	Return formatted string of assamblage at PT point provided by mouse click.
identify(x, y)	Return key (frozenset) of divariant field for given temperature and pressure.
invs_from_unilist(ix, unilist)	Return set of IDs of invariant points associated with unilines.
isopleths(phase[, expr])	Method to draw compositional isopleths.
isopleths_vector(phase[, expr])	Method to draw vectorized compositional isopleths.
merge_data(phase, expr[, which])	Returns merged data obtained by *collect_data* method for all divariant fields.
onclick(event)	
overlap_isopleths(*args, **kwargs)	Function to overlaped isopleths ranges.
pointcalc([label, skiplabels, labelfs])	
remove_grid_data(key, phase, expr)	Remove calculated data from grid and mark as failed
save()	Save gridded copositions and constructed divariant fields into psbuilder project file.
save_tab(comps[, tabfile])	Export gridded values to Perple_X tab format.
search_composition(*args[, interpolation, ...])	Function to plot root squared error of calculated and estimated values.
show(**kwargs)	Method to draw PT pseudosection.
show_data(key, phase[, expr, which])	Convinient function to show values of expression for given phase for user-defined combination of results of divariant field identified by key.
show_delta([label, pointsec, skiplabels, ...])	Shows THERMOCALC execution time for all grid points.
show_grid(phase[, expr, interpolation, ...])	Convinient function to show values of expression for given phase only from Grid Data.
show_status([label, skiplabels, labelfs])	Shows status of grid calculations

Attributes

endmembers	Returns dictionary with phases and their endmembers names
gridded	True when compositional grid(s) is calculated, otherwise False
gridxstep	
gridystep	
keys	Returns set of all existing multivariant fields.
name	Get project directory name.
phases	Returns set of all phases present in pseudosection
ratio	
variance	Returns dictionary of variances
x_var	
xrange	
y_var	
yrange	

class pypsbuilder.psexplorer.**PTPS**(**args*, ***kwargs*)

Bases: PS

Class to postprocess ptbuilder project

calculate_composition(*nx=50*, *ny=50*)

Method to calculate compositional variations on grid.

A compositions are calculated for stable assemblages in regular grid covering pT range of pseudosection. A stable assemblage is identified from constructed divariant fields. Results are stored in *grid* property as *GridData* instance. A property *all_data_keys* is updated.

Before any grid point calculation, ptguesses are updated from nearest invariant point. If calculation fails, nearest solution from univariant line is used to update ptguesses. Finally, if solution is still not found, the method *fix_solutions* is called and neigbouring grid calculations are used to provide ptguess.

Parameters

- **nx** (*int*) – Number of grid points along x direction (T)
- **ny** (*int*) – Number of grid points along y direction (p)

fix_solutions()

Method try to find solution for grid points with failed status.

Ptguesses are used from successfully calculated neighboring points until solution is find. Otherwise ststus remains failed.

collect_ptpath(*tpath*, *ppath*, *N=100*, *kind='quadratic'*)

Method to collect THERMOCALC calculations along defined PT path.

PT path is interpolated from provided points using defined method. For each point THERMOCALC seek for solution using ptguess from nearest *GridData* point.

Parameters

- **tpath** (*numpy.array*) – 1D array of temperatures for given PT path
- **ppath** (*numpy.array*) – 1D array of pressures for given PT path
- **N** (*int*) – Number of calculation steps. Default 100.
- **kind** (*str*) – Kind of interpolation. See scipy.interpolate.interp1d

Returns

returns instance of PTpath class storing all calculations
along PT path.

Return type
PTpath

show_path_data(*ptpath*, *phase*, *expr=None*, *label=False*, *pathwidth=4*, *allpath=True*, *skiplabels=0*, *labelfs=6*)

Show values of expression for given phase calculated along PTpath.

It plots colored strip on PT space. Strips arenot drawn accross fields, where 'phase' is not present.

Parameters

- **ptpath** (*PTpath*) – Results obtained by *collect_ptpath* method.
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6
- **pathwidth** (*int*) – Width of colored strip. Default 4.
- **allpath** (*bool*) – Whether to plot full PT path (dashed line).

show_path_modes(*ptpath*, *exclude=[]*, *cmap='tab20'*)

Show stacked area diagram of phase modes along PT path

Parameters

- **ptpath** (*PTpath*) – Results obtained by *collect_ptpath* method.
- **exclude** (*list*) – List of phases to exclude. Included phases area normalized to 100%.
- **cmap** (*str*) – matplotlib colormap. Default 'tab20'

collect_all_data_keys()

Collect all phases and variables calculated on grid.

Result is stored in *all_data_keys* property as dictionary of dictionaries.

Example

To get list of all variables calculated for phase 'g' or end-member 'g(alm)' use:

```
>>> pt.all_data_keys['g']
['mode', 'x', 'z', 'm', 'f', 'xMgX', 'xFeX', 'xMnX', 'xCaX', 'xAlY',
'xFe3Y', 'H2O', 'SiO2', 'Al2O3', 'CaO', 'MgO', 'FeO', 'K2O', 'Na2O',
'TiO2', 'MnO', 'O', 'factor', 'G', 'H', 'S', 'V', 'rho']
>>> pt.all_data_keys['g(alm)']
['ideal', 'gamma', 'activity', 'prop', 'mu', 'RTlna']
```

collect_data(*key*, *phase*, *expr*, *which=7*)

Convinient function to retrieve values of expression for given phase for user-defined combination of results of divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

collect_grid_data(*key*, *phase*, *expr*)

Retrieve values of expression for given phase for all GridData points within divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

collect_inv_data(*key*, *phase*, *expr*)

Retrieve value of variables based expression for given phase for all invariant points surrounding divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

collect_uni_data(*key*, *phase*, *expr*)

Retrieve values of expression for given phase for all univariant lines surrounding divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named

- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

common_grid_and_masks(***kwargs*)

Initialize common grid and mask for all partial grids

create_masks()

Update grid masks from existing divariant fields

property endmembers

Returns dictionary with phases and their end-members names

gendrawpd(*export_areas=True*)

Method to write drawpd file

Parameters

export_areas (*bool*) – Whether to include constructed areas. Default True.

get_grids(*phase*, *expr=None*)

Convinient function to get dictionary of grids.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

get_nearest_grid_data(*x*, *y*)

Retrieve nearest results from GridData to point.

Parameters

- **x** (*float*) – x-coordinate of point
- **y** – y-coordiante of point

get_section_id(*x*, *y*)

Return index of pseudosection and grid containing point

gidentify(*label=False*, *skiplabels=0*, *labelfs=6*)

Visual version of *identify* method. PT point is provided by mouse click.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

ginput_path(*label=False*, *skiplabels=0*, *labelfs=6*)

Collect Path data by mouse digitizing.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

glabel(*label=False*, *skiplabels=0*, *labelfs=6*)

Return formatted string of assamblage at PT point provided by mouse click.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

property gridded

True when compositional grid(s) is calculated, otherwise False

identify(*x*, *y*)

Return key (frozenset) of divariant field for given temperature and pressure.

Parameters

- **x** (*float*) – x coord
- **y** (*float*) – y coord

invs_from_unilist(*ix*, *unilist*)

Return set of IDs of invariant points associated with unilines. lines.

Parameters
unilist (*iterable*) – list of (section_id, uni_id) pairs

Returns
set of associated invariant points

Return type
set

isopleths(*phase*, *expr=None*, ***kwargs*)

Method to draw compositional isopleths.

Isopleths are drawn as contours for values evaluated from provided expression. Individual divariant fields are contoured separately, so final plot allows sharp changes accross univariant lines. Within divariant field the selected interpolation is used.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **N** (*int*) – Max number of contours. Default 10.
- **step** (*int*) – Step between contour levels. If defined, N is ignored. Default None.
- **cdf** (*bool*) – When True contour levels are percentile based. Default False.
- **alpha** (*float*) – Alpha value for filled contours. Default 1
- **levels** (*list*) – User-defined contour levels. If defined, N and step is ignored.
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points. Default 7 (all data)
- **colorbar** (*bool*) – Whether to show colorbar. Default True
- **method** – Interpolation method. Default is 'rbf', other option is 'quadratic', which uses least-square fit to quadratic surface or 'spline' for bivariate spline interpolation.

- **rbf_func** – Default 'thin_plate'. See scipy.interpolation.Rbf
- **smooth** (*int*) – Values greater than zero increase the smoothness of the approximation. 0 is for interpolation (default), the function will always go through the nodal points.
- **epsilon** (*int*) – Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start).
- **degree** (*int*) – Degrees of the bivariate spline. Default is 3.
- **refine** (*int*) – Degree of grid refinement. Default 1
- **filter_outliers** (*bool*) – Whether to filter outliers. Defaut False.
- **filled** (*bool*) – Whether to contours should be filled. Defaut True.
- **filled_over** (*bool*) – Whether to overlay contourline over filled contours. Defaut False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used for contours. Default 'viridis'.
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **labelkeys** (*frozenset or list*) – Keys of divariant fields where contours should be labeled.
- **nosplit** (*bool*) – Controls whether the contour underlying labels are removed or not. Defaut True
- **gradient** (*bool*) – Whether the first derivate of values should be used. Default False.
- **dt** (*bool*) – Whether the gradient should be calculated along temperature or pressure. Default True.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

isopleths_vector(*phase*, *expr=None*, ***kwargs*)

Method to draw vectorized compositional isopleths.

Isopleths are drawn as lines for values evaluated from provided expression. Individual divariant fields are contoured separately, so final plot allows sharp changes accross univariant lines. Within divariant field the selected interpolation is used.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **N** (*int*) – Max number of contours. Default 10.
- **step** (*int*) – Step between contour levels. If defined, N is ignored. Default None.
- **cdf** (*bool*) – When True contour levels are percentile based. Default False.

- **levels** (*list*) – User-defined contour levels. If defined, N and step is ignored.
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points. Default 7 (all data)
- **colorbar** (*bool*) – Whether to show colorbar. Default True
- **method** – Interpolation method. Default is 'rbf', other option is 'quadratic', which uses least-square fit to quadratic surface.
- **rbf_func** – Default 'thin_plate'. See scipy.interpolation.Rbf
- **smooth** (*int*) – Values greater than zero increase the smoothness of the approximation. 0 is for interpolation (default), the function will always go through the nodal points.
- **epsilon** (*int*) – Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start).
- **degree** (*int*) – Degrees of the bivariate spline. Default is 3.
- **refine** (*int*) – Degree of grid refinement. Default 1
- **filter_outliers** (*bool*) – Whether to filter outliers. Defaut False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **overlay** (*bool*) – Whether to show assemblage fields. Default True
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used isopleths coloring. Default 'viridis'.
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **lw** (*float*) – Linewidth of isopleths. Default 1.0
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **color** – matplotlib color for isopleths. If None cmap is used. Default None.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

property keys

Returns set of all existing multivariant fields. Fields are identified by frozenset of present phases called key.

merge_data(*phase*, *expr*, *which=7*)

Returns merged data obtained by *collect_data* method for all divariant fields.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

property name

Get project directory name.

overlap_isopleths(**args*, ***kwargs*)

Function to overlaped isopleths ranges.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **levels** (*tuple*) – tuple of min an max values for isopleth band
- **alpha** (*float*) – Alpha value for filled contours. Default 1
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

Example

```
>>> pt.overlap_isopleths(
        'g', 'xMgX', (0.07, 0.13),
        'g', 'xFeX', (0.51, 0.65),
        'g', 'xCaX', (0.06, 0.15),
        'g', 'xMnX', (0.16, 0.27),
    )
```

property phases

Returns set of all phases present in pseudosection

remove_grid_data(*key*, *phase*, *expr*)

Remove calculated data from grid and mark as failed

Note: Click on all data to be discarded by mouse and finish be Enter

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

save()

Save gridded copositions and constructed divariant fields into psbuilder project file.

Note that once project is edited with psbuilder, calculated compositions are removed and need to be recalculated using *PTPS.calculate_composition* method.

save_tab(*comps*, *tabfile=None*)

Export gridded values to Perple_X tab format. Could be used in pywerami.

Note: Gridded values are interpolated from raw results

Parameters

- **comps** (*list*) – List of (phase, expr) tuples. phase (str): Phase or end-member named expr (str): Expression to evaluate. It could use any

  variable existing for given phase. Check *all_data_keys* property for possible variables.

- **tabfile** (*str*) – filename for tabfile. Default pseudosection name.tab

search_composition(**args*, *interpolation=None*, *label=False*, *skiplabels=0*, *labelfs=6*, *geterror=False*, *getpt=False*, *fig=None*, *fig_kw={}*, *isopleths=False*, *which=7*, *smooth=0*, *alpha=0.5*, *lw=2*)

Function to plot root squared error of calculated and estimated values.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **val** (*float*) – searched value
- **interpolation** (*str*) – matplotlib imshow interpolation method. Default None.
- **label** (*bool*) – Whether to label divariant fields. Default False.
- **geterror** (*bool*) – When True, calculated RMSE array is returned. Otherwise error is plotted. Deafult False
- **getpt** (*bool*) – When True return tuple of (p, T, err) where error is minimal. Default False
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **isopleths** (*bool*) – When True, searched isopleths are shown. Default False
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

Example

```
>>> pt.search_composition(
        'g', 'xMgX', 0.076,
        'g', 'xFeX', 0.525,
        'g', 'xCaX', 0.126,
        'g', 'xMnX', 0.268
    )
```

show(***kwargs*)

Method to draw PT pseudosection.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used to divariant fields coloring. Colors are based on variance. Default 'Purples'.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **alpha** (*float*) – alpha value for colors. Default 0.6
- **connect** (*bool*) – Whether mouse click echo stable assemblage to STDOUT. Default False.
- **show_vertices** (*bool*) – Whether to show vertices of drawn areas. Default False.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

show_data(*key*, *phase*, *expr=None*, *which=7*)

Convinient function to show values of expression for given phase for user-defined combination of results of divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

show_delta(*label=False*, *pointsec=False*, *skiplabels=0*, *labelfs=6*)

Shows THERMOCALC execution time for all grid points.

Parameters

- **pointsec** (*bool*) – Whether to show points/sec or secs/point. Default False.
- **label** (*bool*) – Whether to label divariant fields. Default False.

show_grid(*phase*, *expr=None*, *interpolation=None*, *label=False*, *skiplabels=0*, *labelfs=6*)

Convinient function to show values of expression for given phase only from Grid Data.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

- **interpolation** (*str*) – matplotlib imshow interpolation method. Default None.
- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

show_status(*label=False*, *skiplabels=0*, *labelfs=6*)

Shows status of grid calculations

property variance

Returns dictionary of variances

class pypsbuilder.psexplorer.**TXPS**(**args*, ***kwargs*)

Bases: PS

Class to postprocess txbuilder project

calculate_composition(*nx=50*, *ny=50*)

Method to calculate compositional variations on grid.

A compositions are calculated for stable assemblages in regular grid covering pT range of pseudosection. A stable assemblage is identified from constructed divariant fields. Results are stored in *grid* property as *GridData* instance. A property *all_data_keys* is updated.

Before any grid point calculation, ptguesses are updated from nearest invariant point. If calculation fails, nearest solution from univariant line is used to update ptguesses. Finally, if solution is still not found, the method *fix_solutions* is called and neigbouring grid calculations are used to provide ptguess.

Parameters

- **nx** (*int*) – Number of grid points along x direction (T)
- **ny** (*int*) – Number of grid points along y direction (p)

fix_solutions()

Method try to find solution for grid points with failed status.

Ptguesses are used from successfully calculated neighboring points until solution is find. Otherwise ststus remains failed.

collect_all_data_keys()

Collect all phases and variables calculated on grid.

Result is stored in *all_data_keys* property as dictionary of dictionaries.

Example

To get list of all variables calculated for phase ‘g’ or end-member ‘g(alm)’ use:

```
>>> pt.all_data_keys['g']
['mode', 'x', 'z', 'm', 'f', 'xMgX', 'xFeX', 'xMnX', 'xCaX', 'xAlY',
'xFe3Y', 'H2O', 'SiO2', 'Al2O3', 'CaO', 'MgO', 'FeO', 'K2O', 'Na2O',
'TiO2', 'MnO', 'O', 'factor', 'G', 'H', 'S', 'V', 'rho']
>>> pt.all_data_keys['g(alm)']
['ideal', 'gamma', 'activity', 'prop', 'mu', 'RTlna']
```

collect_data(*key*, *phase*, *expr*, *which=7*)

Convinient function to retrieve values of expression for given phase for user-defined combination of results of divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

collect_grid_data(*key*, *phase*, *expr*)

Retrieve values of expression for given phase for all GridData points within divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

collect_inv_data(*key*, *phase*, *expr*)

Retrieve value of variables based expression for given phase for all invariant points surrounding divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

collect_uni_data(*key*, *phase*, *expr*)

Retrieve values of expression for given phase for all univariant lines surrounding divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

common_grid_and_masks(***kwargs*)

Initialize common grid and mask for all partial grids

create_masks()

Update grid masks from existing divariant fields

property endmembers

Returns dictionary with phases and their end-members names

gendrawpd(*export_areas=True*)

Method to write drawpd file

Parameters

export_areas (*bool*) – Whether to include constructed areas. Default True.

get_grids(*phase*, *expr=None*)

Convinient function to get dictionary of grids.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

get_nearest_grid_data(*x*, *y*)

Retrieve nearest results from GridData to point.

Parameters

- **x** (*float*) – x-coordinate of point
- **y** – y-coordiante of point

get_section_id(*x*, *y*)

Return index of pseudosection and grid containing point

gidentify(*label=False*, *skiplabels=0*, *labelfs=6*)

Visual version of *identify* method. PT point is provided by mouse click.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

ginput_path(*label=False*, *skiplabels=0*, *labelfs=6*)

Collect Path data by mouse digitizing.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

glabel(*label=False*, *skiplabels=0*, *labelfs=6*)

Return formatted string of assamblage at PT point provided by mouse click.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

property gridded

True when compositional grid(s) is calculated, otherwise False

identify(*x*, *y*)

Return key (frozenset) of divariant field for given temperature and pressure.

Parameters

- **x** (*float*) – x coord
- **y** (*float*) – y coord

invs_from_unilist(*ix*, *unilist*)

Return set of IDs of invariant points associated with unilines. lines.

Parameters
unilist (*iterable*) – list of (section_id, uni_id) pairs

Returns
set of associated invariant points

Return type
set

isopleths(*phase*, *expr=None*, ***kwargs*)

Method to draw compositional isopleths.

Isopleths are drawn as contours for values evaluated from provided expression. Individual divariant fields are contoured separately, so final plot allows sharp changes accross univariant lines. Within divariant field the selected interpolation is used.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **N** (*int*) – Max number of contours. Default 10.
- **step** (*int*) – Step between contour levels. If defined, N is ignored. Default None.
- **cdf** (*bool*) – When True contour levels are percentile based. Default False.
- **alpha** (*float*) – Alpha value for filled contours. Default 1
- **levels** (*list*) – User-defined contour levels. If defined, N and step is ignored.
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points. Default 7 (all data)
- **colorbar** (*bool*) – Whether to show colorbar. Default True
- **method** – Interpolation method. Default is 'rbf', other option is 'quadratic', which uses least-square fit to quadratic surface or 'spline' for bivariate spline interpolation.
- **rbf_func** – Default 'thin_plate'. See scipy.interpolation.Rbf
- **smooth** (*int*) – Values greater than zero increase the smoothness of the approximation. 0 is for interpolation (default), the function will always go through the nodal points.
- **epsilon** (*int*) – Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start).
- **degree** (*int*) – Degrees of the bivariate spline. Default is 3.
- **refine** (*int*) – Degree of grid refinement. Default 1
- **filter_outliers** (*bool*) – Whether to filter outliers. Defaut False.

- **filled** (*bool*) – Whether to contours should be filled. Defaut True.
- **filled_over** (*bool*) – Whether to overlay contourline over filled contours. Defaut False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used for contours. Default 'viridis'.
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **labelkeys** (*frozenset or list*) – Keys of divariant fields where contours should be labeled.
- **nosplit** (*bool*) – Controls whether the contour underlying labels are removed or not. Defaut True
- **gradient** (*bool*) – Whether the first derivate of values should be used. Default False.
- **dt** (*bool*) – Whether the gradient should be calculated along temperature or pressure. Default True.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

isopleths_vector(*phase*, *expr=None*, ***kwargs*)

Method to draw vectorized compositional isopleths.

Isopleths are drawn as lines for values evaluated from provided expression. Individual divariant fields are contoured separately, so final plot allows sharp changes accross univariant lines. Within divariant field the selected interpolation is used.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **N** (*int*) – Max number of contours. Default 10.
- **step** (*int*) – Step between contour levels. If defined, N is ignored. Default None.
- **cdf** (*bool*) – When True contour levels are percentile based. Default False.
- **levels** (*list*) – User-defined contour levels. If defined, N and step is ignored.
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points. Default 7 (all data)
- **colorbar** (*bool*) – Whether to show colorbar. Default True
- **method** – Interpolation method. Default is 'rbf', other option is 'quadratic', which uses least-square fit to quadratic surface.
- **rbf_func** – Default 'thin_plate'. See scipy.interpolation.Rbf

- **smooth** (*int*) – Values greater than zero increase the smoothness of the approximation. 0 is for interpolation (default), the function will always go through the nodal points.
- **epsilon** (*int*) – Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start).
- **degree** (*int*) – Degrees of the bivariate spline. Default is 3.
- **refine** (*int*) – Degree of grid refinement. Default 1
- **filter_outliers** (*bool*) – Whether to filter outliers. Defaut False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **overlay** (*bool*) – Whether to show assemblage fields. Default True
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used isopleths coloring. Default 'viridis'.
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **lw** (*float*) – Linewidth of isopleths. Default 1.0
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **color** – matplotlib color for isopleths. If None cmap is used. Default None.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

property keys

Returns set of all existing multivariant fields. Fields are identified by frozenset of present phases called key.

merge_data(*phase*, *expr*, *which=7*)

Returns merged data obtained by *collect_data* method for all divariant fields.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

property name

Get project directory name.

overlap_isopleths(**args*, ***kwargs*)

Function to overlaped isopleths ranges.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

- **levels** (*tuple*) – tuple of min an max values for isopleth band
- **alpha** (*float*) – Alpha value for filled contours. Default 1
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

Example

```
>>> pt.overlap_isopleths(
        'g', 'xMgX', (0.07, 0.13),
        'g', 'xFeX', (0.51, 0.65),
        'g', 'xCaX', (0.06, 0.15),
        'g', 'xMnX', (0.16, 0.27),
    )
```

property phases

Returns set of all phases present in pseudosection

remove_grid_data(*key*, *phase*, *expr*)

Remove calculated data from grid and mark as failed

Note: Click on all data to be discarded by mouse and finish be Enter

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

save()

Save gridded copositions and constructed divariant fields into psbuilder project file.

Note that once project is edited with psbuilder, calculated compositions are removed and need to be recalculated using *PTPS.calculate_composition* method.

save_tab(*comps*, *tabfile=None*)

Export gridded values to Perple_X tab format. Could be used in pywerami.

Note: Gridded values are interpolated from raw results

Parameters

- **comps** (*list*) – List of (phase, expr) tuples. phase (str): Phase or end-member named expr (str): Expression to evaluate. It could use any

  variable existing for given phase. Check *all_data_keys* property for possible variables.
- **tabfile** (*str*) – filename for tabfile. Default pseudosection name.tab

search_composition(**args*, *interpolation=None*, *label=False*, *skiplabels=0*, *labelfs=6*, *geterror=False*, *getpt=False*, *fig=None*, *fig_kw={}*, *isopleths=False*, *which=7*, *smooth=0*, *alpha=0.5*, *lw=2*)

Function to plot root squared error of calculated and estimated values.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **val** (*float*) – searched value
- **interpolation** (*str*) – matplotlib imshow interpolation method. Default None.
- **label** (*bool*) – Whether to label divariant fields. Default False.
- **geterror** (*bool*) – When True, calculated RMSE array is returned. Otherwise error is plotted. Deafult False
- **getpt** (*bool*) – When True return tuple of (p, T, err) where error is minimal. Default False
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **isopleths** (*bool*) – When True, searched isopleths are shown. Default False
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

Example

```
>>> pt.search_composition(
        'g', 'xMgX', 0.076,
        'g', 'xFeX', 0.525,
        'g', 'xCaX', 0.126,
        'g', 'xMnX', 0.268
    )
```

show(***kwargs*)

Method to draw PT pseudosection.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used to divariant fields coloring. Colors are based on variance. Default 'Purples'.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **alpha** (*float*) – alpha value for colors. Default 0.6
- **connect** (*bool*) – Whether mouse click echo stable assemblage to STDOUT. Default False.
- **show_vertices** (*bool*) – Whether to show vertices of drawn areas. Default False.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

show_data(*key*, *phase*, *expr=None*, *which=7*)

Convinient function to show values of expression for given phase for user-defined combination of results of divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

show_delta(*label=False*, *pointsec=False*, *skiplabels=0*, *labelfs=6*)

Shows THERMOCALC execution time for all grid points.

Parameters

- **pointsec** (*bool*) – Whether to show points/sec or secs/point. Default False.
- **label** (*bool*) – Whether to label divariant fields. Default False.

show_grid(*phase*, *expr=None*, *interpolation=None*, *label=False*, *skiplabels=0*, *labelfs=6*)

Convinient function to show values of expression for given phase only from Grid Data.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **interpolation** (*str*) – matplotlib imshow interpolation method. Default None.
- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

show_status(*label=False*, *skiplabels=0*, *labelfs=6*)

Shows status of grid calculations

property variance

Returns dictionary of variances

class pypsbuilder.psexplorer.**PXPS**(**args*, ***kwargs*)

Bases: PS

Class to postprocess pxbuilder project

calculate_composition(*nx=50*, *ny=50*)

Method to calculate compositional variations on grid.

A compositions are calculated for stable assemblages in regular grid covering pT range of pseudosection. A stable assemblage is identified from constructed divariant fields. Results are stored in *grid* property as *GridData* instance. A property *all_data_keys* is updated.

Before any grid point calculation, ptguesses are updated from nearest invariant point. If calculation fails, nearest solution from univariant line is used to update ptguesses. Finally, if solution is still not found, the method *fix_solutions* is called and neigbouring grid calculations are used to provide ptguess.

Parameters

- **nx** (*int*) – Number of grid points along x direction (T)
- **ny** (*int*) – Number of grid points along y direction (p)

fix_solutions()

Method try to find solution for grid points with failed status.

Ptguesses are used from successfully calculated neighboring points until solution is find. Otherwise ststus remains failed.

collect_all_data_keys()

Collect all phases and variables calculated on grid.

Result is stored in *all_data_keys* property as dictionary of dictionaries.

Example

To get list of all variables calculated for phase 'g' or end-member 'g(alm)' use:

```
>>> pt.all_data_keys['g']
['mode', 'x', 'z', 'm', 'f', 'xMgX', 'xFeX', 'xMnX', 'xCaX', 'xAlY',
'xFe3Y', 'H2O', 'SiO2', 'Al2O3', 'CaO', 'MgO', 'FeO', 'K2O', 'Na2O',
'TiO2', 'MnO', 'O', 'factor', 'G', 'H', 'S', 'V', 'rho']
>>> pt.all_data_keys['g(alm)']
['ideal', 'gamma', 'activity', 'prop', 'mu', 'RTlna']
```

collect_data(*key*, *phase*, *expr*, *which=7*)

Convinient function to retrieve values of expression for given phase for user-defined combination of results of divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

collect_grid_data(*key*, *phase*, *expr*)

Retrieve values of expression for given phase for all GridData points within divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

collect_inv_data(*key*, *phase*, *expr*)

Retrieve value of variables based expression for given phase for all invariant points surrounding divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

collect_uni_data(*key*, *phase*, *expr*)

Retrieve values of expression for given phase for all univariant lines surrounding divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

common_grid_and_masks(***kwargs*)

Initialize common grid and mask for all partial grids

create_masks()

Update grid masks from existing divariant fields

property endmembers

Returns dictionary with phases and their end-members names

gendrawpd(*export_areas=True*)

Method to write drawpd file

Parameters
export_areas (*bool*) – Whether to include constructed areas. Default True.

get_grids(*phase*, *expr=None*)

Convinient function to get dictionary of grids.

Parameters

- **phase** (*str*) – Phase or end-member named

- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

get_nearest_grid_data(*x*, *y*)

Retrieve nearest results from GridData to point.

Parameters

- **x** (*float*) – x-coordinate of point
- **y** – y-coordiante of point

get_section_id(*x*, *y*)

Return index of pseudosection and grid containing point

gidentify(*label=False*, *skiplabels=0*, *labelfs=6*)

Visual version of *identify* method. PT point is provided by mouse click.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

ginput_path(*label=False*, *skiplabels=0*, *labelfs=6*)

Collect Path data by mouse digitizing.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

glabel(*label=False*, *skiplabels=0*, *labelfs=6*)

Return formatted string of assamblage at PT point provided by mouse click.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

property gridded

True when compositional grid(s) is calculated, otherwise False

identify(*x*, *y*)

Return key (frozenset) of divariant field for given temperature and pressure.

Parameters

- **x** (*float*) – x coord
- **y** (*float*) – y coord

invs_from_unilist(*ix*, *unilist*)

Return set of IDs of invariant points associated with unilines. lines.

Parameters

unilist (*iterable*) – list of (section_id, uni_id) pairs

Returns
set of associated invariant points

Return type
set

isopleths(*phase*, *expr=None*, ***kwargs*)

Method to draw compositional isopleths.

Isopleths are drawn as contours for values evaluated from provided expression. Individual divariant fields are contoured separately, so final plot allows sharp changes accross univariant lines. Within divariant field the selected interpolation is used.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **N** (*int*) – Max number of contours. Default 10.
- **step** (*int*) – Step between contour levels. If defined, N is ignored. Default None.
- **cdf** (*bool*) – When True contour levels are percentile based. Default False.
- **alpha** (*float*) – Alpha value for filled contours. Default 1
- **levels** (*list*) – User-defined contour levels. If defined, N and step is ignored.
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points. Default 7 (all data)
- **colorbar** (*bool*) – Whether to show colorbar. Default True
- **method** – Interpolation method. Default is 'rbf', other option is 'quadratic', which uses least-square fit to quadratic surface or 'spline' for bivariate spline interpolation.
- **rbf_func** – Default 'thin_plate'. See scipy.interpolation.Rbf
- **smooth** (*int*) – Values greater than zero increase the smoothness of the approximation. 0 is for interpolation (default), the function will always go through the nodal points.
- **epsilon** (*int*) – Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start).
- **degree** (*int*) – Degrees of the bivariate spline. Default is 3.
- **refine** (*int*) – Degree of grid refinement. Default 1
- **filter_outliers** (*bool*) – Whether to filter outliers. Defaut False.
- **filled** (*bool*) – Whether to contours should be filled. Defaut True.
- **filled_over** (*bool*) – Whether to overlay contourline over filled contours. Defaut False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used for contours. Default 'viridis'.
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **labelkeys** (*frozenset or list*) – Keys of divariant fields where contours should be labeled.

- **nosplit** (*bool*) – Controls whether the contour underlying labels are removed or not. Defaut True
- **gradient** (*bool*) – Whether the first derivate of values should be used. Default False.
- **dt** (*bool*) – Whether the gradient should be calculated along temperature or pressure. Default True.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

isopleths_vector(*phase*, *expr=None*, ***kwargs*)

Method to draw vectorized compositional isopleths.

Isopleths are drawn as lines for values evaluated from provided expression. Individual divariant fields are contoured separately, so final plot allows sharp changes accross univariant lines. Within divariant field the selected interpolation is used.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **N** (*int*) – Max number of contours. Default 10.
- **step** (*int*) – Step between contour levels. If defined, N is ignored. Default None.
- **cdf** (*bool*) – When True contour levels are percentile based. Default False.
- **levels** (*list*) – User-defined contour levels. If defined, N and step is ignored.
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points. Default 7 (all data)
- **colorbar** (*bool*) – Whether to show colorbar. Default True
- **method** – Interpolation method. Default is 'rbf', other option is 'quadratic', which uses least-square fit to quadratic surface.
- **rbf_func** – Default 'thin_plate'. See scipy.interpolation.Rbf
- **smooth** (*int*) – Values greater than zero increase the smoothness of the approximation. 0 is for interpolation (default), the function will always go through the nodal points.
- **epsilon** (*int*) – Adjustable constant for gaussian or multiquadrics functions - defaults to approximate average distance between nodes (which is a good start).
- **degree** (*int*) – Degrees of the bivariate spline. Default is 3.
- **refine** (*int*) – Degree of grid refinement. Default 1
- **filter_outliers** (*bool*) – Whether to filter outliers. Defaut False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **overlay** (*bool*) – Whether to show assemblage fields. Default True

- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used isopleths coloring. Default 'viridis'.
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **lw** (*float*) – Linewidth of isopleths. Default 1.0
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **color** – matplotlib color for isopleths. If None cmap is used. Default None.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

property keys

Returns set of all existing multivariant fields. Fields are identified by frozenset of present phases called key.

merge_data(*phase*, *expr*, *which=7*)

Returns merged data obtained by *collect_data* method for all divariant fields.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

property name

Get project directory name.

overlap_isopleths(**args*, ***kwargs*)

Function to overlaped isopleths ranges.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **levels** (*tuple*) – tuple of min an max values for isopleth band
- **alpha** (*float*) – Alpha value for filled contours. Default 1
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

Example

```
>>> pt.overlap_isopleths(
        'g', 'xMgX', (0.07, 0.13),
        'g', 'xFeX', (0.51, 0.65),
        'g', 'xCaX', (0.06, 0.15),
        'g', 'xMnX', (0.16, 0.27),
    )
```

property phases

Returns set of all phases present in pseudosection

remove_grid_data(*key*, *phase*, *expr*)

Remove calculated data from grid and mark as failed

Note: Click on all data to be discarded by mouse and finish be Enter

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.

save()

Save gridded copositions and constructed divariant fields into psbuilder project file.

Note that once project is edited with psbuilder, calculated compositions are removed and need to be recalculated using *PTPS.calculate_composition* method.

save_tab(*comps*, *tabfile=None*)

Export gridded values to Perple_X tab format. Could be used in pywerami.

Note: Gridded values are interpolated from raw results

Parameters

- **comps** (*list*) – List of (phase, expr) tuples. phase (str): Phase or end-member named expr (str): Expression to evaluate. It could use any

  variable existing for given phase. Check *all_data_keys* property for possible variables.
- **tabfile** (*str*) – filename for tabfile. Default pseudosection name.tab

search_composition(**args*, *interpolation=None*, *label=False*, *skiplabels=0*, *labelfs=6*, *geterror=False*, *getpt=False*, *fig=None*, *fig_kw={}*, *isopleths=False*, *which=7*, *smooth=0*, *alpha=0.5*, *lw=2*)

Function to plot root squared error of calculated and estimated values.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **val** (*float*) – searched value
- **interpolation** (*str*) – matplotlib imshow interpolation method. Default None.
- **label** (*bool*) – Whether to label divariant fields. Default False.

- **geterror** (*bool*) – When True, calculated RMSE array is returned. Otherwise error is plotted. Deafult False
- **getpt** (*bool*) – When True return tuple of (p, T, err) where error is minimal. Default False
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **isopleths** (*bool*) – When True, searched isopleths are shown. Default False
- **which** (*int*) – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

Example

```
>>> pt.search_composition(
        'g', 'xMgX', 0.076,
        'g', 'xFeX', 0.525,
        'g', 'xCaX', 0.126,
        'g', 'xMnX', 0.268
    )
```

show(***kwargs*)

Method to draw PT pseudosection.

Parameters

- **label** (*bool*) – Whether to label divariant fields. Default False.
- **out** (*str or list*) – Highligt zero-mode lines for given phases.
- **high** (*frozenset or list*) – Highlight divariant fields identified by key(s).
- **cmap** (*str*) – matplotlib colormap used to divariant fields coloring. Colors are based on variance. Default 'Purples'.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6
- **bulk** (*bool*) – Whether to show bulk composition on top of diagram. Default False.
- **alpha** (*float*) – alpha value for colors. Default 0.6
- **connect** (*bool*) – Whether mouse click echo stable assemblage to STDOUT. Default False.
- **show_vertices** (*bool*) – Whether to show vertices of drawn areas. Default False.
- **fig** (*Figure*) – If not None, axes are added to fig and returned. Default None
- **fig_kw** – dict passed to subplots method.
- **ax** (*Axes*) – Axes to be used. Default None
- **filename** – If not None, figure is saved to file
- **save_kw** – dict passed to savefig method.
- **show** (*bool*) – When False, Axes are returned, otherwise plot is shown. Default True

show_data(*key*, *phase*, *expr=None*, *which=7*)

Convinient function to show values of expression for given phase for user-defined combination of results of divariant field identified by key.

Parameters

- **key** (*frozenset*) – Key identifying divariant field
- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **which** – Bitopt defining from where data are collected. 0 bit - invariant points, 1 bit - uniariant lines and 2 bit - GridData points

show_delta(*label=False*, *pointsec=False*, *skiplabels=0*, *labelfs=6*)

Shows THERMOCALC execution time for all grid points.

Parameters

- **pointsec** (*bool*) – Whether to show points/sec or secs/point. Default False.
- **label** (*bool*) – Whether to label divariant fields. Default False.

show_grid(*phase*, *expr=None*, *interpolation=None*, *label=False*, *skiplabels=0*, *labelfs=6*)

Convinient function to show values of expression for given phase only from Grid Data.

Parameters

- **phase** (*str*) – Phase or end-member named
- **expr** (*str*) – Expression to evaluate. It could use any variable existing for given phase. Check *all_data_keys* property for possible variables.
- **interpolation** (*str*) – matplotlib imshow interpolation method. Default None.
- **label** (*bool*) – Whether to label divariant fields. Default False.
- **skiplabels** (*float*) – Minimal area fraction of fields to be labelled
- **labelfs** (*float*) – Size of label font. Default 6

show_status(*label=False*, *skiplabels=0*, *labelfs=6*)

Shows status of grid calculations

property variance

Returns dictionary of variances

5.2 tcapi module

TC35API
TC34API

5.2.1 pypsbuilder.tcapi.TC35API

class pypsbuilder.tcapi.**TC35API**(*workdir*, *tcexe*, *drexe*, *encoding='mac-roman'*)

Bases: TCAPI

__init__(*workdir*, *tcexe*, *drexe*, *encoding='mac-roman'*)

Methods

__init__(workdir, tcexe, drexe[, encoding])	
calc_assemblage(phases, p, t[, onebulk])	Method to run THERMOCALC to calculate compositions of stable assemblage.
calc_p(phases, out, **kwargs)	Method to run THERMOCALC to find univariant line using Calc P at T strategy.
calc_pt(phases, out, **kwargs)	Method to run THERMOCALC to find invariant point.
calc_px(phases, out, **kwargs)	Method to run THERMOCALC for p-X pseudosection calculations.
calc_t(phases, out, **kwargs)	Method to run THERMOCALC to find univariant line using Calc T at P strategy.
calc_tx(phases, out, **kwargs)	Method to run THERMOCALC for T-X pseudosection calculations.
calc_variance(phases)	Get variance of assemblage.
dogmin(phases, p, t, variance[, doglevel, ...])	Run THERMOCALC dogmin session.
parse_dogmin()	Dogmin parser.
parse_logfile(**kwargs)	Parser for THERMOCALC 3.50beta output.
parse_logfile_backup(**kwargs)	
read_prefsfile()	
read_scriptfile()	
rundr()	Method to run drawpd.
runtc([instr])	Low-level method to actually run THERMOCALC.
update_scriptfile(**kwargs)	Method to update scriptfile.

Attributes

axfile	Path to used a-x file.
csvfile	Path to csv file.
dataset	Version identification of thermodynamic dataset in use.
datasetfile	Path to dataset file.
drawpdfile	Path to drawpd file.
drfile	Path to -dr output file.
icfile	Path to ic file.
itfile	Path to it file.
logfile	Path to THERMOCALC log file.
ofile	Path to project output file.
prefsfile	Path to THERMOCALC prefs file.
scriptfile	Path to scriptfile.
tcnewversion	False for THERMOCALC older than 3.5.
tcversion	THERMOCALC version string

5.2.2 pypsbuilder.tcapi.TC34API

class pypsbuilder.tcapi.**TC34API**(*workdir*, *tcexe*, *drexe*, *encoding='mac-roman'*)

Bases: TCAPI

__init__(*workdir*, *tcexe*, *drexe*, *encoding='mac-roman'*)

Methods

__init__(workdir, tcexe, drexe[, encoding])	
calc_assemblage(phases, p, t)	Method to run THERMOCALC to calculate compositions of stable assemblage.
calc_p(phases, out, **kwargs)	Method to run THERMOCALC to find univariant line using Calc P at T strategy.
calc_pt(phases, out, **kwargs)	Method to run THERMOCALC to find invariant point.
calc_px(phases, out, **kwargs)	Method to run THERMOCALC for P-X pseudosection calculations.
calc_t(phases, out, **kwargs)	Method to run THERMOCALC to find univariant line using Calc T at P strategy.
calc_tx(phases, out, **kwargs)	Method to run THERMOCALC for T-X pseudosection calculations.
calc_variance(phases)	Get variance of assemblage.
dogmin(phases, p, t, variance[, doglevel, ...])	Run THERMOCALC dogmin session.
interpolate_bulk(x)	
parse_dogmin()	Dogmin parser.
parse_kwargs(**kwargs)	
parse_logfile(**kwargs)	Parser for THERMOCALC 3.4x output.
read_prefsfile()	
read_scriptfile()	
rundr()	Method to run drawpd.
runtc([instr])	Low-level method to actually run THERMOCALC.
update_scriptfile(**kwargs)	Method to update scriptfile.

Attributes

axfile	Path to used a-x file.
csvfile	Path to csv file.
dataset	Version identification of thermodynamic dataset in use.
datasetfile	Path to dataset file.
drawpdfile	Path to drawpd file.
drfile	Path to -dr output file.
icfile	Path to ic file.
itfile	Path to it file.
logfile	Path to THERMOCALC log file.
ofile	Path to project output file.
prefsfile	Path to THERMOCALC prefs file.
scriptfile	Path to scriptfile.
tcnewversion	False for THERMOCALC older than 3.5.
tcversion	THERMOCALC version string

class pypsbuilder.tcapi.**TC35API**(*workdir*, *tcexe*, *drexe*, *encoding='mac-roman'*)

Bases: TCAPI

parse_logfile(***kwargs*)

Parser for THERMOCALC 3.50beta output.

It parses the outputs of THERMOCALC after calculation.

Parameters

- **output** (*str*) – When not None, used as content of logfile. Default None.
- **resic** (*str*) – When not None, used as content of icfile. Default None.
- **get_phases** (*bool*) – When true returns also tuple (phases, out, calcs). Default False

Returns

Result of parsing. 'ok', 'nir' (nothing in range) or 'bombed'. results (TCResultSet): Results of TC calculation. output (str): Full nonparsed THERMOCALC output.

Return type

status (str)

Example

Parse output after univariant line calculation in P-T pseudosection:

```
>>> tc = TCAPI('pat/to/dir')
>>> status, result, output = tc.parse_logfile()
```

update_scriptfile(***kwargs*)

Method to update scriptfile.

This method is used to programatically edit scriptfile.

Kwargs:

calcs: List of lines defining fully hands-off calculations. Default None. get_old_calcs: When True method returns existing calcs lines

before possible modification. Default False.

guesses: List of lines defining ptguesses. If None guesses

are not modified. Default None.

get_old_guesses: When True method returns existing ptguess lines

before possible modification. Default False.

bulk: List of lines defining bulk composition. Default None. xsteps: Number of compositional steps between two bulks.

Default 20.

calc_t(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC to find univariant line using Calc T at P strategy.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of single zero mode phase

- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_p(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC to find univariant line using Calc P at T strategy.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of single zero mode phase
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_pt(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC to find invariant point.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of two zero mode phases
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_tx(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC for T-X pseudosection calculations.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of zero mode phases
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation

- **xvals** (*tuple*) – range for X variable
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_px(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC for p-X pseudosection calculations.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of zero mode phases
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **xvals** (*tuple*) – range for X variable
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_assemblage(*phases*, *p*, *t*, *onebulk=None*)

Method to run THERMOCALC to calculate compositions of stable assemblage.

Parameters

- **phases** (*set*) – Set of present phases
- **p** (*float*) – Temperature for calculation
- **t** (*float*) – Pressure for calculation

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

dogmin(*phases*, *p*, *t*, *variance*, *doglevel=1*, *onebulk=None*)

Run THERMOCALC dogmin session.

Parameters
variance (*int*) – Maximum variance to be considered

Returns
THERMOCALC standard output

Return type
str

`calc_variance`(*phases*)

Get variance of assemblage.

Parameters
phases (*set*) – Set of present phases

Returns
variance

Return type
int

`property axfile`

Path to used a-x file.

Type
pathlib.Path

`property csvfile`

Path to csv file.

Type
pathlib.Path

`property dataset`

Version identification of thermodynamic dataset in use.

Type
str

`property datasetfile`

Path to dataset file.

Type
pathlib.Path

`property drawpdfile`

Path to drawpd file.

Type
pathlib.Path

`property drfile`

Path to -dr output file.

Type
pathlib.Path

`property icfile`

Path to ic file.

Type
pathlib.Path

`property itfile`

Path to it file.

Type
pathlib.Path

property logfile

Path to THERMOCALC log file.

Type
pathlib.Path

property ofile

Path to project output file.

Type
pathlib.Path

parse_dogmin()

Dogmin parser.

property prefsfile

Path to THERMOCALC prefs file.

Type
pathlib.Path

rundr()

Method to run drawpd.

runtc(*instr='kill\n\n'*)

Low-level method to actually run THERMOCALC.

Parameters
instr (*str*) – String to be passed to standard input for session.

Returns
THERMOCALC standard output

Return type
str

property scriptfile

Path to scriptfile.

Type
pathlib.Path

property tcnewversion

False for THERMOCALC older than 3.5.

Type
bool

property tcversion

THERMOCALC version string

Type
str

class pypsbuilder.tcapi.**TC34API**(*workdir*, *tcexe*, *drexe*, *encoding='mac-roman'*)

Bases: TCAPI

parse_logfile(***kwargs*)

Parser for THERMOCALC 3.4x output.

It parses the outputs of THERMOCALC after calculation.

Parameters

- **output** (*str*) – When not None, used as content of logfile. Default None.
- **resic** (*str*) – When not None, used as content of icfile. Default None.
- **get_phases** (*bool*) – When true returns also tuple (phases, out, calcs). Default False

Returns
Result of parsing. 'ok', 'nir' (nothing in range) or 'bombed'. results (TCResultSet): Results of TC calculation. output (str): Full nonparsed THERMOCALC output.

Return type
status (str)

Example

Parse output after univariant line calculation in P-T pseudosection:

```
>>> tc = TCAPI('pat/to/dir')
>>> status, result, output = tc.parse_logfile()
```

update_scriptfile(***kwargs*)

Method to update scriptfile.

This methodcould be used to read or update ptguess or dogmin settings.

Parameters

- **guesses** (*list*) – List of lines defining ptguesses. If None guesses are not modified. Default None.
- **get_old_guesses** (*bool*) – When True method returns existing ptguess before possible modification. Default False.
- **dogmin** (*str*) – Argument of dogmin script. Could be 'no' or 'yes' or 'yes X', where X is log level. When None no modification is done. Default None.
- **which** (*set*) – Set of phases used for dogmin.
- **bulk** (*list*) – Bulk composition. Default None.
- **xvals** (*tuple*) – x values for compositions. Default (0, 1)
- **xsteps** (*int*) – Number of compositional steps between two bulks. Default 20.
- **p** (*float*) – Pressure for dogmin calculation
- **T** (*float*) – Temperature for dogmin calculation

property axfile

Path to used a-x file.

Type
pathlib.Path

property csvfile

Path to csv file.

Type
pathlib.Path

property dataset

Version identification of thermodynamic dataset in use.

Type
str

property datasetfile

Path to dataset file.

Type
pathlib.Path

property drawpdfile

Path to drawpd file.

Type
pathlib.Path

property drfile

Path to -dr output file.

Type
pathlib.Path

property icfile

Path to ic file.

Type
pathlib.Path

property itfile

Path to it file.

Type
pathlib.Path

property logfile

Path to THERMOCALC log file.

Type
pathlib.Path

property ofile

Path to project output file.

Type
pathlib.Path

parse_dogmin()

Dogmin parser.

property prefsfile

Path to THERMOCALC prefs file.

Type
pathlib.Path

rundr()

Method to run drawpd.

runtc(*instr='kill\n\n'*)

Low-level method to actually run THERMOCALC.

Parameters
instr (*str*) – String to be passed to standard input for session.

Returns
THERMOCALC standard output

Return type
str

property scriptfile

Path to scriptfile.

Type
pathlib.Path

property tcnewversion

False for THERMOCALC older than 3.5.

Type
bool

property tcversion

THERMOCALC version string

Type
str

calc_t(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC to find univariant line using Calc T at P strategy.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of single zero mode phase
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_p(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC to find univariant line using Calc P at T strategy.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of single zero mode phase
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation

- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_pt(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC to find invariant point.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of two zero mode phases
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_tx(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC for T-X pseudosection calculations.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of zero mode phases
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

calc_px(*phases*, *out*, ***kwargs*)

Method to run THERMOCALC for P-X pseudosection calculations.

Parameters

- **phases** (*set*) – Set of present phases
- **out** (*set*) – Set of zero mode phases
- **prange** (*tuple*) – Temperature range for calculation
- **trange** (*tuple*) – Pressure range for calculation
- **steps** (*int*) – Number of steps

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

`calc_assemblage`(*phases*, *p*, *t*)

Method to run THERMOCALC to calculate compositions of stable assemblage.

Parameters

- **phases** (*set*) – Set of present phases
- **p** (*float*) – Temperature for calculation
- **t** (*float*) – Pressure for calculation

Returns
(tcout, ans) standard output and input for THERMOCALC run. Input ans could be used to reproduce calculation.

Return type
tuple

`dogmin`(*phases*, *p*, *t*, *variance*, *doglevel=1*, *onebulk=None*)

Run THERMOCALC dogmin session.

Parameters
variance (*int*) – Maximum variance to be considered

Returns
THERMOCALC standard output

Return type
str

`calc_variance`(*phases*)

Get variance of assemblage.

Parameters
phases (*set*) – Set of present phases

Returns
variance

Return type
int

5.3 psclasses module

InvPoint	Class to store invariant point
UniLine	Class to store univariant line
Dogmin	
TCResult	
TCResultSet	
PTsection	P-T pseudosection class
TXsection	T-X pseudosection class
PXsection	P-X pseudosection class

5.3.1 pypsbuilder.psclasses.InvPoint

class pypsbuilder.psclasses.**InvPoint**(***kwargs*)

Bases: PseudoBase

Class to store invariant point

id

Invariant point identification

Type
int

phases

set of present phases

Type
set

out

set of zero mode phases

Type
set

cmd

THERMOCALC standard input to calculate this point

Type
str

variance

variance

Type
int

x

Array of x coordinates (even if only one, it is stored as array)

Type
numpy.array

y

Array of x coordinates (even if only one, it is stored as array)

Type
numpy.array

results

List of results dicts with data and ptgues keys.

Type
list

output

Full THERMOCALC output

Type
str

manual

True when inavariant point is user-defined and not calculated

Type
bool

__init__(***kwargs*)

Methods

__init__(**kwargs)	
all_unilines()	Return four tuples (phases, out) indicating possible four univariant lines passing trough this invariant point
annotation([state])	str: String representation of ID with possible zermo mode phase.
datakeys([phase])	list: Get list of variables for phase.
label([excess])	str: full label with space delimeted phases - zero mode phase.
ptguess(**kwargs)	list: Get stored ptguesses.
shape()	Return shapely Point representing invariant point.

Attributes

midix

5.3.2 pypsbuilder.psclasses.UniLine

class pypsbuilder.psclasses.**UniLine**(***kwargs*)

Bases: PseudoBase

Class to store univariant line

id

Invariant point identification

Type
int

phases

set of present phases

Type
set

out

set of zero mode phase

Type
set

cmd

THERMOCALC standard input to calculate this point

Type
str

variance

variance

Type
int

_x

Array of x coordinates (all calculated)

Type
numpy.array

_y

Array of x coordinates (all calculated)

Type
numpy.array

results

List of results dicts with data and ptgues keys.

Type
list

output

Full THERMOCALC output

Type
str

manual

True when inavariant point is user-defined and not calculated

Type
bool

begin

id of invariant point defining begining of the line. 0 for no begin

Type
int

end

id of invariant point defining end of the line. 0 for no end

Type
int

used

slice indicating which point on calculated line are between begin and end

Type
slice

__init__(***kwargs*)

Methods

__init__(**kwargs)	
annotation([state])	str: String representation of ID with possible zermo mode phase.
contains_inv(ip)	Check whether invariant point theoretically belong to univariant line.
datakeys([phase])	list: Get list of variables for phase.
get_label_point()	Returns coordinate tuple of labeling point for univariant line.
label([excess])	str: full label with space delimeted phases - zero mode phase.
ptguess(**kwargs)	list: Get stored ptguesses.
shape([ratio, tolerance])	Return shapely LineString representing univariant line.

Attributes

connected
midix

5.3.3 pypsbuilder.psclasses.Dogmin

class pypsbuilder.psclasses.**Dogmin**(***kwargs*)

Bases: object

__init__(***kwargs*)

Methods

__init__(**kwargs)	
annotation([show_out, excess])	str: String representation of ID with possible zermo mode phase.
label([excess])	str: full label with space delimeted phases.
ptguess()	

Attributes

out

5.3.4 pypsbuilder.psclasses.TCResult

class pypsbuilder.psclasses.**TCResult**(*T*, *p*, *variance=0*, *c=0*, *data={}*, *ptguess=['']*)

Bases: object

__init__(*T*, *p*, *variance=0*, *c=0*, *data={}*, *ptguess=['']*)

Methods

__init__(T, p[, variance, c, data, ptguess])
from_block(block, ptguess)
rename_phase(old, new)

Attributes

phases

5.3.5 pypsbuilder.psclasses.TCResultSet

class pypsbuilder.psclasses.**TCResultSet**(*results*)

Bases: object

__init__(*results*)

Methods

__init__(results)
insert(ix, result)
ptguess(ix)
rename_phase(old, new)

Attributes

c
phases
variance
x
y

5.3.6 pypsbuilder.psclasses.PTsection

class pypsbuilder.psclasses.**PTsection**(***kwargs*)

Bases: SectionBase

P-T pseudosection class

__init__(***kwargs*)

Methods

__init__(**kwargs)	
add_dogmin(id, dgm)	
add_inv(id, inv)	
add_uni(id, uni)	
cleanup_data()	
create_shapes([tolerance])	
from_file(projfile)	
getidinv([inv])	Return id of either new or existing invariant point
getiduni([uni])	Return id of either new or existing univariant line
read_file(projfile)	
show()	
trim_uni(id)	

Attributes

range_shapes
ratio
type

5.3.7 pypsbuilder.psclasses.TXsection

class pypsbuilder.psclasses.**TXsection**(***kwargs*)

Bases: SectionBase

T-X pseudosection class

__init__(***kwargs*)

Methods

__init__(**kwargs)	
add_dogmin(id, dgm)	
add_inv(id, inv)	
add_uni(id, uni)	
cleanup_data()	
create_shapes([tolerance])	
from_file(projfile)	
getidinv([inv])	Return id of either new or existing invariant point
getiduni([uni])	Return id of either new or existing univariant line
read_file(projfile)	
show()	
trim_uni(id)	

Attributes

range_shapes
ratio
type

5.3.8 pypsbuilder.psclasses.PXsection

class pypsbuilder.psclasses.**PXsection**(***kwargs*)

Bases: SectionBase

P-X pseudosection class

__init__(***kwargs*)

Methods

__init__(**kwargs)	
add_dogmin(id, dgm)	
add_inv(id, inv)	
add_uni(id, uni)	
cleanup_data()	
create_shapes([tolerance])	
from_file(projfile)	
getidinv([inv])	Return id of either new or existing invariant point
getiduni([uni])	Return id of either new or existing univariant line
read_file(projfile)	
show()	
trim_uni(id)	

Attributes

range_shapes
ratio
type

class pypsbuilder.psclasses.**InvPoint**(***kwargs*)

Bases: PseudoBase

Class to store invariant point

id

Invariant point identification

Type
int

phases

set of present phases

Type
set

out

set of zero mode phases

Type
set

cmd

THERMOCALC standard input to calculate this point

Type
str

variance

variance

Type
int

x

Array of x coordinates (even if only one, it is stored as array)

Type
numpy.array

y

Array of x coordinates (even if only one, it is stored as array)

Type
numpy.array

results

List of results dicts with data and ptgues keys.

Type
list

output

Full THERMOCALC output

Type
str

manual

True when inavariant point is user-defined and not calculated

Type
bool

shape()

Return shapely Point representing invariant point.

all_unilines()

Return four tuples (phases, out) indicating possible four univariant lines passing trough this invariant point

annotation(*state=0*)

str: String representation of ID with possible zermo mode phase.

datakeys(*phase=None*)

list: Get list of variables for phase.

Parameters
phase (*str*) – name of phase

label(*excess={}*)

str: full label with space delimeted phases - zero mode phase.

ptguess(***kwargs*)

list: Get stored ptguesses.

InvPoint has just single ptguess, but for UniLine idx need to be specified. If omitted, the middle point from calculated ones is used.

Parameters
idx (*int*) – index which guesses to get.

class pypsbuilder.psclasses.**UniLine**(***kwargs*)

Bases: PseudoBase

Class to store univariant line

id

Invariant point identification

Type
int

phases

set of present phases

Type
set

out

set of zero mode phase

Type
set

cmd

THERMOCALC standard input to calculate this point

Type
str

variance

variance

Type
int

_x

Array of x coordinates (all calculated)

Type
numpy.array

_y

Array of x coordinates (all calculated)

Type
numpy.array

results

List of results dicts with data and ptgues keys.

Type
list

output

Full THERMOCALC output

Type
str

manual

True when inavariant point is user-defined and not calculated

Type
bool

begin

id of invariant point defining begining of the line. 0 for no begin

Type
int

end

id of invariant point defining end of the line. 0 for no end

Type
int

used

slice indicating which point on calculated line are between begin and end

Type
slice

shape(*ratio=None*, *tolerance=None*)

Return shapely LineString representing univariant line.

This method is using trimmed points.

Parameters

- **ratio** – y-coordinate multiplier to scale coordinates. Default None
- **tolerance** – tolerance x coordinates. Simplified object will be within
- **None** (*the tolerance distance of the original geometry. Default*)

contains_inv(*ip*)

Check whether invariant point theoretically belong to univariant line.

Parameters
ip (InvPoint) – Invariant point

Returns
True for yes, False for no. Note that metastability is not checked.

Return type
bool

get_label_point()

Returns coordinate tuple of labeling point for univariant line.

annotation(*state=0*)

str: String representation of ID with possible zermo mode phase.

datakeys(*phase=None*)

list: Get list of variables for phase.

Parameters
phase (*str*) – name of phase

label(*excess={}*)

str: full label with space delimeted phases - zero mode phase.

ptguess(***kwargs*)

list: Get stored ptguesses.

InvPoint has just single ptguess, but for UniLine idx need to be specified. If omitted, the middle point from calculated ones is used.

Parameters
idx (*int*) – index which guesses to get.

class pypsbuilder.psclasses.**Dogmin**(***kwargs*)

Bases: object

label(*excess={}*)

str: full label with space delimeted phases.

annotation(*show_out=False*, *excess={}*)

str: String representation of ID with possible zermo mode phase.

class pypsbuilder.psclasses.**TCResult**(*T*, *p*, *variance=0*, *c=0*, *data={}*, *ptguess=['']*)

Bases: object

class pypsbuilder.psclasses.**TCResultSet**(*results*)

Bases: object

class pypsbuilder.psclasses.**PTsection**(***kwargs*)

Bases: SectionBase

P-T pseudosection class

getidinv(*inv=None*)

Return id of either new or existing invariant point

getiduni(*uni=None*)

Return id of either new or existing univariant line

class pypsbuilder.psclasses.**TXsection**(***kwargs*)

Bases: SectionBase

T-X pseudosection class

getidinv(*inv=None*)

Return id of either new or existing invariant point

getiduni(*uni=None*)

Return id of either new or existing univariant line

class pypsbuilder.psclasses.**PXsection**(***kwargs*)

Bases: SectionBase

P-X pseudosection class

getidinv(*inv=None*)

Return id of either new or existing invariant point

getiduni(*uni=None*)

Return id of either new or existing univariant line

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Ondrej Lexa <lexa.ondrej@gmail.com>

6.2 Contributors

- None yet. Why not be the first?

THERMOCALC is a thermodynamic calculation program (Powell & Holland 1988) that uses an internally-consistent thermodynamic dataset (Holland & Powell, 1998, 2011) to undertake thermobarometry and phase diagram calculations for metamorphic rocks. However, using THERMOCALC to create a diagram is quite laborious; each curve must be calculated by hand, and the Schreinemaker's analysis must be done manually. The curves must be built up one by one, and manually combined.

pypsbuilder is developed with the idea to make this tedious process much easier and more enjoyable while keeping the concept to force users understand the Phase Rule, Schreinemaker's analysis, and how variance changes across field boundaries.

Check *Pseudosection builders tutorial* and *Pseudosection explorers tutorial* or watch video below to see **pypsbuilder** in action.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

X

Y